International Journal of Computational Intelligence Systems
Vol. **13(1)**, 2020, *pp.* 98–129
DOI: https://doi.org/10.2991/ijcis.d.200127.001; ISSN: 1875-6891; eISSN: 1875-6883
https://www.atlantis-press.com/journals/ijcis/

Enhanced Particle Swarm Optimization Based on Reference Direction and Inverse Model for Optimization Problems

Wei Li[1,2,*], Yaochi Fan[1], Qingzheng Xu[3]

[1] *School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China*
[2] *Shaanxi Key Laboratory for Network Computing and Security Technology, Xi'an 710048, China*
[3] *College of Information and Communication, National University of Defense Technology, Xi'an 710106, China*

ARTICLE INFO

Article History

Received 05 Oct 2019
Accepted 22 Jan 2020

Keywords

Particle swarm optimization
Reference direction
Non-dominated sorting
Inverse model
Neural networks optimization

ABSTRACT

While particle swarm optimization (PSO) shows good performance for many optimization problems, the weakness in premature convergence and easy trapping into local optimum, due to the ignorance of the diversity information, has been gradually recognized. To improve the optimization performance of PSO, an enhanced PSO based on reference direction and inverse model is proposed, RDIM-PSO for short reference. In RDIM-PSO, the reference particles which are used as reference directions are selected by non-dominated sorting method according to the fitness and diversity contribution of the population. Dynamic neighborhood strategy is introduced to divide the population into several sub-swarms based on the reference directions. For each sub-swarm, the particles focus on exploitation under the guidance of local best particle with a good guarantee of population diversity. Moreover, Gaussian process-based inverse model is introduced to generate equilibrium particles by sampling the objective space to further achieve a good balance between exploration and exploitation. Experimental results on CEC2014 test problems show that RDIM-PSO has overall better performance compared with other well-known optimization algorithms. Finally, the proposed RDIM-PSO is also applied to artificial neural networks and the promising results on the chaotic time series prediction show the effectiveness of RDIM-PSO.

1. INTRODUCTION

With the development of science and technology, there are more and more optimization problems encountered in science and engineering fields. Without loss of generality, a numerical optimization problem can be defined as follows.

$$\min f(\mathbf{x}), \mathbf{x} = (x_1, x_2, x_3, \cdots, x_D) \tag{1}$$

where D denotes the dimensions of the decision variables. In recent years, a lot of studies have been done for solving the optimization problems effectively and efficiently. In particular, biologically inspired computing has attracted a great interest of researchers. Computer scientists turned to biological processes for inspiration and designed optimization algorithms [1–2].

Evolution-based, physics-based and swarm intelligence-based methods [3] are three main categories for nature-inspired optimization algorithms. Genetic algorithm (GA) [4], genetic programming (GP) [5], differential evolution (DE) [6] and derandomized evolution strategy with covariance matrix adaptation (CMA-ES) [7], can be seen as evolution-based optimization algorithms because they are inspired by the laws of biological evolutionary process. Simulated annealing (SA) [8], chemical reaction optimization (CRO) [9], fireworks algorithm (FA) [10] and brain storm optimization (BSO) [11], can be seen as physics-based optimization algorithms because they imitate the physical or natural phenomena in the universe. Particle swarm optimization (PSO) [12], artificial bee colony (ABC) [13], teaching–learning-based optimization (TLBO) [14] and whale optimization algorithm (WOA) [3], can be seen as swarm intelligence-based methods because they mimic the social behavior of all kinds of animals.

In 1995, Kennedy and Eberhart [12] introduced an effective optimizer called PSO. The algorithm mimics the flocking behavior of birds. The PSO method is successfully applied in solving real-world engineering problems because of its better computational efficiency. However, rapid convergence and diversity loss make traditional PSO encounter premature convergence for solving complex optimization problems [15]. Moreover, how to achieve a fine balance between exploration and exploitation remains a big challenge. Over the past decades, a number of methods have been developed for speeding up convergence and alleviate premature convergence. Recently, the popular methods are inverse modeling, reference directions, cooperative approach [16], social learning approach [17], etc. The inverse modeling approach [18] creates offspring by sampling the objective space to improve the search performance of the algorithm. The reference direction method splits the objective space into a number of independent subregions, helping guide the search toward the optimal solution with a good guarantee

*Corresponding author. Email: liwei@xaut.edu.cn of population diversity [19]. To our knowledge, inverse modeling approach and the reference direction method are mainly used to solve multiobjective optimization problems. Therefore, this paper attempts to apply inverse model and reference direction to improve the performance of PSO (RDIM-PSO) for solving single-objective optimization problems.

The main contributions of this paper are summarized as follows:

1. Non-dominated sorting mechanism which takes into account the population fitness and diversity contribution is introduced to select reference particles during optimization search. The main purpose of doing so is to construct a dynamic neighborhood model which encourages the RDIM-PSO to explore the local optimum area and the sparse region.
2. Inverse model is introduced to produce equilibrium particles at each generation, which helps the population balance the exploration and exploitation abilities effectively.
3. A new velocity updating strategy is proposed, whose major role is to achieve fast and accurate convergence.
4. Systematic experiments conducted to compare RDIM-PSO with seven state-of-the-art evolutionary algorithms (EAs) on CEC2014 test problems and three application problems of artificial neural network (ANN) are described. The experimental results show that the proposed method is promising for solving optimization problems.

The remainder of this paper is organized as follows. In Section 2, the basic idea of PSO and related work are reviewed. In Section 3, the proposed RDIM-PSO is presented in detail. Section 4 reports and discusses the experimental results. In Section 5, RDIM-PSO is applied to three application problems of ANN. Finally, the conclusions and possible future research are drawn up in Section 6.

2. PSO AND RELATED WORK

2.1. PSO Algorithm

In PSO algorithm, each particle i represents a potential solution. The velocity $\mathbf{v}_i = (v_{i1}, v_{i2}, \cdots, v_{iD})$ and the position $\mathbf{x}_i = (x_{i1}, x_{i2}, \cdots, x_{iD})$ of the ith particle are updated as follows. D is the dimensions of the problem.

$$v_{id} = v_{id} + c_1 r_{1d}\left(pbest_{id} - x_{id}\right) + c_2 r_{2d}\left(gbest_d - x_{id}\right) \tag{2}$$

$$x_{id} = x_{id} + v_{id} \tag{3}$$

where c_1 and c_2 are acceleration coefficients which are commonly set to 2.0. The r_{1d} and r_{2d} are random numbers generated in the interval [0, 1] for the dth dimension. $pbest_i = \left(pbest_{i1}, pbest_{i2}, \cdots, pbest_{iD}\right)$ is the historical best position for ith particle. $gbest = \left(gbest_1, gbest_2, \cdots, gbest_D\right)$ is the best historical position for the whole population.

To restrict the change of velocities, Shi and Eberhart introduced an inertia weight [20].

$$v_{id} = \omega v_{id} + c_1 r_{1d}\left(pbest_{id} - x_{id}\right) + c_2 r_{2d}\left(gbest_d - x_{id}\right) \tag{4}$$

The inertia weight ω is computed as follows:

$$\omega = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{MaxGen} G \tag{5}$$

where ω_{max} and ω_{min} are usually fixed as 0.9 and 0.4. G and $MaxGen$ are the current generation and maximum generation, respectively. The pseudo code of PSO is shown in Algorithm 1.

Algorithm 1: Particle Swarm Optimization (PSO)

```
1: Initialize parameters and population
2: Evaluate the population
3: Compute $pbest_i$ ($i$ =1,2,⋯, $PopSize$) and $gbest$
4: while the termination criteria are not met do
5: for $i$ = 1 to $PopSize$ do
6: Update $\mathbf{v}_i$ and $\mathbf{x}_i$
7: Evaluate $\mathbf{x}_i$
8: Update $pbest_i$ and $gbest$
9: end for
10: end while
Output: the particle with the smallest objective function value in the population.
```

2.2. Related Work

Traditional PSO cannot effectively solve complex optimization problems. Then, variants of modified PSOs are proposed to improve the performance of PSO. The improvements can be divided into three categories.

1. Improvements of the control parameters [20–24]. Inertia weight is introduced in [20] to restrict the change of velocities and control the scope of search. To guarantee convergence and encourage exploration, PSO with constriction factor (PSOcf) is designed in [21]. Nobile *et al.* [22] introduces fuzzy self-turning PSO (FST-PSO) which uses fuzzy logic to obtain the control parameters used by each particle. Multiple adaptive methods are introduced into PSO in [23] to adaptively control the parameters. A chaotic map and dynamic weight are introduced in PSO to modify the search process [24].
2. Improvements of the evolutionary learning strategy [25–31]. The learning strategy algorithm (CLPSO) is designed to adjust the velocity of the particles and solve premature in [25]. To guide the particles to fly in better directions, a much promising and efficient exemplar is constructed by introducing orthogonal learning strategy (OLPSO) in [26]. Quyang *et al.* [27] presents the idea of an improved global-best-guided (IGPSO) to enhance the exploitation ability of the algorithm. In IGPSO, optimization strategies include the global neighborhood strategy, the local learning strategy, stochastic learning strategy and opposition based learning strategy are employed to improve the optimization potential of PSO. Two kinds of PSO, namely a surrogate-assisted PSO and a surrogated-assisted social learning-based PSO, worked together to search the global optimum [28]. To ensure the swarm diversity and fast convergence, the idea of all-dimension-neighborhood-based and randomly selected neighbors learning strategy is introduced in PSO (ADN-RSN-PSO) [29]. For updating particle velocity, Wang *et al.* employed a dynamic tournament

topology strategy [30] while Jensi *et al.* introduced a levy flight method [31].

3. Neighborhood-based strategies [32–38]. Neighborhood strategy has been widely used because it can effectively improve the performance of EAs. An index-based neighborhood strategy is employed in fully informed PSO [32]. The neighborhood operator, which is used to adjust the local neighborhood size, is introduced in [33]. A dynamic neighborhood learning strategy (DNLPSO), proposed by Nasir *et al.*, is introduced in [34]. In DNLPSO, the exemplar particle is selected from a neighborhood, and the learner particle learns the historical information from its neighborhood. The idea of multi-swarm and dynamic learning strategy (PSO-DLS) is introduced in [35]. In PSO-DLS, the population is divided into several sub-swarms. In each sub-swarm, the task of ordinary particles is to search a better *lbest*. The cyclic neighborhood topology strategy (CNT-CPSO), proposed by Maruta *et al.*, is introduced in [36]. Specifically, the cyclic-network topology is employed to construct neighborhood structure for each particle. The idea of cluster is employed to enhance the performance of PSO (*pk*PSO) in [37]. In [37], the particles, which are clustered on the Euclidean spatial neighborhood structure, explore the promising areas and obtain better solutions. A hybrid PSO algorithm (DNSPSO), which employs two strategies, namely diversity enhancing strategy and neighborhood strategy, is proposed in [38]. A trade-off between exploration and exploitation abilities is achieved with the two strategies.

3. ENHANCED PSO BASED ON REFERENCE DIRECTION AND INVERSE MODEL

It is known that better optimization performance can be achieved by the proper balance between global exploration and local exploitation. In order to further improve the performance of PSO, three major changes are proposed in RDIM-PSO. The first change is the construction of the dynamic neighborhood model. The diversity and the fitness of the particle are considered as a two-objective problem. Then, the non-dominated sorting mechanism is employed to select the reference particles which are considered as the reference directions. Next, the population is partitioned into several independent sub-swarms (neighborhood) with reference particles. The second change is the utilization of the inverse model. The samples are generated with reference particles in the objective space. The inverse models are employed to map the samples back to the decision space. The equilibrium particles which are generated by the inverse models can balance the exploration and exploitation. The last change is the introduction of a new velocity update strategy which is used to improve the performance of the proposed algorithm.

3.1. Dynamic Neighborhood Strategy

In the classical PSO, the personal best experience (*pbest*) and the global best experience (*gbest*) are used to adjust the flying trajectory of each particle. However, if the current *gbest* is far from the global optimum, particles may easily get trapped in a local optimum because of misleading from the current *gbest* [25]. In order to achieve better performance, PSO should establish a good balance between exploration and exploitation. Related works show that the multi-swarm can discourage premature convergence and maintain the diversity in some extent [16,34,39]. There are various types of topology can be used for the implementation of the neighborhood, such as star, wheel, circular, pyramid and 4-clusters [21]. The division of sub-swarm is usually based on index or fitness values [24,35,40,41]. In RDIM-PSO, non-dominated sorting procedure which comes from non-dominated sorting genetic algorithm II (NSGA-II) [42] is employed to realize the dynamic division of neighborhood.

NSGA-II is a classical multiobjective optimization algorithm. It is necessary to note that the conflicting objectives are the prerequisite for the multiobjective optimization problems. The major problem of the optimization algorithm is the balance between the exploration and exploitation. However, the exploration and the exploitation are the conflicting objectives. Generally, the exploration efficiency is depended on the distribution of particles, namely population diversity, while the exploitation efficiency is associated with the fitness of particles. Then, diversity and fitness can be used to indirectly measure the exploration and exploitation, respectively. In view of this idea, NSGA-II can be employed after constructing two objectives, diversity and fitness.

The diversity measures reported in the literature can be largely classified into two categories, namely the diversity contribution measure on whole population level and on individual level, respectively. Here, the diversity for a single particle to whole population is defined based on Euclidean distance.

$$\text{diversity}\left(\mathbf{x}_i\right) = \frac{\|\mathbf{x}_i - \bar{\mathbf{x}}\|}{max_{1\le i,j\le N}\|\mathbf{x}_i - \mathbf{x}_j\|} \tag{6}$$

where N is the population size, $\|\mathbf{x}_i - \mathbf{x}_j\| = \sqrt{\sum_{k=1}^{D}\left(x_{i,k} - x_{j,k}\right)^2}$, and D denotes the dimensionality of the search space. $\bar{\mathbf{x}}$ denotes the mean position which is defined as

$$\bar{\mathbf{x}} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{x}_i \tag{7}$$

As mentioned before, two conflicting objectives should be constructed before employing non-dominated sorting procedure. The better the diversity of population is, the stronger the ability of exploration is. The better the fitness of the population is, the stronger the exploitation ability is. In this paper, population diversity is directly proportional to the diversity value, while population fitness is inversely proportional to the objective function value. In order to utilize the non-dominated sorting procedure, the objective functions are transformed into the minimization optimization problem. In addition, in order to ensure that the values of the fitness and the diversity are in the same range, two objectives, fitness and diversity, associated with each particle in the population can be calculated as

$$\text{Minimize } Fitness\left(\mathbf{x}_i\right) = F\left(\mathbf{x}_i\right) = \frac{f\left(\mathbf{x_i}\right)}{max\left(f\left(\mathbf{x_i}\right)\right) + \varepsilon} \tag{8}$$

$$\text{Minimize } Diversity\left(\mathbf{x}_i\right) = D\left(\mathbf{x}_i\right) = \frac{1}{\text{diversity}\left(\mathbf{x}_i\right)} \tag{9}$$

where $f(\mathbf{x}_i)$ is the objective function value of ith particle. The objective function value is considered as the fitness value in this paper. $max(f(\mathbf{x}_i))$ is the maximum function value of all the particles, $i = 1, 2, \cdots, N$. $D(\mathbf{x}_i)$ is the contribution of diversity of ith particle to the population. ε is set to 10^{-6}.

The reference particles are selected by non-dominated sorting procedure. The motivation behind the idea of applying the selective strategy on the reference particles is that diversity and fitness can be employed to achieve a trade-off between exploration and exploitation [43]. Details of the fast-non-dominated-sort algorithm can be found in [42]. Figure 1 shows the output of non-dominated sorting procedure. The particles in the lowest layer belong to front 1. The reference particles $\boldsymbol{v}_j$ are generated as follows:

$$\boldsymbol{v}_j = \frac{\boldsymbol{x}_j}{||\boldsymbol{x}_j||}\, j = 1, 2, \cdots K \tag{10}$$

where $\mathbf{x}_j$ is the jth particle from front 1. K is the number of particles in front 1.

In RDIM-PSO, the reference particles which have a good guarantee of population diversity are employed to divide the population into several independent sub-swarms, namely neighborhoods. Each particle is associated with a specific reference particle according to its positions in the search space. More exactly, an arbitrary particle $\boldsymbol{x}_i$ is associated with a reference particle $\boldsymbol{v}_j$ if and only if the acute angle θ_i between its position in search space and $\boldsymbol{v}_j$ is the maximum among all reference particles

$$\theta_i = argmax\left(\frac{\boldsymbol{x}_i}{||\boldsymbol{x}_i||} \times \boldsymbol{v}_j\right), j = 1, 2, \cdots, K \tag{11}$$

where operator $\times$ calculates the cosine function value of the acute angle between $\boldsymbol{x}_i/||\boldsymbol{x}_i||$ and $\boldsymbol{v}_j$. Figure 2 shows particle $\mathbf{x}_i$ is associated with reference particle $\mathbf{v}_1$ since $\theta_1 < \theta_2$.

3.2. Equilibrium Particles Generation Strategy

As mentioned above, the reference particles have a good guarantee of population diversity. Then, a Gaussian process-based inverse model [18] is constructed to map all reference particles found so far from the objective space to the decision space at every generation. Like reference particles, the mapped particles which called equilibrium particles have better fitness and diversity. At each generation,

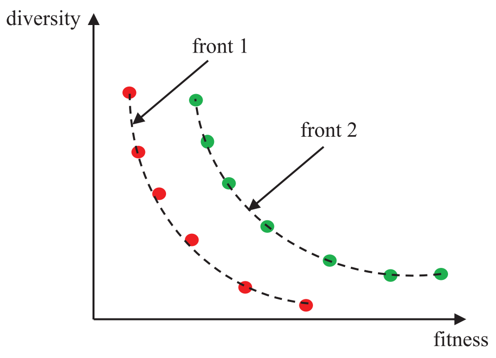


Figure 1 Output of the non-dominated sorting procedure.

randomly selected particles are replaced by the equilibrium particles. In doing so, the exploration and exploitation abilities can be balanced effectively.

Details of Gaussian process-based inverse model can be found in [18]. First, the reference particles are used for training two groups of inverse models. The training data set for training the inverse model $\left(\mathrm{P}\left(x_i|f_k\right)\right)$ in the kth group is denoted as $T_{k,i}$

$$T_{k,i} = [\mathbf{f}_k, \mathbf{x}_i] \tag{12}$$

where $\mathrm{f}_k = \left(f_k^1, \cdots, f_k^K\right)^{\mathrm{T}}$ and $\mathrm{x}_i = \left(x_i^1, \cdots, x_i^K\right)^{\mathrm{T}}$, f_k^j, x_i^j are the kth objective values and ith decision value of the jth particle, $1 \le j \le K$ that are randomly allocated to the kth group of models.

Second, the test input $\mathbf{f}_{k,*}$ is directly sampled in the objective space based on the estimated range of $\mathbf{f}_k$. Moreover, without loss of generality, the test input $\mathbf{f}_{k,*}$ is uniformly generated within an interval extended from $\left[f_k^{min}, f_k^{max}\right]$ as

$$\mathbf{f}_{k,*} = \left\{\frac{\left[f_k^{min} - 2\xi_k, f_k^{max} - 2\xi_k\right]}{N_p}\right\} \tag{13}$$

where N_p is the sample size, f_k^{min} and f_k^{max} are the minimum and maximum values among the components in the training data $\mathbf{f}_k$, respectively, and $\xi_k = \frac{f_k^{max} - f_k^{min}}{2}$. Figure 3 shows the Gaussian process-based inverse model. The purpose of training is to regress the inverse mapping from objective $\mathbf{f}_k$ to decision variable $\mathbf{x}_i$ given the training data $T_{k,i}$. The curve in the middle of the figure consists of the mean values $\mu_{k,i}$ of the predictions on the test input points uniformly distributed within $\left[f_k^{min} - 2\xi_k, f_k^{max} - 2\xi_k\right]$, refer to (13),

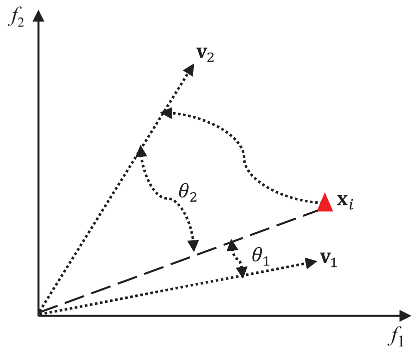


Figure 2 Relation between a particle and a reference particle.

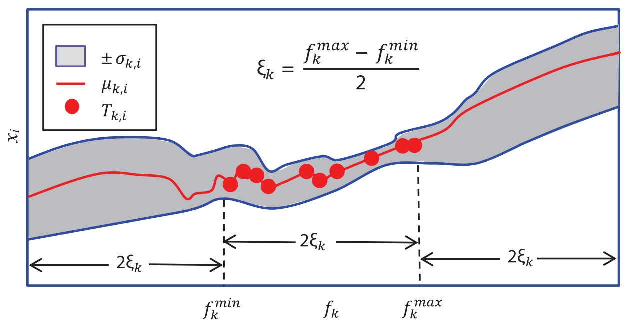


Figure 3 Gaussian process-based inverse model.

and the dashed area covers the length of $\pm\sigma_{k,j}$ around the mean value curve $\mu_{k,i}$, which reflects the uncertainty of the prediction on each test input point.

Finally, the test input $\mathbf{f}_{k,*}$ which is uniformly generated in the objective space can be mapped to the decision space $\mathbf{x}_{i,*}$ after the inverse models learned by the Gaussian process are available. $\mathbf{x}_{i,*}$ can be obtained by

$$\mathbf{x}_{i,*} = \mu_{k,i}\left(\mathbf{f}_{k,*}\right) + \mathbf{z}_{k,i} \tag{14}$$

where $\mu_{k,i} = \left(\mu_{k,i}^{1}, \cdots, \mu_{k,i}^{N_p}\right)$ is the mean value and $\mathbf{z} \sim N\left(0, (\sigma_n)^2 I\right)$, is the Gaussian white noise, and N_p is the sample dataset size. I is an identity matrix. The newly generated equilibrium particle will replace the particle selected randomly in the current population.

3.3. Velocity Update Strategy

As mentioned above, the dynamic neighborhood strategy based on reference particles is employed to divide the population into different sub-swarms. The sub-swarm guided by the reference particle which has better fitness but worse diversity undertakes the task of exploitation. The sub-swarm guided by the reference particle which has worse fitness but better diversity is responsible for the exploration task. The sub-swarm guided by the reference particle which has medium fitness and diversity contributes to balancing between exploration and exploitation. The velocity of RDIM-PSO is updated according to the following equation:

$$\begin{aligned} v_{id} = {} & \omega v_{id} + f_1 r_{1d}\left(pbest_{id} - x_{id}\right) \\ & + r_{2d}\left(gbest_d - x_{id}\right) + f_2 r_{3d}\left(nbest_{id} - x_{id}\right) \end{aligned} \tag{15}$$

where r_{1d}, r_{2d} and r_{3d} are random numbers generated in the interval [0, 1] for the dth dimension. $nbest_{id}$ denotes the best particle achieved so far in the ith sub-swarm. $f_1 = f_2 = 0.5$. The position of RDIM-PSO is updated according to Eq. (2). At each generation, l particles are randomly selected to be replaced by the offspring generated by the inverse model when conditions are met.

3.4. Complexity Analysis of RDIM-PSO

In RDIM-PSO, the computational costs involve the initialization (T_i), evaluation (T_e), non-dominated sorting procedure (T_s), velocity (T_v), Gaussian process-based inverse model (T_{gp}) and position (T_p), for each particle. As mentioned in [43], the time complexity of the non-dominated sorting procedure is $O(M \times N^2)$, where M is the number of objectives and N is the population size. Therefore, the total time complexity of RDIM-PSO can be estimated as follows:

$$\begin{aligned} T(N) &= T_i + \left(T_e + T_s + T_v + T_{gp} + T_p\right) \times D \\ &= N \times D + \left(2N^2 + 4N\right) \times D \\ &= \left(2N^2 + 5N\right) \times D \end{aligned} \tag{16}$$

D represents the dimension of the search space. Therefore, the time complexity of the proposed algorithm is $O\left(N^2 \times D\right)$. The complexity of PSO-w, CPSO-H, CLPSO, SLPSO TLBO, CSO and Jaya is $O(N \times D)$. Accordingly, RDIM-PSO improves the performance at the expense of the complexity.

Based on the above explanation, the pseudo-code of the RDIM-PSO algorithm is illustrated in Algorithm 2. *rand* denotes a random number from (0, 1]. *rand* (K) denotes K random numbers from (0, 1]. $\mathbf{x}^*$ denotes a particle reproduced with inverse model.

Algorithm 2: RDIM-PSO algorithm

```
1: Initialize $D$(number of dimensions), $N$
2: Initialize position $\mathbf{x}_i$, velocity $\mathbf{v}_i$ of the $N$ particles ($i = 1, 2, \cdots, N$)
3: Compute the objective function value and diversity of each particle according to (8) and (9)
4: Set **pbest**$_i$ and **gbest** of the $N$ particles ($i = 1, 2, \cdots, N$)
5: Find the particles in the first non-dominated front with the non-dominated sorting procedure
6: Compute reference particles according to (10)
7: Partition the population by $K$ reference particles according to (11)
8: Inverse Modeling: build inverse models and training a Gaussian process for each inverse model according to (12) and (13)
9: Reproduction: reproduce $K$ particles by sampling the objective space and mapping them back to the decision space using the inverse models according to (14)
10: **while** the termination criteria are not met **do**
11: Compute inertia weight $\omega$ according to (5)
12: Set **nbest**$_j$ of the $K$ sub-swarms ($j = 1, 2, \cdots, K$)
13: **for** $i = 1$ to $N$ do
14: Update the velocity $\mathbf{v}_i$ according to (15)
15: Update the position $\mathbf{x}_i$ according to (3)
16: **end for**
17: **if** $rand <= 1/N$
18: index = $\lfloor rand(K) \times N \rfloor$
19: $\mathbf{x}_{index} = \mathbf{x}_k^* \ (k = 1, 2 \cdots, K)$
20: **end if**
21: Calculate objective function value of each particle
22: **for** $i = 1$ to $N$ do
23: **if** $\mathbf{x}_i$ is better than pbest$_i$
24: pbest$_i$ = $\mathbf{x}_i$
25: **end if**
26: **if** $\mathbf{x}_i$ is better than gbest
27: gbest= $\mathbf{x}_i$
28: **end if**
29: **end for**
30: Compute the diversity of each particle according to (9)
31: Find the particles in the first non-dominated front with the non-dominated sorting procedure
32: Compute reference particles according to (10)
33: Partition the population by $K$ reference particles according to (11)
34: Inverse Modeling: build inverse models and training a Gaussian process for each inverse model according to (12) and (13)
35: Reproduction: reproduce $K$ particles by sampling the objective space and mapping them back to the decision space using the inverse models according to (14)
36: **end while**
**Output**: **the particle with the smallest objective function value in the population.**
```

To evaluate the performance of several components introduced in RDIM-PSO, RDIM-PSO is compared with PSO-N, PSO-NV and PSO-w on f_3, f_4, f_{17} and f_{29} with $D = 30$. Based on PSO-w, we introduce dynamic neighborhood strategy to achieve PSO-N. Based on PSO-N, we introduce velocity update strategy to achieve PSO-NV. Based on PSO-NV, we introduce Gaussian process-based inverse model to achieve RDIM-PSO. Four representative problems including unimodal problem (f_3), simple multimodal problem (f_4), hybrid problem (f_{17}) and the composite problem (f_{29}) are selected. The convergence curves in Figure 4 illustrate mean errors (in logarithmic scale) in 30 independent simulations of PSO-w, PSO-N, PSO-NV and RDIM-PSO. The mean error is defined as $f(\mathrm{x})$-$f(\mathrm{x}^*)$. $f(\mathrm{x})$ is the best fitness value and $f(\mathrm{x}^*)$ is the real global optimization value. The termination criterion is decided by the maximal number of function evaluations (*MaxFES*), which is set to 30,000.

Figure 4 shows that PSO-N performs worse on unimodal problem compared PSO-w, while it performs better on simple multimodal problem and composite problem. The reason is that the neighborhood strategy is suitable for solving these problems. The velocity update strategy can improve the convergence rate. Then, PSO-NV converge faster than PSO-N and PSO-w on f_3,f_4,f_{17} and f_{29}. Gaussian process-based inverse model is helpful for balancing the exploration and exploitation. Then, RDIM-PSO performs better than these compared algorithms because of introducing three improvement strategies into PSO.

4. EXPERIMENTS AND DISCUSSIONS

In this section, CEC2014 contest benchmark problems, which are widely adopted in numerical optimization methods, are used to verify the performance of RDIM-PSO. The general experimental setting is explained in Section 4.1. The experimental results and comparison with other algorithms are explained in Section 4.2.

4.1. General Experimental Setting

Thirty benchmark problems proposed in the special session on real-parameter optimization of CEC2014 [35] are used to evaluate the performance of RDIM-PSO in our experiments. The search ranges of all the test problems are $[-100, 100]^D$. CEC2014 contest benchmark problems are divided into four groups according to their diverse characteristics. In this paper, f_i denotes the ith benchmark problem, that is, optimization function. The first group includes three unimodal problems $f_1 - f_3$. The second group includes 13 simple multimodal problems $f_4 - f_{16}$. The third group includes six hybrid problems $f_{17} - f_{22}$. The fourth group includes eight composite problems $f_{23} - f_{30}$. The details of these benchmark problems can be found in [35]. In this paper, seven algorithms are selected to compare with the proposed algorithm. These algorithms can be divided into three categories. PSO-w, SLPSO belongs to the category of improving the control parameters. CLPSO, TLBO, CSO and Jaya belong to the category of improving the evolutionary learning strategy. CPSO-H belongs to the category of neighborhood-based strategies.

The compared algorithms and parameters settings are listed below:

- PSO with inertia weight (PSO-w) [20];
- Cooperative particle swarm optimizer (CPSO-H) [16];
- Comprehensive learning particle swarm optimizer (CLPSO) [25];
- Social learning PSO (SLPSO) [17];
- Teaching–learning-based optimization (TLBO) [14];
- Competitive swarm optimizer (CSO) [44];
- Jaya (a Sanskrit word meaning victory) algorithm [45];
- RDIM-PSO.

The performance of the algorithm can be affected by its population size. Generally, larger population is needed to enhance exploration at the beginning of the search, while at the end of the search, smaller population is needed to enhance exploitation. However, it is well-known that the loss of population diversity may result into premature convergence while and excessive diversity may cause stagnation [46]. In this paper, the population size of each compared algorithm is refer to the original paper. For PSO-w, CPSO-H and SLPSO, the population size is set to 20 [16,17,20]. For CLPSO, Jaya and RDIM-PSO, the population size is set to 40 [25,45]. For TLBO and CSO, the population size is set to 100 [14,44]. To make a fair comparison, the termination criterion for all algorithms is decided by the maximal number of function evaluations (*MaxFES*), which is set to $D \times 10{,}000$. The dimension (D) is set to 30, 50 and 100, respectively.

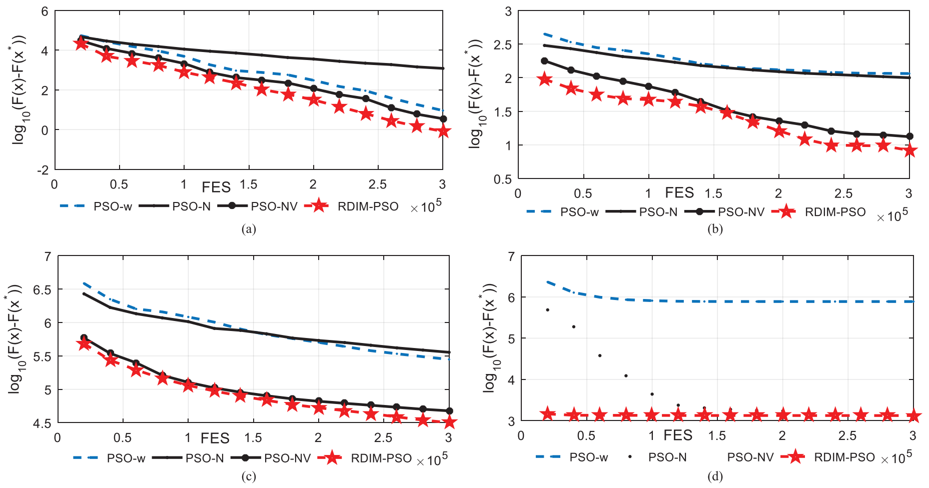


Figure 4 | Evolution of the mean function error values derived from PSO-w, PSO-N, PSO-NV and RDIM-PSO *versus* the number of *FES* on four test problems with $D = 30$. (a) f_3; (b) f_4; (c) f_{17}; (d) f_{29}.

Furthermore, the metrics, such as F_{mean} (mean value), *SD* (standard deviation), *Max* (maximum value) and *Min* (minimum value) of the *solution error measure* [47], are used to appraise the performance of each algorithm. The *solution error measure* is defined as $f(x)–f(x^*)$. $f(x)$ is the best fitness value and $f(x^*)$ is the real global optimization value. The smaller the solution error is, the better the solution quality is. To make a clear comparison, the error is assumed to be 0 when the value $f(x)–f(x^*)$ is less than 10^{-8}. In view of statistics, the Wilcoxon signed-rank test [48] at the 5% significance level is used to compare RDIM-PSO with other compared algorithms. "≈", "−" and "+" are applied to express the performance of the compared algorithm is similar to, worse than, and better than that of RDIM-PSO, respectively.

To obtain an unbiased comparison, all the experiments are run on a PC with an Intel Core i7-3770 3.40 GHz CPU and 4 GB memory. All experiments were run 30 times, and the codes are implemented in Matlab R2013a.

4.2. Comparison with Eight Optimization Algorithms

Before comparing with other algorithms, f_1 is used as an example to show the major process of the proposed algorithm. We select the first two dimensions of the particles for clarity. First, initialize the population. For each generation cycle, we sort the population according to the fitness and diversity of each particle using the non-dominated sorting procedure. The reference particles is generated according to formula (10). Figure 5(a) shows the first two dimensions of reference particles. Next, the population is divided into several neighborhoods by these reference particles. The neighborhood size is equal to the number of the reference particles. The inverse models are built and a Gaussian process is trained for each inverse model. For each reference particle, an offspring is reproduced by sampling the objective space and mapping them back to the decision space using the inverse model. Figure 5(b) shows the offspring generated by the inverse model. In addition, for each sub-swarm, the best particle is selected as *nbest*. Then, the velocity and the position of each particle are updated. When the given criteria is satisfied, the randomly selected particles are replaced by the offspring which are generated by the inverse models. Next, the *pbest* for each particle and the *gbest* are updated. Figure 5(c) shows the updated particle (*pbest*). The optimal solution is achieved when the termination criteria are met.

(1) *Unimodal problems* f_1–f_3

From the statistical results of Table 1, we can see that RDIM-PSO performs well on f_1–f_3 when D = 30. In case of f_1–f_3, the results show that RDIM-PSO significantly outperforms PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 2, 3, 3, 3, 3, 3 and 3 test problems, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, PSO-w, TLBO, CSO, CPSO-H (CLPSO), SLPSO and Jaya in a descending direction. The average rank of CPSO-H is the same as that of CLPSO. Table 2 shows that RDIM-PSO achieves better results on f_1–f_3 under D = 50. RDIM-PSO outperforms PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 3, 3, 3, 3, 3, 3 and 3 test problems, respectively.

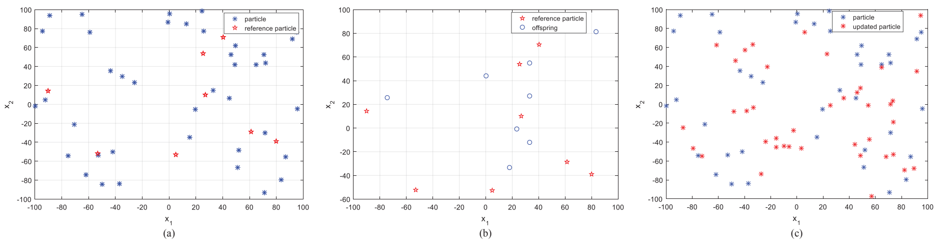


Figure 5 Major optimization process of RDIM-PSO.

Table 1 Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_1–f_3 with D = 30.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
	f_1	−/4	−/5	−/6	−/7	−/3	−/2	−/8	√/1
Compare/rank	f_2	≈/1	−/5	−/6	−/7	−/3	−/4	−/8	√/1
	f_3	−/2	−/6	−/4	−/7	−/3	−/5	−/8	√/1
−/≈/+		2/1/0	3/0/0	3/0/0	3/0/0	3/0/0	3/0/0	3/0/0	\
Avg-rank		2.33	5.33	5.33	7.00	3.00	3.67	8.00	1.00

Table 2 Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_1–f_3 with D = 50.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
	f_1	−/4	−/5	−/6	−/7	−/3	−/2	−/8	√/1
Compare/rank	f_2	−/5	−/4	−/6	−/7	−/3	−/2	−/8	√/1
	f_3	−/2	−/6	−/4	−/7	−/3	−/5	−/8	√/1
−/≈/+		3/0/0	3/0/0	3/0/0	3/0/0	3/0/0	3/0/0	3/0/0	\
Avg-rank		3.67	5.00	5.33	7.00	3.00	3.00	8.00	1.00

The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, TLBO (CSO), PSO-w, CPSO-H, CLPSO, SLPSO and Jaya in a descending direction. The average rank of TLBO is the same as that of CSO. Table 3 shows that RDIM-PSO outperforms PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 2, 2, 2, 3, 2, 2 and 3 test problems under $D = 100$, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, TLBO, PSO-w (CSO), CPSO-H (CLPSO), SLPSO and Jaya in a descending direction. The average rank of PSO-w is the same as that of CSO. The average rank of CPSO-H is the same as that of CLPSO. For solving unimodal problems, more exploitation is beneficial. RDIM-PSO employs the reference directions to divide the population into several sub-swarms. Each sub-swarm effectively employs neighborhood information to improve local exploitation ability. Thus, RDIM-PSO can find better results for unimodal problems compared with other optimization algorithms.

(2) *Simple multimodal problems* f_4–f_{16}

Regarding the simple multimodal problems f_4–f_{16} when $D = 30$, Table 4 and Figure 6 show that the proposed RDIM-PSO outperforms other compared algorithms on f_4, f_{14} and f_{16}. In addition, RDIM-PSO performs similarly to CPSO-H and CSO on f_5 and f_7, respectively. CPSO-H shows the best performances on f_8 and f_{10}. SLPSO obtains the best result on f_{12}. CSO takes first place on f_6, f_9, f_{11}, f_{13} and f_{15}. RDIM-PSO wins PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 9, 9, 13, 12, 12, 5 and 13 test problems, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CSO, PSO-w, CPSO-H, CLPSO (TLBO), SLPSO and Jaya in a descending direction. The average rank of CLPSO is the same as that of TLBO. Table 5 shows that RDIM-PSO wins PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 10, 7, 12, 9, 12, 6 and 13 test problems under $D = 50$, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CSO, CPSO-H, CLPSO, PSO-w, SLPSO, TLBO and Jaya in a descending direction. Table 6 shows that RDIM-PSO wins PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 10, 7, 6, 8, 11, 6 and 13 test problems under $D = 100$, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CPSO-H, CLPSO, CSO, PSO-w (SLPSO), TLBO and Jaya in a descending direction. The average rank of PSO-w is the same as that of SLPSO. The results demonstrate that employing the non-dominated sorting procedure based on fitness and diversity can enhance exploration and mitigate the premature convergence problem.

(3) *Hybrid problems* f_{17}–f_{22}

With regard to the hybrid problems f_{17}–f_{22} when $D = 30$, the results in Table 7 and Figure 7 show that the proposed RDIM-PSO outperforms other algorithms on f_{17} and f_{21}. CSO gets the best results on f_{18}, f_{19} and f_{22}. PSO-w ranks first on f_{20}. RDIM-PSO outperforms PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 4, 6, 5, 6, 5, 3 and 6 test problems, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CSO, PSO-w, TLBO, CLPSO, CPSO-H (SLPSO) and Jaya in a descending direction. The average rank of CPSO-H is the same as that of SLPSO. Table 8 shows that RDIM-PSO outperforms PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 4, 5, 4, 5, 4, 3 and 6 test problems under $D = 50$, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CSO, TLBO, PSO-w, CLPSO, CPSO-H, SLPSO and Jaya in a descending direction. Table 9 shows that RDIM-PSO outperforms PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 5, 5, 3, 5, 3, 3 and 6

Table 3 Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_1–f_3 with $D = 100$.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
	f_1	−/5	−/4	−/6	−/7	−/3	−/2	−/8	√/1
Compare/rank	f_2	≈/1	≈/1	≈/1	−/7	≈/1	≈/1	−/8	√/1
	f_3	−/2	−/6	−/4	−/7	−/3	−/5	−/8	√/1
−/≈/+		2/1/0	2/1/0	2/1/0	3/0/0	2/1/0	2/1/0	3/0/0	\
Avg-rank		2.67	3.67	3.67	7.00	2.33	2.67	8.00	1.00

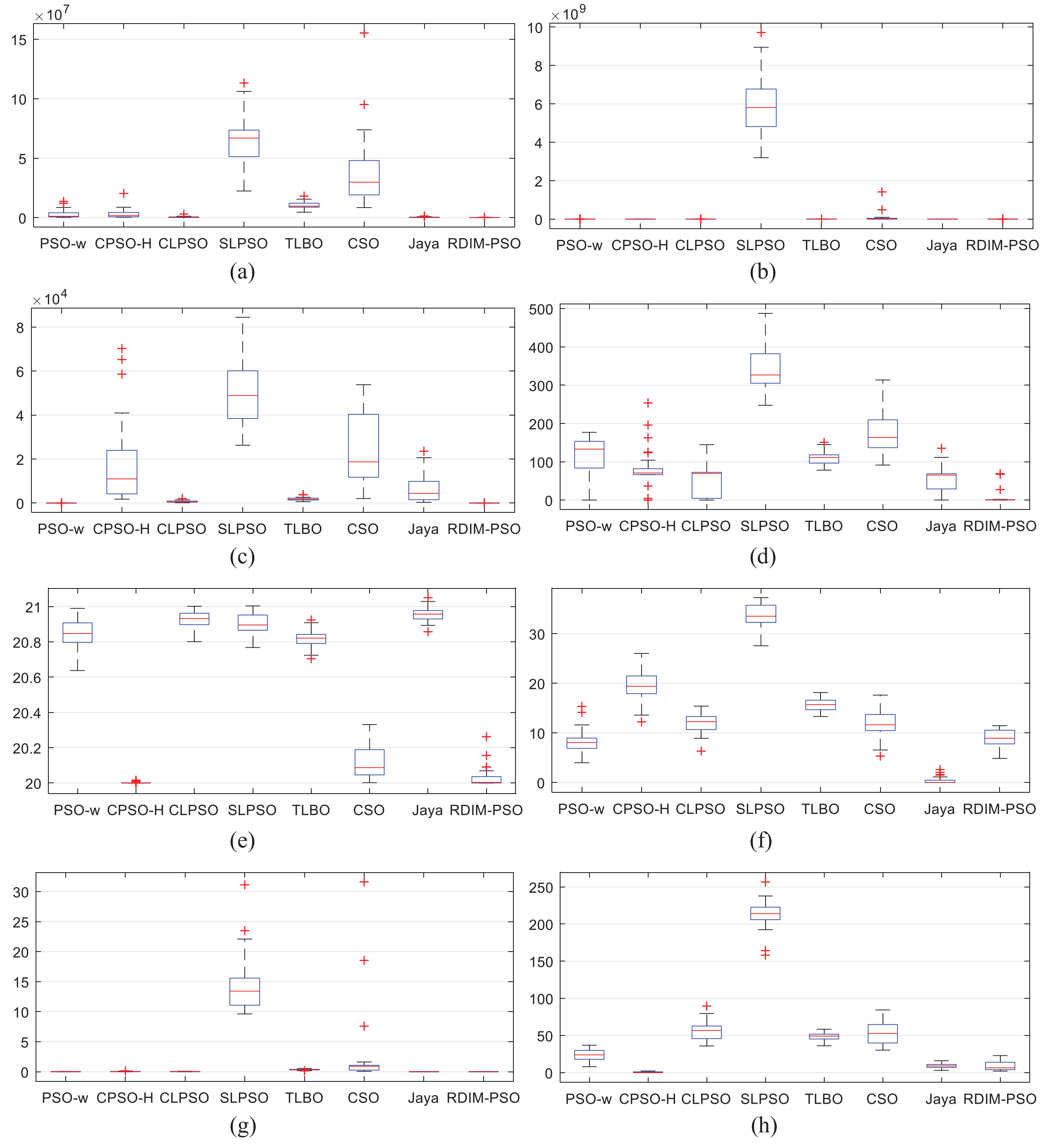
Table 4 Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_4–f_{16} with $D = 30$.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
	f_4	−/6	−/4	−/5	−/7	−/3	−/2	−/8	√/1
	f_5	−/5	+/1	−/4	−/3	−/7	−/8	−/6	√/2
	f_6	≈/2	−/7	−/6	−/4	−/5	+/1	−/8	√/2
	f_7	−/3	−/5	−/6	−/7	−/4	≈/1	−/8	√/1
	f_8	−/4	+/1	−/5	−/6	−/7	≈/2	−/8	√/2
	f_9	≈/2	−/6	−/7	−/5	≈/2	+/1	−/8	√/2
Compare/rank	f_{10}	≈/3	+/1	−/5	−/7	−/6	+/2	−/8	√/3
	f_{11}	−/3	−/4	−/6	−/5	−/7	+/1	−/8	√/2
	f_{12}	−/5	+/2	−/4	+/1	−/8	−/6	−/7	√/3
	f_{13}	≈/2	−/6	−/4	−/7	−/5	+/1	−/8	√/2
	f_{14}	−/4	−/5	−/3	−/7	−/2	−/6	−/8	√/1
	f_{15}	−/3	−/5	−/6	−/8	−/4	+/1	−/7	√/2
	f_{16}	−/4	−/3	−/5	−/7	−/6	−/2	−/8	√/1
−/≈/+		9/4/0	9/0/4	13/0/0	12/0/1	12/1/0	5/2/6	13/0/0	\
Avg-rank		3.54	3.85	5.08	5.69	5.08	2.62	7.69	1.85

test problems under D = 100, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CSO, TLBO, CLPSO, PSO-w, CPSO-H, SLPSO and Jaya in a descending direction.

(4) *Composite problems* f_{23}–f_{30}

Concerning the hybrid problems f_{23}–f_{30} when D = 30, Table 10 and Figure 7 present that PSO-w, CLPSO, TLBO, CSO and RDIM-PSO achieve similar results on f_{23}. CLPSO, TLBO, Jaya and RDIM-PSO achieve similar results on f_{26}. TLBO ranks first on f_{24} and f_{25}. CSO is the winner on f_{27} and f_{28}. RDIM-PSO surpasses all other algorithms on f_{29} and f_{30}. RDIM-PSO performs better than PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 8, 8, 5, 8, 5, 3 and 8 test problems, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CSO, CLPSO, TLBO, PSO-w, SLPSO and CPSO-H (Jaya) in a descending direction. The average rank of CPSO-H is the same as that of Jaya. Table 11 shows that RDIM-PSO performs better than PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 4, 5, 2, 6, 4, 2 and 7 test problems under D = 50, respectively. The overall Wilcoxon ranking sequences for the test problems are CLPSO (CSO), TLBO (RDIM-PSO), PSO-w (CPSO-H), SLPSO and Jaya in a descending direction. The average rank of CLPSO is the same as that of CSO. The average rank of TLBO is the same as that of RDIM-PSO. The average rank of PSO-w is the same as that of CPSO-H. Table 12 shows that RDIM-PSO performs better than PSO-w,



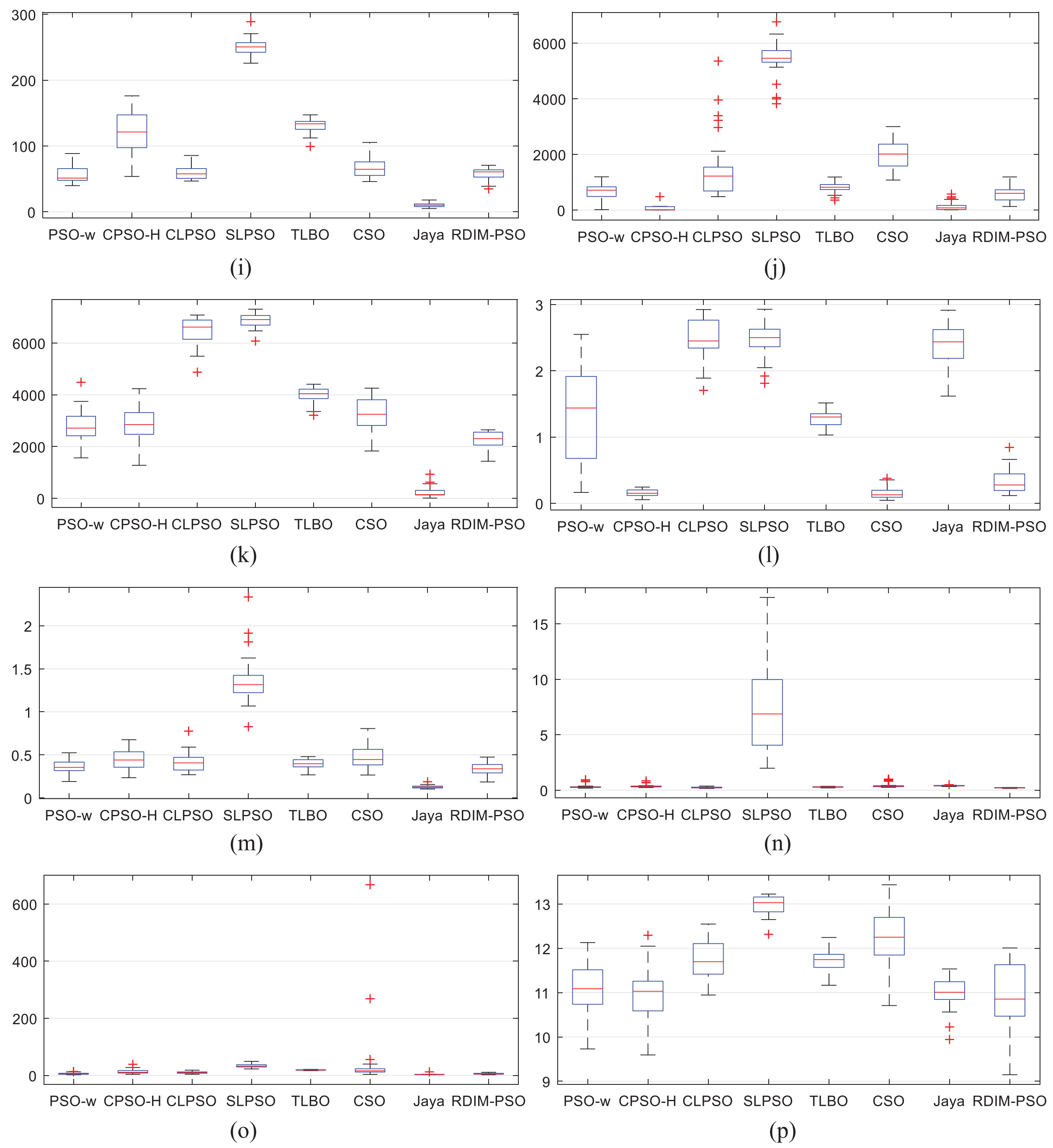


Figure 6 Box-plot of function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on 16 test problems with $D = 30$. (a) f_1; (b) f_2; (c) f_3; (d) f_4; (e) f_5; (f) f_6; (g) f_7; (h) f_8; (i) f_9; (j) f_{10}; (k) f_{11}; (l) f_{12}; (m) f_{13}; (n) f_{14}; (o) f_{15}; (p) f_{16}.

CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 6, 5, 4, 6, 3, 5 and 7 test problems under $D = 100$, respectively. The overall Wilcoxon ranking sequences for the test problems are CSO, CLPSO, RDIM-PSO, TLBO, CPSO-H, PSO-w, SLPSO and Jaya in a descending direction.

Overall, compared with these seven popular optimization algorithms when $D = 30$, the results based on Wilcoxon's rank-sum test in Table 13 shows the proposed RDIM-PSO is the winner in most cases. Compared with PSO-w, TLBO, SLPSO and Jaya, RDIM-PSO wins on 23, 25, 29 and 30 problems, respectively. Compared with CPSO-H and CLPSO, RDIM-PSO wins on 26 problems. Compared with CSO, RDIM-PSO wins on 14 problems and loses on 11 problems. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CSO, PSO-w, TLBO, CLPSO, CPSO-H, SLPSO and Jaya in a descending direction. RDIM-PSO performs better than PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 21, 20, 21, 23, 23, 14 and 29 test problems under $D = 50$, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CSO, CLPSO, PSO-w (TLBO), CPSO-H, SLPSO and Jaya in a descending direction. The average rank of PSO-w is the same as that of TLBO. RDIM-PSO performs better than PSO-w,

Table 5 Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_4–f_{16} with $D = 50$.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
	f_4	−/6	−/2	−/4	−/7	−/5	−/3	−/8	√/1
	f_5	−/5	−/2	−/4	−/3	−/7	−/8	−/6	√/1
	f_6	+/2	−/7	−/5	+/3	−/6	+/1	−/8	√/4
	f_7	−/3	−/4	−/5	−/7	−/6	+/1	−/8	√/3
	f_8	−/5	+/1	−/4	−/6	−/7	+/2	−/8	√/3
	f_9	+/2	−/7	−/6	+/3	+/4	+/1	−/8	√/5
Compare/rank	f_{10}	−/5	+/1	+/3	−/7	−/6	+/2	−/8	√/4
	f_{11}	≈/2	≈/2	−/6	≈/2	−/7	+/1	−/8	√/2
	f_{12}	−/5	+/2	−/4	+/1	−/6	−/7	−/8	√/3
	f_{13}	−/5	≈/2	−/4	−/7	−/6	+/1	−/8	√/2
	f_{14}	−/7	−/5	−/2	−/6	−/4	−/3	−/8	√/1
	f_{15}	−/3	−/6	−/4	−/7	−/5	−/2	−/8	√/1
	f_{16}	−/5	≈/1	−/3	−/7	−/6	−/4	−/8	√/1
−/≈/+		10/1/2	7/3/3	12/0/1	9/1/3	12/0/1	6/0/7	13/0/0	\
Avg-rank		4.23	3.23	4.15	5.08	5.77	2.77	7.85	2.31

Table 6 Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_4–f_{16} with $D = 100$.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
	f_4	−/6	−/4	−/3	−/7	−/5	−/2	−/8	√/1
	f_5	−/5	−/2	−/4	−/3	−/6	−/7	−/8	√/1
	f_6	−/3	−/6	−/4	−/2	−/7	−/1	−/8	√/5
	f_7	−/4	−/5	−/3	−/7	−/6	≈/1	−/8	√/1
	f_8	−/5	+/1	+/2	−/6	−/7	+/3	−/8	√/4
	f_9	+/2	−/6	≈/3	+/1	≈/3	−/7	−/8	√/3
Compare/rank	f_{10}	−/5	+/2	+/1	−/6	−/7	+/3	−/8	√/4
	f_{11}	−/5	−/4	−/6	+/2	−/7	+/1	−/8	√/3
	f_{12}	−/5	+/2	−/4	+/1	−/7	−/8	−/6	√/3
	f_{13}	−/7	+/1	+/3	+/4	≈/5	+/2	−/8	√/5
	f_{14}	−/4	−/2	−/6	−/7	−/3	−/5	−/8	√/1
	f_{15}	+/1	≈/2	≈/2	−/7	−/6	≈/2	−/8	√/2
	f_{16}	−/5	+/1	≈/2	−/4	−/7	−/6	−/8	√/2
−/≈/+		10/0/3	7/1/5	6/3/4	8/0/5	11/2/0	6/2/5	13/0/0	\
Avg-rank		4.38	2.92	3.31	4.38	5.85	3.69	7.85	2.69

Table 7 Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_{17}–f_{22} with $D = 30$.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
	f_{17}	−/4	−/6	−/5	−/7	−/3	−/2	−/8	√/1
	f_{18}	−/6	−/7	−/3	−/5	−/4	+/1	−/8	√/2
Compare/rank	f_{19}	−/3	−/6	−/4	−/7	−/5	+/1	−/8	√/2
	f_{20}	≈/1	−/7	−/4	−/8	−/3	−/6	−/5	√/1
	f_{21}	−/2	−/6	−/4	−/7	−/3	−/5	−/8	√/1
	f_{22}	+/4	−/8	+/2	−/6	+/3	+/1	−/7	√/5
−/≈/+		4/1/1	6/0/0	5/0/1	6/0/0	5/0/1	3/0/3	6/0/0	\
Avg-rank		3.33	6.67	3.67	6.67	3.50	2.67	7.33	2.00

CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on 23, 19, 15, 22, 19, 16 and 29 test problems under $D = 100$, respectively. The overall Wilcoxon ranking sequences for the test problems are RDIM-PSO, CSO, CLPSO, CPSO-H, TLBO, PSO-w, SLPSO and Jaya in a descending direction.

Figures 6–11 show the Box-plot of function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on 30 test problems with $D = 30$, $D = 50$ and $D = 100$, respectively. Generally, the global search and local search abilities always contradict each other. The performance of optimization algorithm relies heavily on the good balance between exploration and exploitation. In the early stage, more exploration is beneficial, while more exploitation is beneficial in the later stage. The proposed RDIM-PSO considers both fitness and diversity information during the optimization process, where the fitness is related to exploitation and the diversity is associated with exploration. In addition, the particles are attracted to different regions (neighborhoods) by the reference directions, which allow more promising neighborhoods to be more intensively searched. Finally, the promising particles are generated by the Gaussian process-based inverse model, which can further improve the performance of RDIM-PSO.

In order to test the statistical significance of the eight compared algorithms, the Wilcoxon's test at the 5% significance level, which is implemented by using KEEL software [49], is employed based on

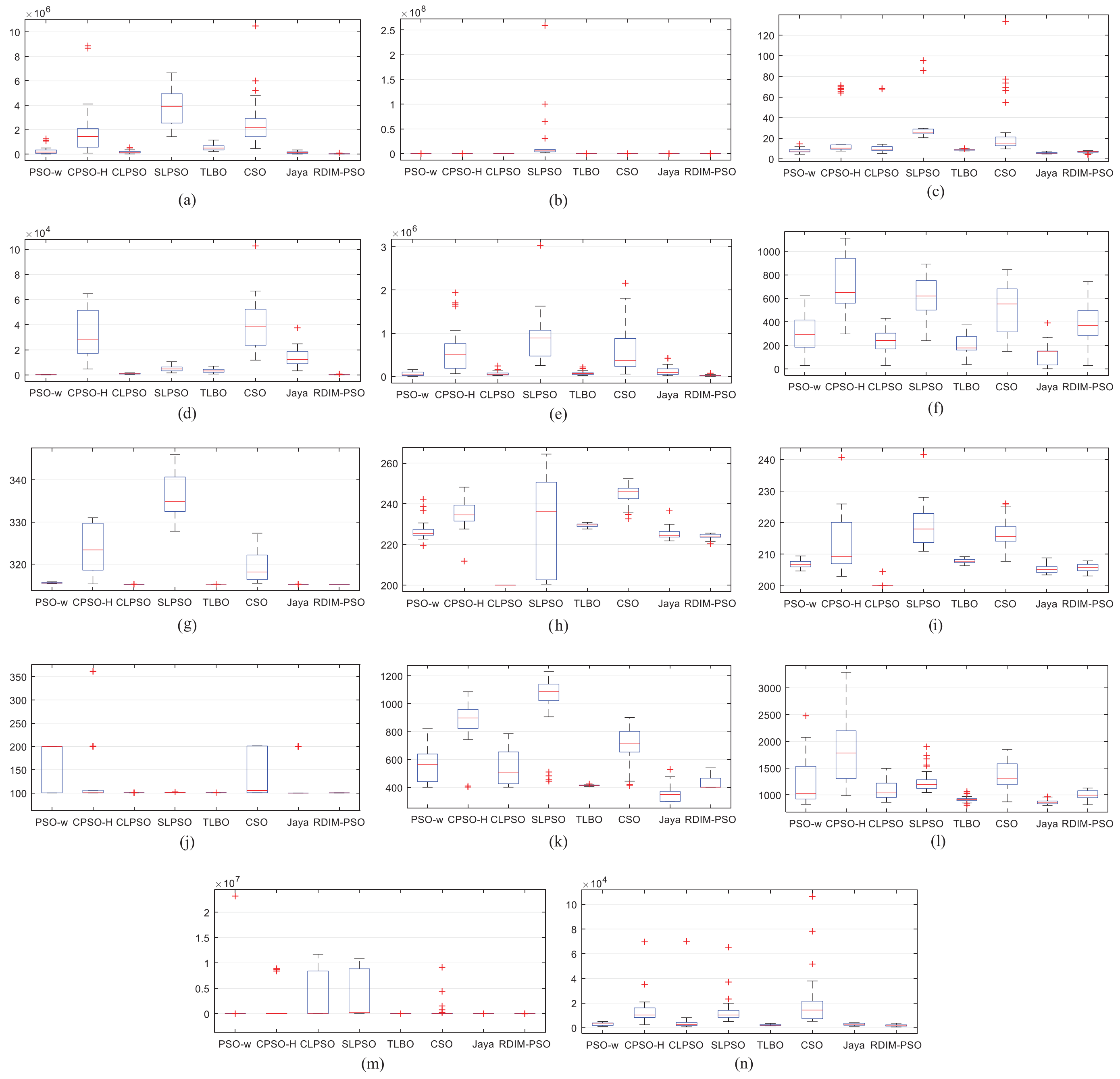


Figure 7 Box-plot of function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on 14 test problems with $D = 30$. (a) f_{17}; (b) f_{18}; (c) f_{19}; (d) f_{20}; (e) f_{21}; (f) f_{22}; (g) f_{23}; (h) f_{24}; (i) f_{25}; (j) f_{26}; (k) f_{27}; (l) f_{28}; (m) f_{29}; (n) f_{30}.

Table 8 Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_{17}–f_{22} with $D = 50$.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
Compare/rank	f_{17}	−/4	−/5	−/6	−/7	−/3	−/2	−/8	\/1
	f_{18}	≈/1	≈/1	≈/1	−/7	≈/1	≈/1	−/8	\/1
	f_{19}	−/6	−/5	−/3	−/7	−/4	+/1	−/8	\/2
	f_{20}	−/2	−/8	−/4	−/7	−/3	−/5	−/6	\/1
	f_{21}	−/4	−/5	−/6	−/7	−/3	−/2	−/8	\/1
	f_{22}	+/4	−/7	+/2	≈/5	+/3	+/1	−/8	\/5
−/≈/+		4/1/1	5/1/0	4/1/1	5/1/0	4/1/1	3/1/2	6/0/0	\
Avg-rank		3.50	5.17	3.67	6.67	2.83	2.00	7.67	1.83

Table 9 | Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_{17}–f_{22} with $D = 100$.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
Compare/rank	f_{17}	−/5	−/4	−/6	−/7	−/3	−/2	−/8	√/1
	f_{18}	−/5	−/6	+/1	−/7	≈/3	+/2	−/8	√/3
	f_{19}	−/6	≈/1	≈/1	−/7	≈/1	≈/1	−/8	√/1
	f_{20}	−/2	−/8	−/5	−/7	−/3	−/4	−/6	√/1
	f_{21}	−/5	−/4	−/6	−/7	−/3	−/2	−/8	√/1
	f_{22}	≈/4	+/7	+/2	≈/4	+/3	+/1	+/8	√/4
−/≈/+		5/1/0	5/1/0	3/1/2	5/1/0	3/2/1	3/1/2	6/0/0	\
Avg-rank		4.50	5.00	3.50	6.50	2.67	2.00	7.67	1.83

Table 10 | Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_{23}–f_{30} with $D = 30$.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
Compare/rank	f_{23}	−/5	−/7	−/4	−/6	≈/1	≈/1	−/8	√/1
	f_{24}	−/4	−/7	−/5	−/8	+/1	≈/2	−/6	√/2
	f_{25}	−/4	−/6	−/5	−/7	+/1	≈/2	−/8	√/2
	f_{26}	−/8	−/6	−/3	−/7	−/2	−/5	−/4	√/1
	f_{27}	−/5	−/7	+/2	−/6	−/4	+/1	−/8	√/3
	f_{28}	−/5	−/8	+/2	−/7	−/4	+/1	−/6	√/3
	f_{29}	−/5	−/6	−/2	−/4	−/7	−/3	−/8	√/1
	f_{30}	−/4	−/7	≈/1	−/8	−/5	−/3	−/6	√/1
−/≈/+		8/0/0	8/0/0	5/1/2	8/0/0	5/1/2	3/3/2	8/0/0	\
Avg-rank		5.00	6.75	3.00	6.63	3.13	2.25	6.75	1.75

Table 11 | Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_{23}–f_{30} with $D = 50$.

	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
Compare/rank	f_{23}	−/5	−/6	−/4	−/7	−/3	≈/1	−/8	√/1
	f_{24}	≈/3	≈/3	+/2	−/7	+/1	≈/3	−/8	√/3
	f_{25}	+/5	+/3	+/4	−/7	+/1	+/2	−/8	√/6
	f_{26}	−/7	−/4	≈/1	−/8	≈/1	−/5	−/6	√/1
	f_{27}	+/3	−/7	+/2	+/4	−/6	+/1	−/8	√/5
	f_{28}	+/5	≈/7	+/2	+/6	+/3	+/1	+/4	√/7
	f_{29}	−/6	−/5	−/2	−/4	−/7	−/3	−/8	√/1
	f_{30}	−/6	−/5	+/1	−/7	−/4	≈/2	−/8	√/2
−/≈/+		4/1/3	5/2/1	2/1/5	6/0/2	4/1/3	2/3/3	7/0/1	\
Avg-rank		5.00	5.00	2.25	6.25	3.25	2.25	7.25	3.25

Table 12 | Experimental results of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on f_{23}–f_{30} with $D = 100$.

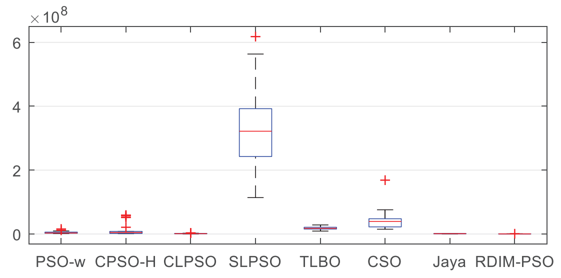
	F	PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
Compare/rank	f_{23}	−/5	−/6	−/4	−/7	+/1	−/3	−/8	√/2
	f_{24}	−/5	−/3	−/4	−/7	+/1	−/6	−/8	√/2
	f_{25}	−/6	≈/4	+/3	−/7	+/1	+/2	−/8	√/4
	f_{26}	−/6	+/1	+/2	−/7	+/3	−/5	−/8	√/4
	f_{27}	+/4	−/6	+/3	+/2	−/7	+/1	−/8	√/5
	f_{28}	+/4	+/6	+/2	+/3	+/5	+/1	+/7	√/8
	f_{29}	−/4	−/6	−/3	−/5	−/7	−/2	−/8	√/1
	f_{30}	−/5	−/6	−/4	−/7	−/3	−/2	−/8	√/1
−/≈/+		6/0/2	5/1/2	4/0/4	6/0/2	3/0/5	5/0/3	7/0/1	\
Avg-rank		4.88	4.75	3.13	5.63	3.50	2.75	7.88	3.38

the PR values. Tables 14–19 summarizes the statistical test results. It can be seen from Tables 14, 16, and 18 that RDIM-PSO provides higher R+ values than R− values compared with PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya. Furthermore, the p values of PSO-w, CPSO-H, SLPSO, TLBO and Jaya are less than 0.05, which means that RDIM-PSO is significantly better than these competitors. The p values of CSO is greater than 0.05, which means that the performance of RDIM-PSO is not different from that of CSO. Similarly, the p values of CLPSO is greater than 0.05 when $D = 50$ and $D = 100$, which means that the performance of CLPSO is not different from that of CSO. To further determine the ranking of the eight compared algorithms, the Friedman's test, which is also implemented by using KEEL software, is conducted. As shown in Table 15, the overall ranking sequences for the test problems are RDIM-PSO, CSO, PSO-w, TLBO, CLPSO, CPSO-H, SLPSO and Jaya. Table 17 shows the overall ranking sequences for the test problems are RDIM-PSO, CSO, CLPSO, TLBO, PSO-w, CPSO-H, SLPSO and Jaya. Table 19 shows the overall ranking sequences for the test problems are RDIM-PSO, CSO, CLPSO, CPSO-H, TLBO, PSO-w, SLPSO and Jaya. Therefore, it can be concluded that the improvement strategies are effective.

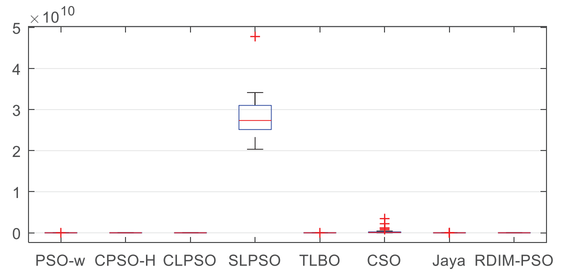
To illustrate the convergence speed of the proposed RDIM-PSO, the convergence curves on 30-dimensional, 50-dimensional and 100-dimensional CEC2014 benchmark problems are presented in

Table 13 Comparison of RDIM-PSO with PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO and Jaya on the CEC2014 benchmark problems ($D = 30$, $D = 50$ and $D = 100$ dimensions).

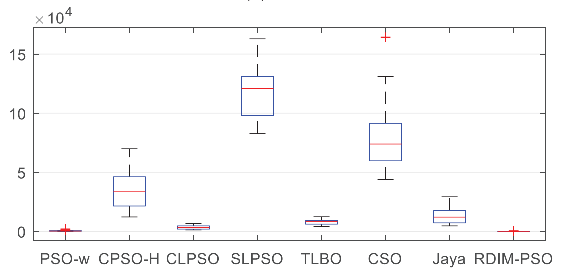
D		PSO-w	CPSO-H	CLPSO	SLPSO	TLBO	CSO	Jaya	RDIM-PSO
30	−/≈/+	23/6/1	26/0/4	26/1/3	29/0/1	25/2/3	14/5/11	30/0/0	√
	Avg-rank	3.77	5.33	4.27	6.27	4.03	2.63	7.40	1.77
50	−/≈/+	21/3/6	20/6/4	21/2/7	23/2/5	23/2/5	14/4/12	29/0/1	√
	Avg-rank	4.23	4.27	3.67	5.90	4.23	2.50	7.67	2.33
100	−/≈/+	23/2/5	19/4/7	15/5/10	22/1/7	19/5/6	16/4/10	29/0/1	√
	Avg-rank	4.37	3.90	3.33	5.40	4.23	3.00	7.83	2.53



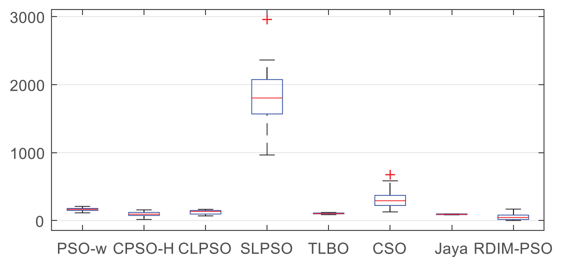
(a)



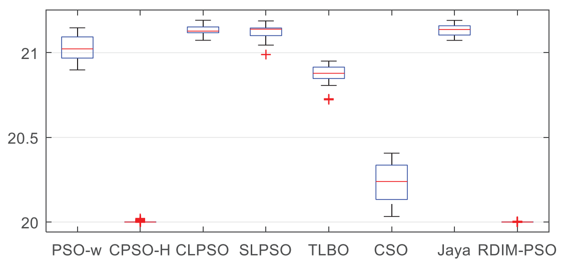
(b)



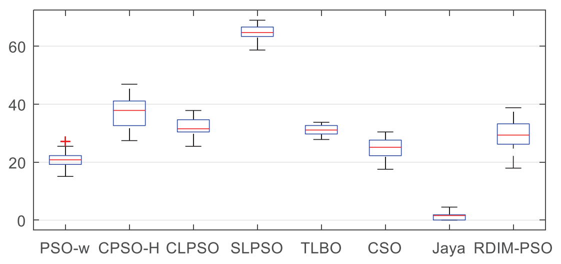
(c)



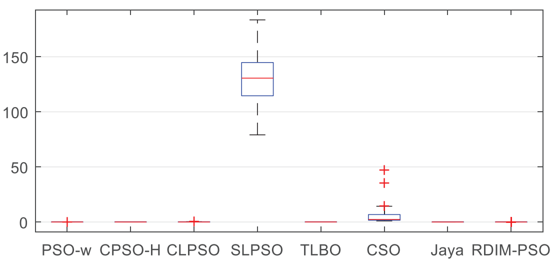
(d)



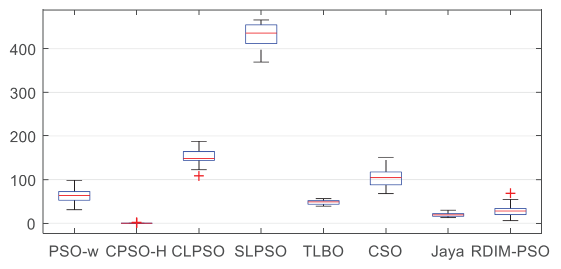
(e)



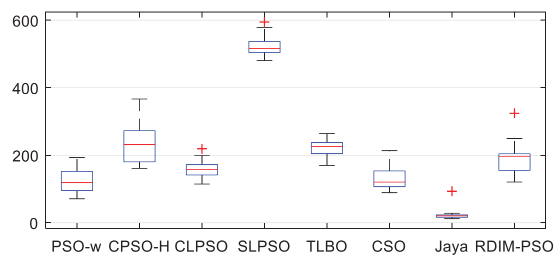
(f)



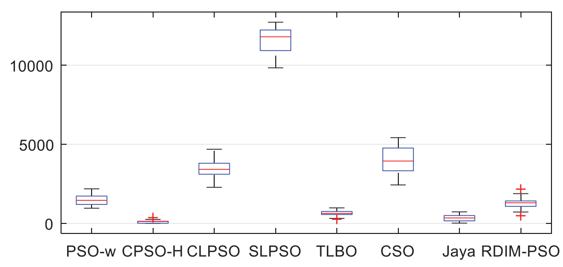
(g)



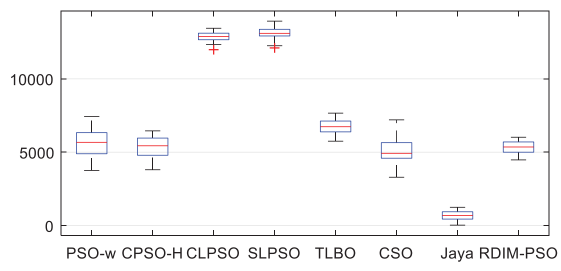
(h)



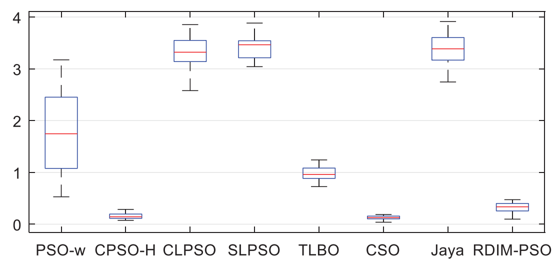
(i)



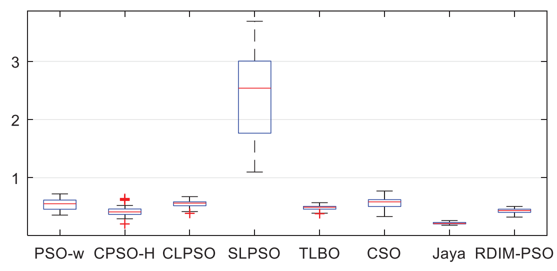
(j)



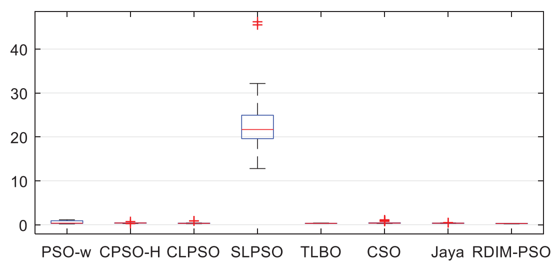
(k)



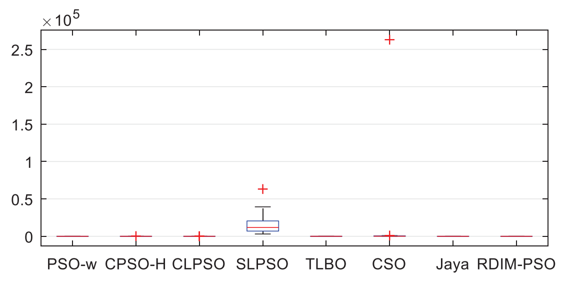
(l)



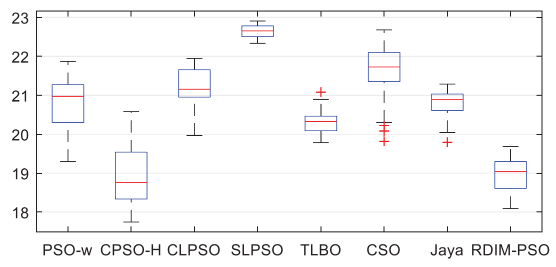
(m)



(n)

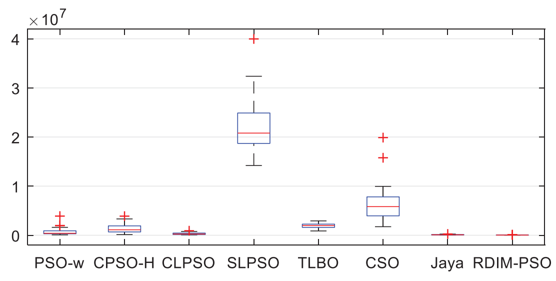


(o)

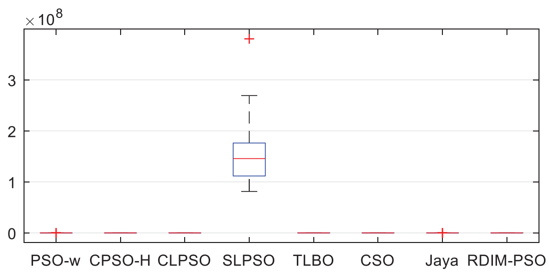


(p)

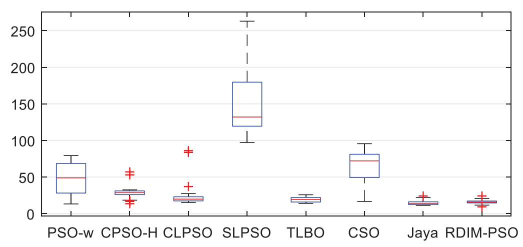
Figure 8 Box-plot of function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on 16 test problems with $D = 50$. (a) f_1; (b) f_2; (c) f_3; (d) f_4; (e) f_5; (f) f_6; (g) f_7; (h) f_8; (i) f_9; (j) f_{10}; (k) f_{11}; (l) f_{12}; (m) f_{13}; (n) f_{14}; (o) f_{15}; (p) f_{16}.



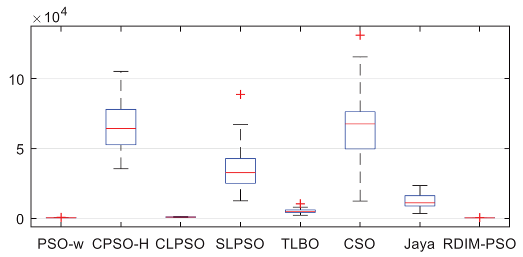
(a)



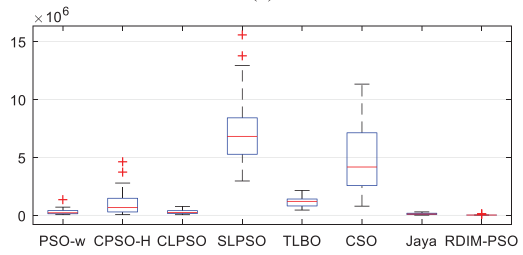
(b)



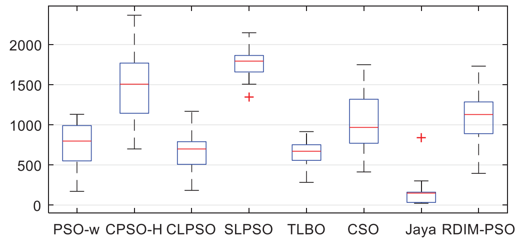
(c)



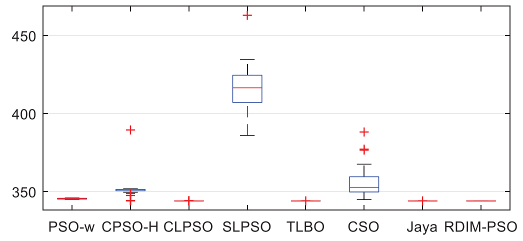
(d)



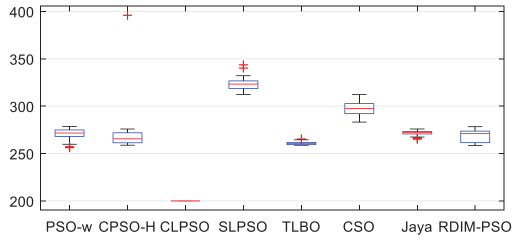
(e)



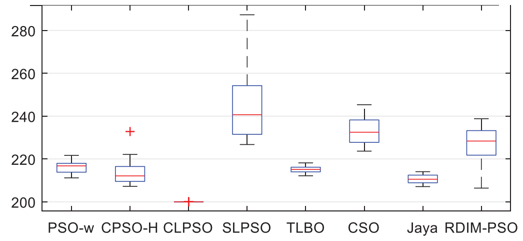
(f)



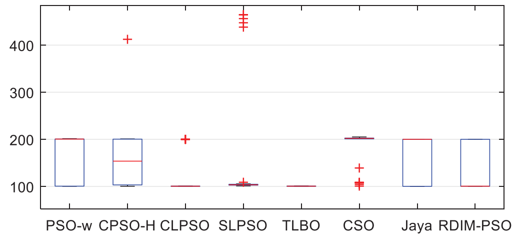
(g)



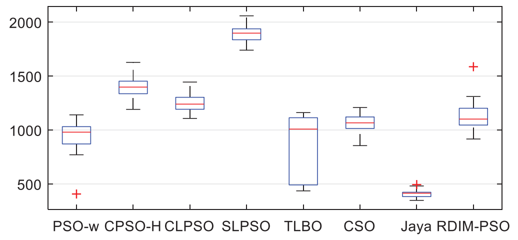
(h)



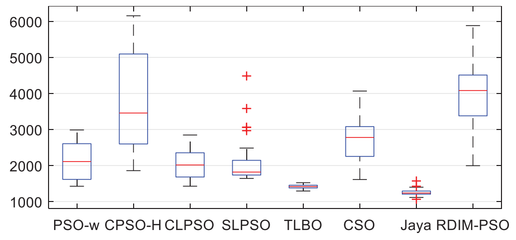
(i)



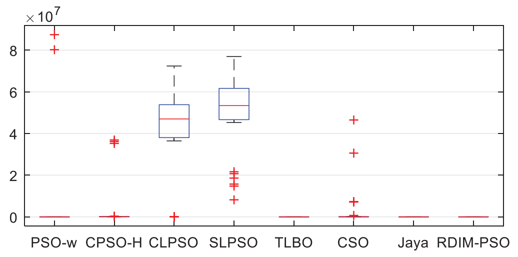
(j)



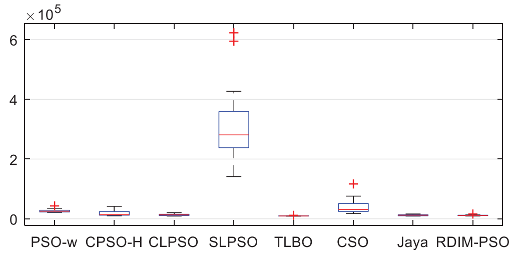
(k)



(l)



(m)



(n)

Figure 9 Box-plot of function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on 14 test problems with $D = 50$. (a) f_{17}; (b) f_{18}; (c) f_{19}; (d) f_{20}; (e) f_{21}; (f) f_{22}; (g) f_{23}; (h) f_{24}; (i) f_{25}; (j) f_{26}; (k) f_{27}; (l) f_{28}; (m) f_{29}; (n) f_{30}.

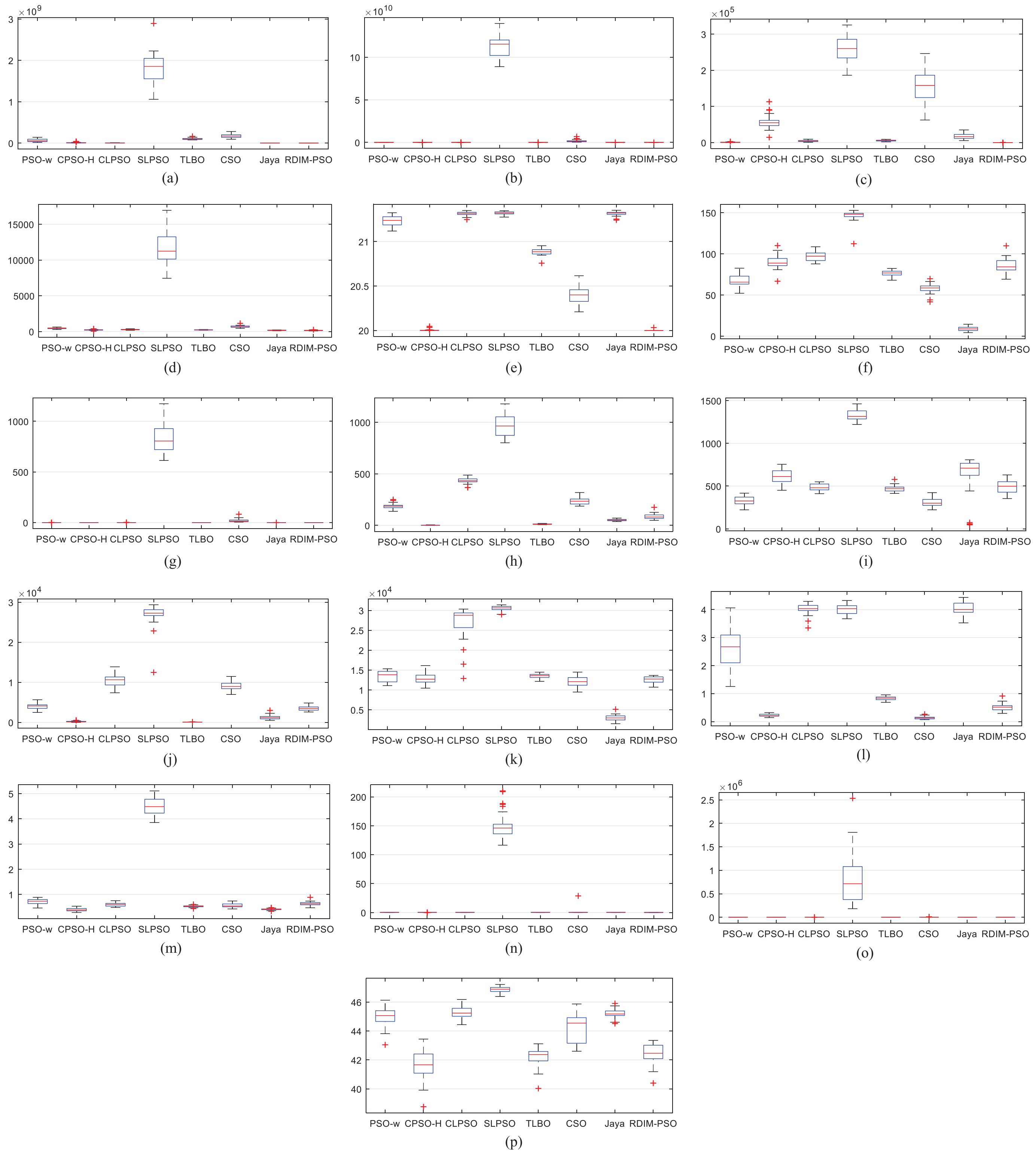


Figure 10 Box-plot of function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on 16 test problems with $D = 100$. (a) f_1; (b) f_2; (c) f_3; (d) f_4; (e) f_5; (f) f_6; (g) f_7; (h) f_8; (i) f_9; (j) f_{10}; (k) f_{11}; (l) f_{12}; (m) f_{13}; (n) f_{14}; (o) f_{15}; (p) f_{16}.

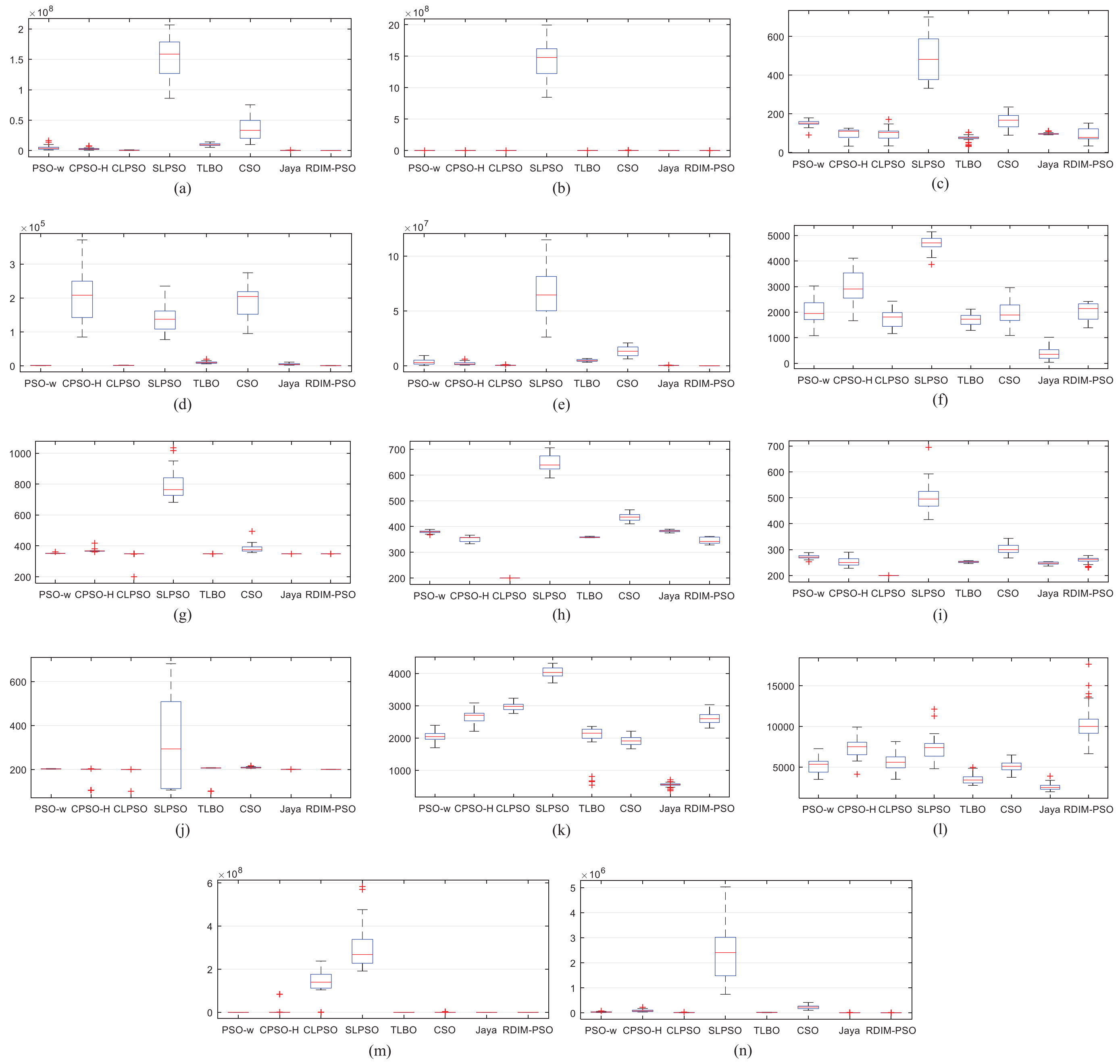


Figure 11 Box-plot of function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on 14 test problems with $D = 100$. (a) f_{17}; (b) f_{18}; (c) f_{19}; (d) f_{20}; (e) f_{21}; (f) f_{22}; (g) f_{23}; (h) f_{24}; (i) f_{25}; (j) f_{26}; (k) f_{27}; (l) f_{28}; (m) f_{29}; (n) f_{30}.

Table 14 Results obtained by the Wilcoxon test for algorithm RDIM-PSO with $D = 30$.

VS	R^+	R^-	Exact p-Value	Asymptotic p-Value
PSO-w	390.0	45.0	≥0.2	0.000183
CPSO-H	404.0	31.0	≥0.2	0.00005
CLPSO	411.5	53.5	≥0.2	0.000222
SLPSO	462.0	3.0	≥0.2	0.000002
TLBO	417.5	47.5	≥0.2	0.000131
CSO	278.0	187.0	≥0.2	0.344075
Jaya	435.0	0.0	≥0.2	0.000002

Table 15 Average ranking of the algorithms (Friedman) with $D = 30$.

Algorithm	Ranking
RDIM-PSO	2.0167
PSO-w	3.7667
CPSO-H	5.3667
CLPSO	4.2833
SLPSO	6.2667
TLBO	4.15
CSO	2.75
Jaya	7.4

Table 16 Results obtained by the Wilcoxon test for algorithm RDIM-PSO with $D = 50$.

VS	R$^+$	R$^-$	Exact p-Value	Asymptotic p-Value
PSO-w	361.0	104.0	≥0.2	0.00781
CPSO-H	392.5	72.5	≥0.2	0.000963
CLPSO	284.0	151.0	≥0.2	0.147406
SLPSO	377.0	88.0	≥0.2	0.00286
TLBO	352.0	83.0	≥0.2	0.00351
CSO	264.0	171.0	≥0.2	0.309491
Jaya	447.0	18.0	≥0.2	0.00001

Table 17 Average ranking of the algorithms (Friedman) with $D = 50$.

Algorithm	Ranking
RDIM-PSO	2.65
PSO-w	4.4833
CPSO-H	4.5667
CLPSO	3.6167
SLPSO	5.9
TLBO	4.3833
CSO	2.7
Jaya	7.7

Table 18 Results obtained by the Wilcoxon test for algorithm RDIM-PSO with $D = 100$.

VS	R$^+$	R$^-$	Exact p-Value	Asymptotic p-Value
PSO-w	366.5	98.5	≥0.2	0.005544
CPSO-H	332.0	103.0	≥0.2	0.012895
CLPSO	238.0	197.0	≥0.2	0.649766
SLPSO	371.0	94.0	≥0.2	0.00425
TLBO	333.5	131.5	≥0.2	0.036343
CSO	238.5	196.5	≥0.2	0.642004
Jaya	448.0	17.0	≥0.2	0.000009

Table 19 Average ranking of the algorithms (Friedman) with $D = 100$.

Algorithm	Ranking
RDIM-PSO	3.0833
PSO-w	4.5667
CPSO-H	4.1833
CLPSO	3.3833
SLPSO	5.4
TLBO	4.4167
CSO	3.1667
Jaya	7.8

Figures 12–14, respectively. There are 16 representative problems including 2 unimodal problems, 6 multimodal problems, 4 hybrid problems and 4 composite problems. The curves illustrate mean errors (in logarithmic scale) in 30 independent simulations of PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO. As mentioned above, the curves indicate that, in most problems, RDIM-PSO either achieves a fast convergence or behaves similarly to it.

5. THE REAL-WORLD OPTIMIZATION PROBLEM

To test the performance of the proposed RDIM-PSO algorithm on solving the real-world problem, time series prediction with the ANN is chosen as the testing objective. The ANN is fit for solving the problems of chaotic time series prediction and functions approximation. It is an effective way that verify the effectiveness of algorithm by training the parameters of ANN. Various heuristic optimization algorithms have been used to do this, such as PSO [50], DE [51], GAs [52], Ant colony algorithm [53], hybrid particle swarm optimization and gravitational search algorithm (PSOGSA) [54], etc. To deeply verify the effectiveness of the RDIM-PSO, a three-layer feed-forward network which includes input units, hidden units and output units is selected for chaotic time series prediction and function approximation. The three-layer ANN is depicted in Figure 15.

In Figure 15, $\mathrm{X} = (\mathrm{x}_1, \mathrm{x}_2, \cdots, \mathrm{x}_\mathrm{i}, \cdots \mathrm{x}_\mathrm{n})^\mathrm{T}$ is input layer, $\mathrm{O} = (\mathrm{o}_1, \mathrm{o}_2, \cdots, \mathrm{o}_\mathrm{k}, \cdots \mathrm{o}_\mathrm{t})^\mathrm{T}$ is output layer. W_{ij} and W_{jk} is the weight between the I–H (input layer and hidden layer) and H–O (hidden layer and output layer), respectively. $i = 1, 2, \cdots, n$. $j = 1, 2, \cdots, p$, $k = 1, 2, \cdots, t$. The aim of neural network training is to find a set of weights with the smallest error measure. Therefore, the mean sum of squared errors (MSE) is selected as an evaluation indicator of optimization algorithms, shown as follows:

$$\mathrm{MSE} = \frac{1}{QK} \sum_{i=1}^{Q} \sum_{j=1}^{K} \frac{1}{2} \left(d_{ij} - o_{ij}\right)^2 \tag{17}$$

where Q and K is the number of training data set and output units, respectively. d_{ij} and o_{ij} is desired output and output inferred from neural network, respectively. The fitness of the ith training sample is used as evaluation indicator and is calculated by:

$$\text{Fitness}\left(X_i\right) = \mathrm{MSE}\left(X_i\right) \tag{18}$$

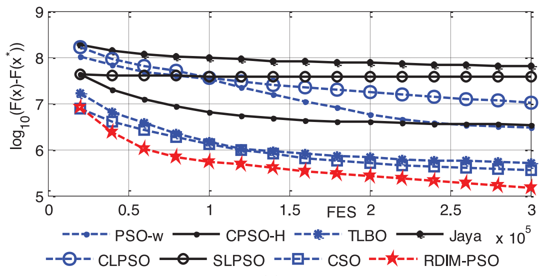
5.1. SISO Nonlinear Function Approximation

In this example, there are one input units, five hidden units and one output units in the three-layer feed-forward ANN. The SISO nonlinear system used in this experiment is given as follows [55]:

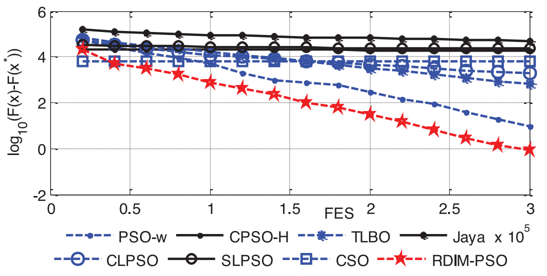
$$\mathrm{y} = \sin(2x)\, e^{-2x} \tag{19}$$

In this experiment, the number (dimension) of the variables is 16. Moreover, 200 pairs of data are chosen from Eq. (19) as the training samples. Eight optimization algorithms are used to optimize the neural network. NP (population size) = 50; FES (maximal number of function evaluations) = 10,000. The other parameters of the different algorithms are the same as those used in CEC2014. Experiments introduce 30db additive with Gaussian noise to assess the performance of each algorithm in noise. For each algorithm, 30 independent runs are performed. Here, "Mean" (average best value of the MSE) and "Std" (standard deviation of MSE) are used as evaluation indicators.

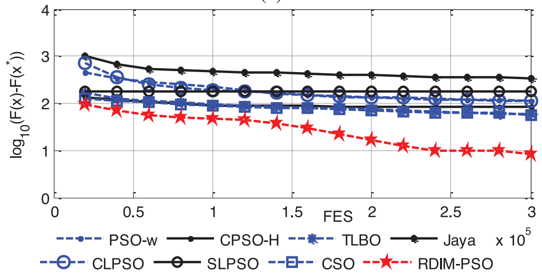
Table 20 shows that mean MSE of RDIM-PSO is the smallest, and the Testing Error Std of RDIM-PSO is the smallest of the eight algorithms. In the case of Training Error Std, SLPSO performs best and RDIM-PSO ranks second. To show the modeling effectiveness of different optimization algorithms, the modeling curves for training and test are shown in Figure 16, and the absolute errors are also displayed in the corresponding curves for the algorithms. Figure 16 indicates that RDIM-PSO outperforms other algorithms for the testing error of ANN. The convergence process according to FES is shown in Figure 17, which shows that the convergence speed of RDIM-PSO is faster than that of the other algorithms.



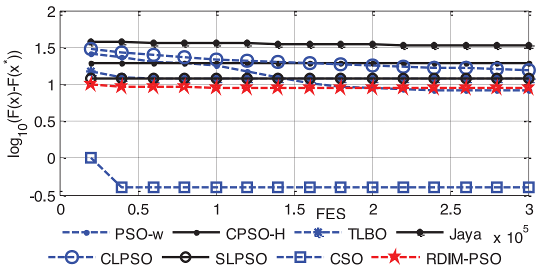
(a)



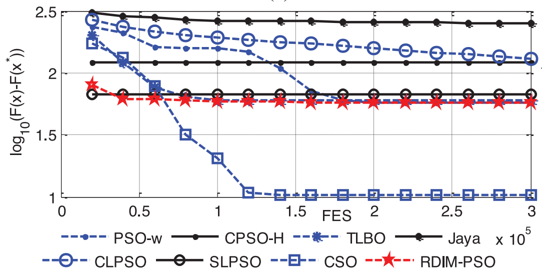
(b)



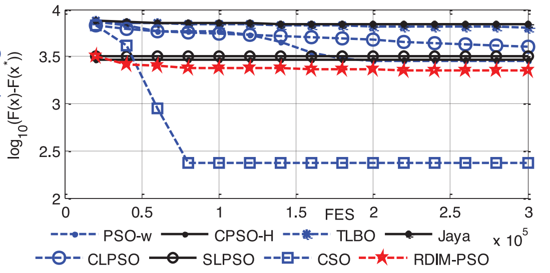
(c)



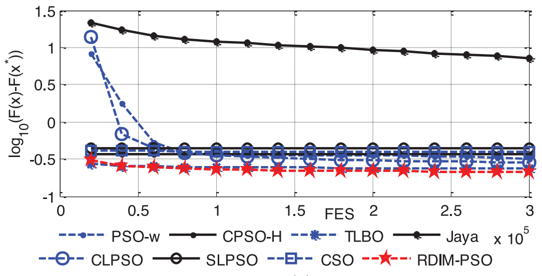
(d)



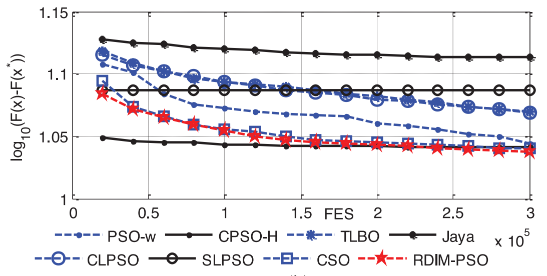
(e)



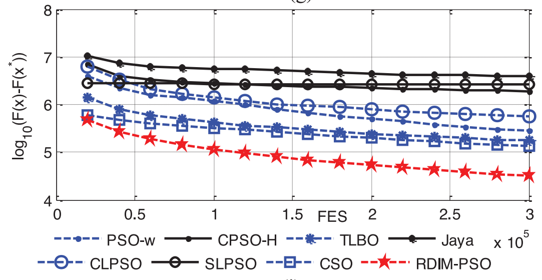
(f)



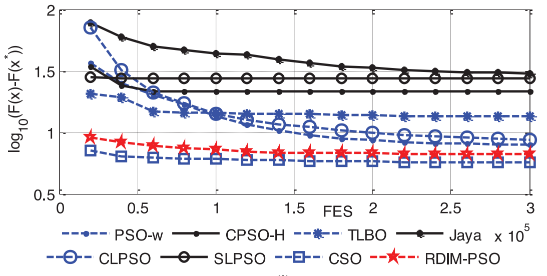
(g)



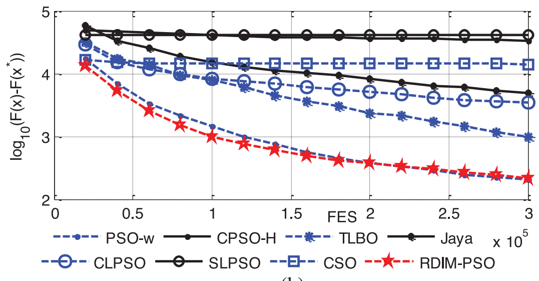
(h)



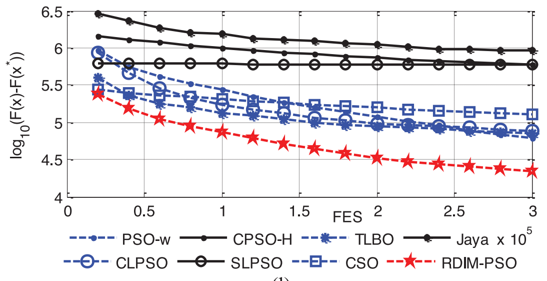
(i)



(j)



(k)



(l)

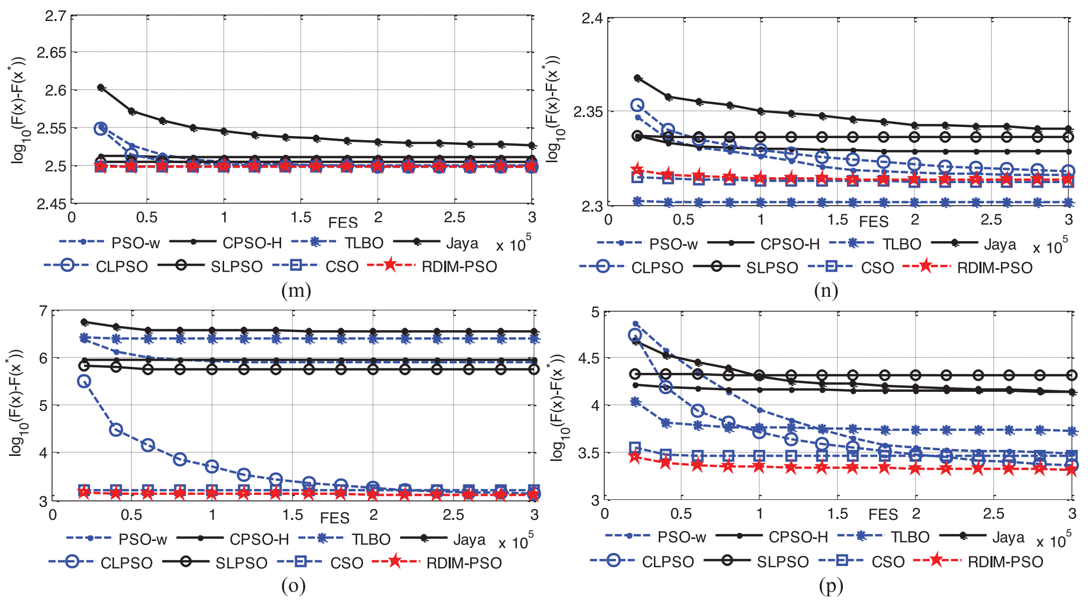
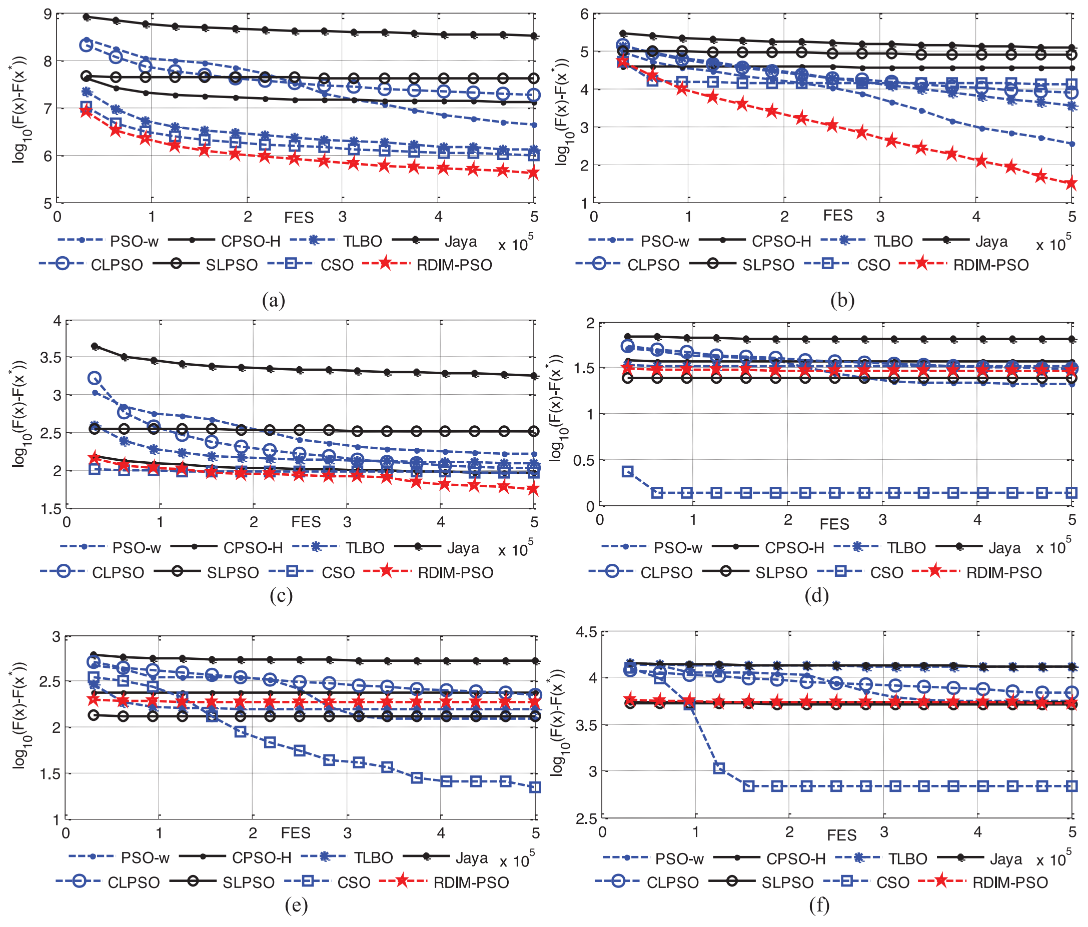


Figure 12 | Evolution of the mean function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO *versus* the number of *FES* on 16 test problems with $D = 30$. (a) f_1; (b) f_3; (c) f_4; (d) f_6; (e) f_9; (f) f_{11}; (g) f_{14}; (h) f_{16}; (i) f_{17}; (j) f_{19}; (k) f_{20}; (l) f_{21}; (m) f_{23}; (n) f_{25}; (o) f_{29}; (p) f_{30}.



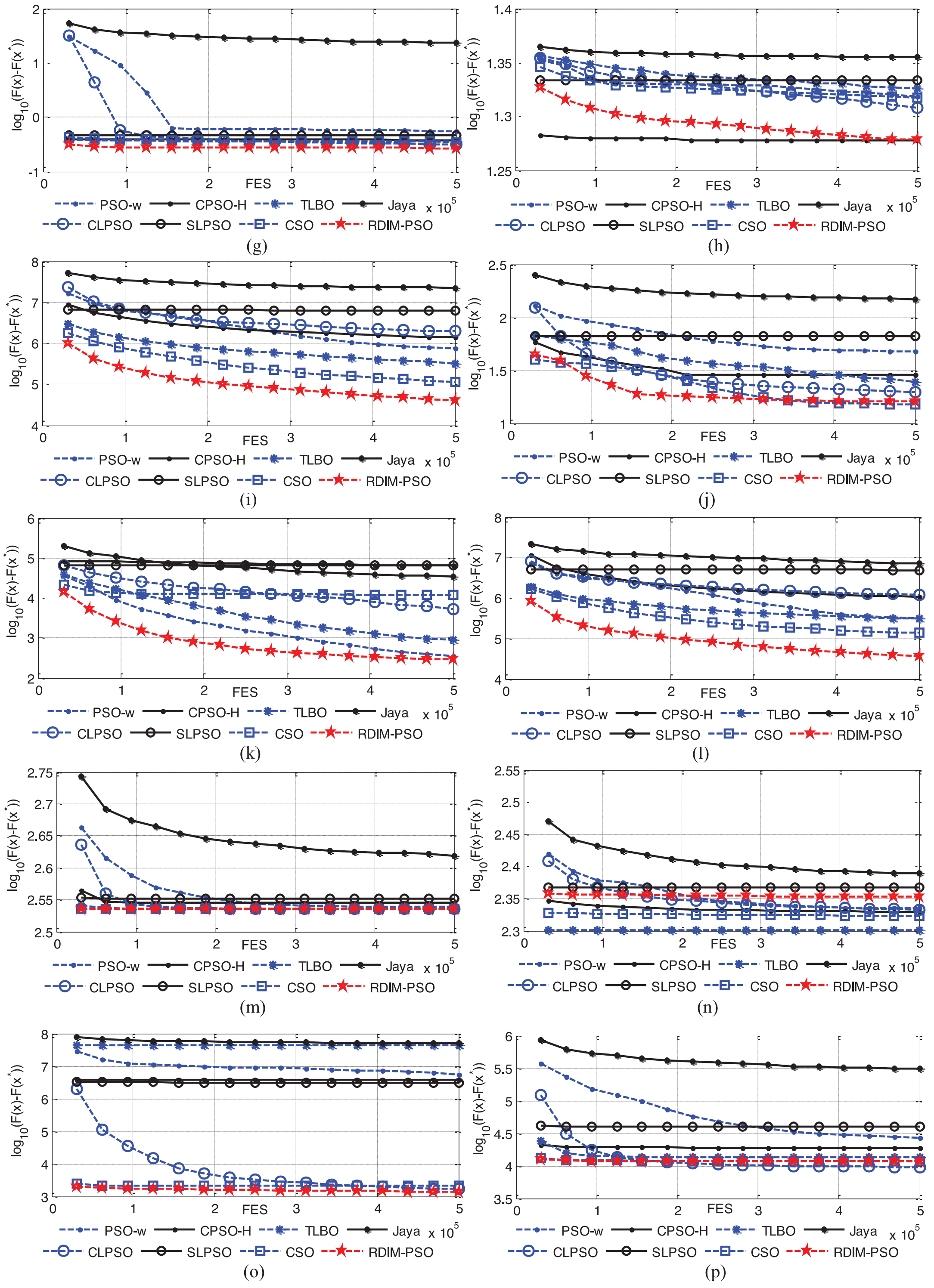
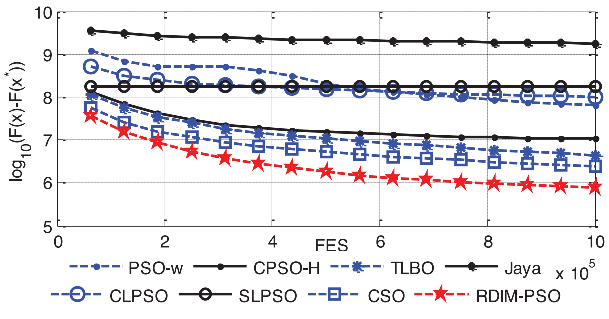
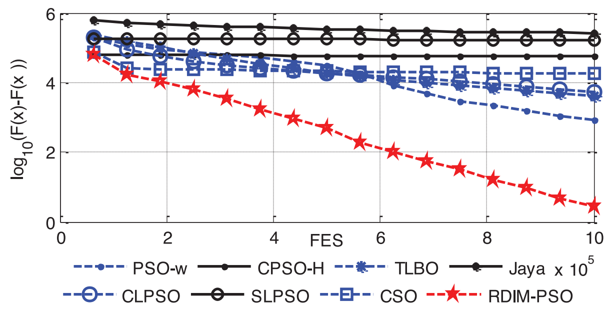


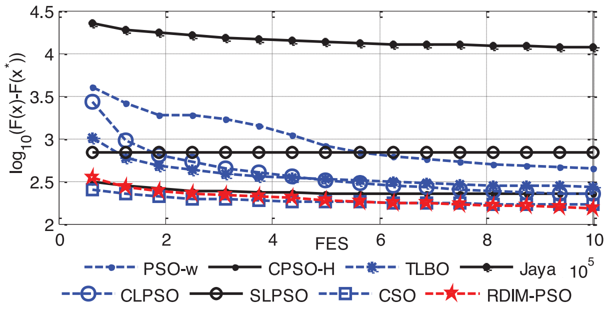
Figure 13 Evolution of the mean function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO *versus* the number of *FES* on 16 test problems with $D = 50$. (a) f_1; (b) f_3; (c) f_4; (d) f_6; (e) f_9; (f) f_{11}; (g) f_{14}; (h) f_{16}; (i) f_{17}; (j) f_{19}; (k) f_{20}; (l) f_{21}; (m) f_{23}; (n) f_{25}; (o) f_{29}; (p) f_{30}.


log10(F(x)-F(x*))
FES
x 10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

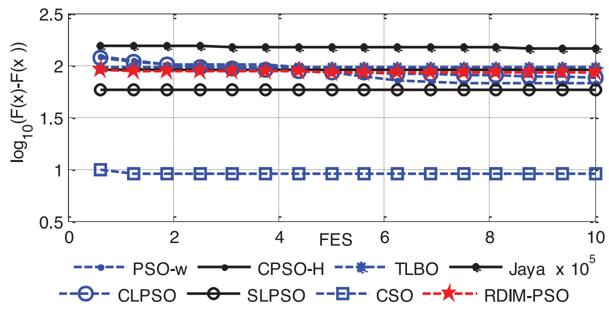
(a)


log10(F(x)-F(x*))
FES
x 10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

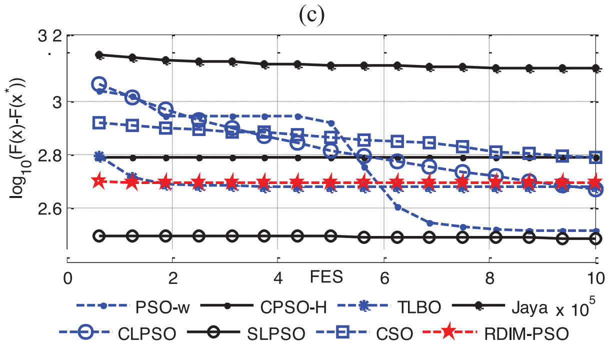
(b)


log10(F(x)-F(x*))
FES
10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

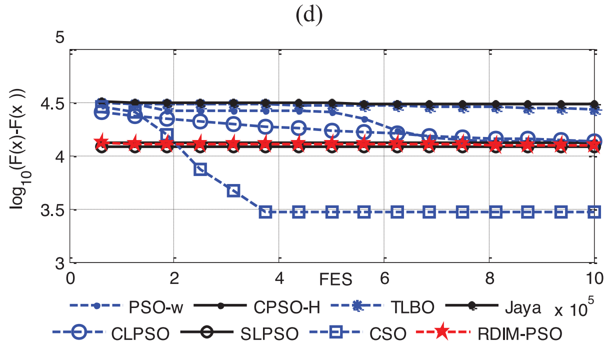
(c)


log10(F(x)-F(x*))
FES
x 10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

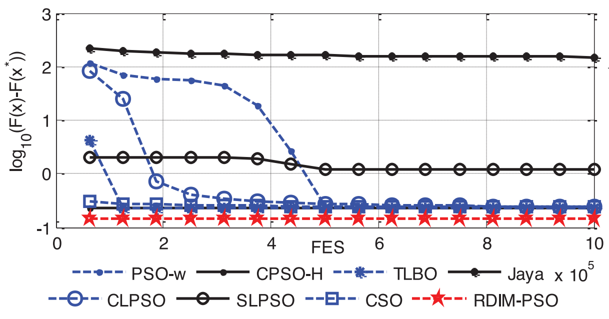
(d)


log10(F(x)-F(x*))
FES
x 10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

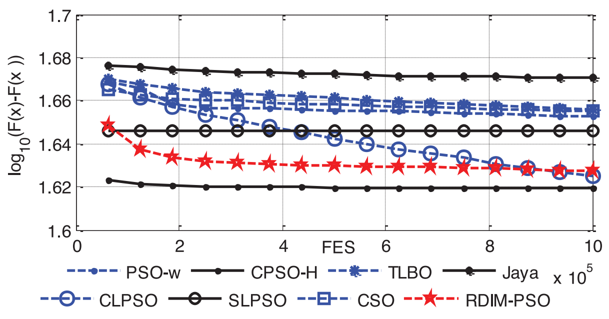
(e)


log10(F(x)-F(x*))
FES
x 10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

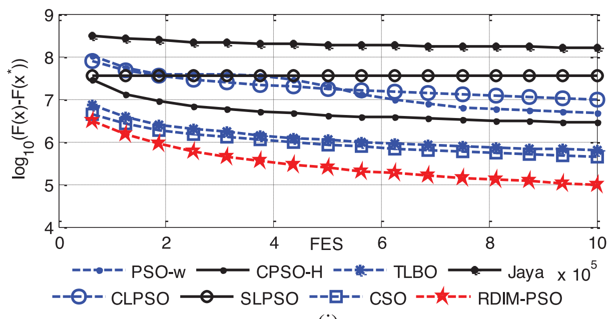
(f)


log10(F(x)-F(x*))
FES
x 10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

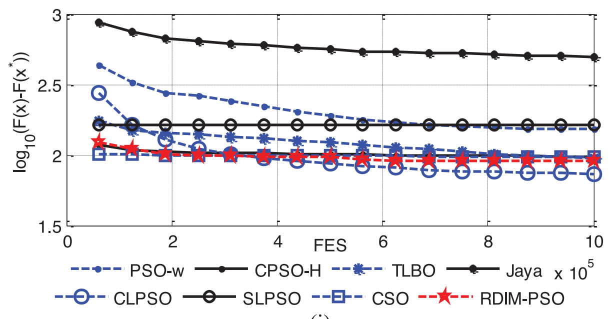
(g)


log10(F(x)-F(x*))
FES
x 10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

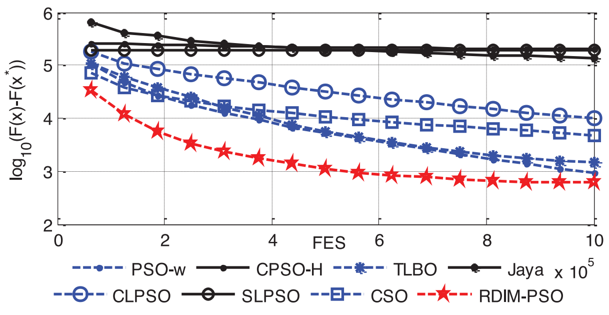
(h)


log10(F(x)-F(x*))
FES
x 10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

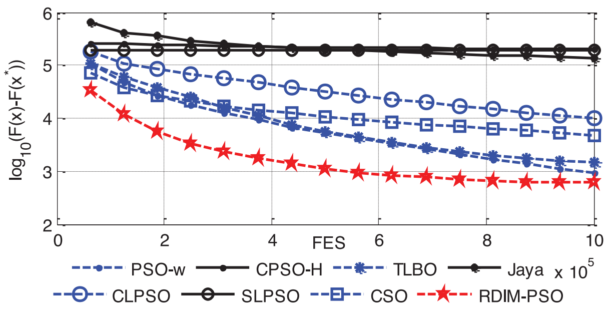
(i)


log10(F(x)-F(x*))
FES
x 10^5
PSO-w
CPSO-H
TLBO
Jaya
CLPSO
SLPSO
CSO
RDIM-PSO

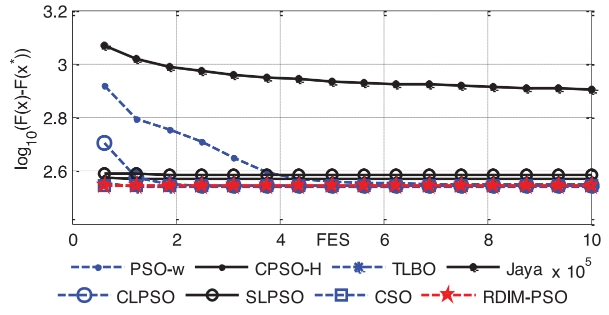
(j)



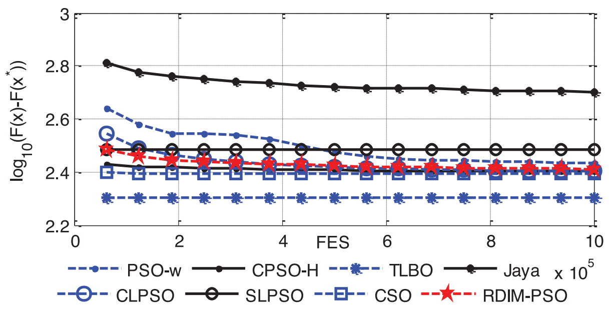
(k)



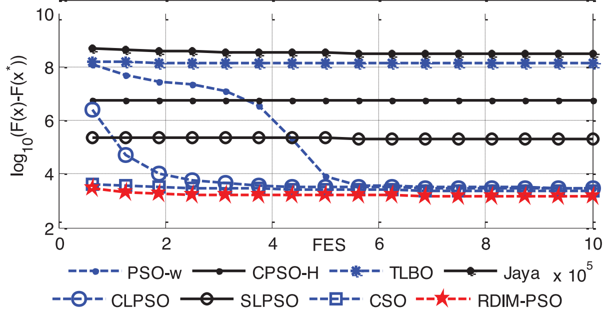
(l)



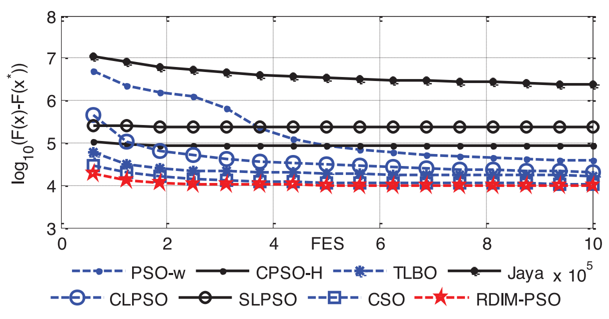
(m)



(n)



(o)



(p)

Figure 14 Evolution of the mean function error values derived from PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO *versus* the number of *FES* on 16 test problems with $D = 100$. (a) f_1; (b) f_3; (c) f_4; (d) f_6; (e) f_9; (f) f_{11}; (g) f_{14}; (h) f_{16}; (i) f_{17}; (j) f_{19}; (k) f_{20}; (l) f_{21}; (m) f_{23}; (n) f_{25}; (o) f_{29}; (p) f_{30}.

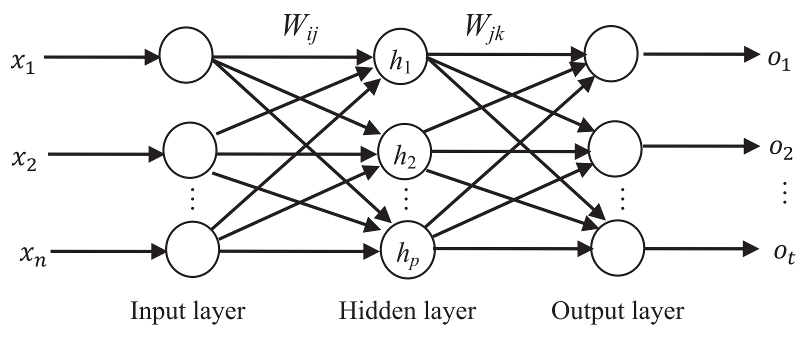


Figure 15 Structure of a three-layer feed forward neural network.

5.2. Lorenz Chaotic Time Series Prediction

The Lorenz chaotic system [56] is described as follows:

$$\begin{cases} \dfrac{dx}{dt} = a\left(y - x\right) \\ \dfrac{dy}{dt} = bx - xz + cy \\ \dfrac{dz}{dt} = xy - dz \end{cases} \tag{20}$$

where $a = 10$, $b = 28$ and $c = 8/3$.

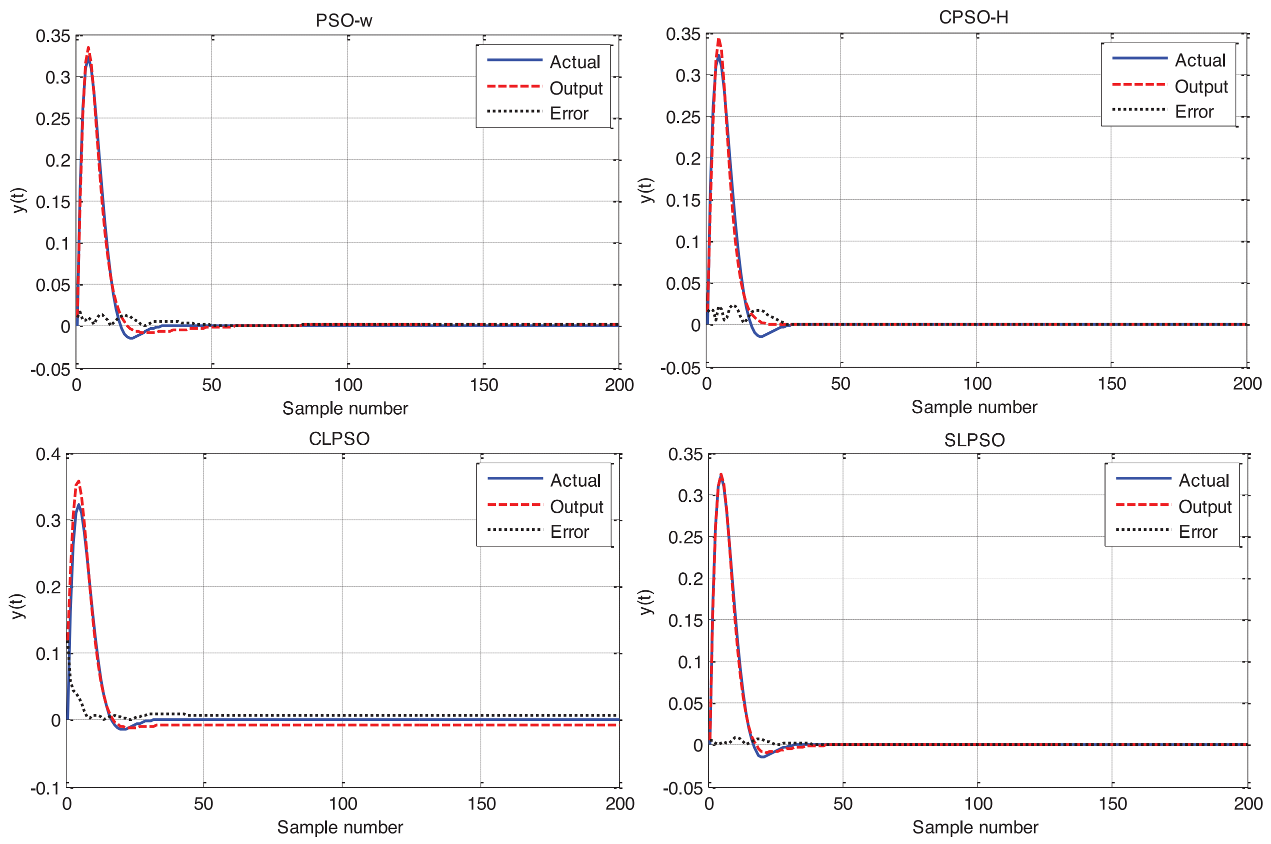
Our task of this study is to use the earlier three points $x(t-6)$, $x(t-3)$ and $x(t)$ to predict the next point $x(t+1)$. The goal of this experiment is to build the single-step-ahead prediction model of chaotic time series of Lorenz, shown as follows [56]:

$$\hat{x}(t+1) = f(x(t-6), x(t-3), x(t)) \tag{21}$$

To predict the Lorenz model, there are three input units, five hidden units and one output units in the three-layer feed-forward ANN. Like the experiment of Section 5.1, $NP = 50$, $FES = 10{,}000$. The 10,000 data are selected from the model, in which the first 8000 data are discarded. The rest 2000 data are normalized and divided into two groups. One group has 1500 points which act as the training samples, and the other group has 500 points which act as the testing samples. MSE is chosen as the fitness function of all algorithms. The merits in terms of Mean and Std of the training and testing process are shown in Table 21. These results come from 30 independent runs of each of the eight algorithms.

Table 20 Comparisons between PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on MSE.

Algorithm	Training Error		Testing Error	
	Mean	Std	Mean	Std
PSO-w	2.0497e-004	1.3639e-004	6.2057e-003	3.1014e-002
CPSO-H	9.8596e-004	1.0796e-003	2.9835e-002	6.2301e-002
CLPSO	5.9241e-003	4.5272e-003	3.4094e-003	2.9087e-003
SLPSO	6.8686e-005	2.9278e-005	7.9149e-005	2.5779e-004
TLBO	3.7084e-003	2.1537e-003	3.2755e-003	7.3220e-003
CSO	1.0683e-004	5.5269e-005	1.6601e-004	4.9784e-004
Jaya	1.5481e-002	1.0563e-002	8.4399e-003	6.8823e-003
RDIM-PSO	4.5627e-005	3.0486e-005	6.8852e-005	2.2011e-004



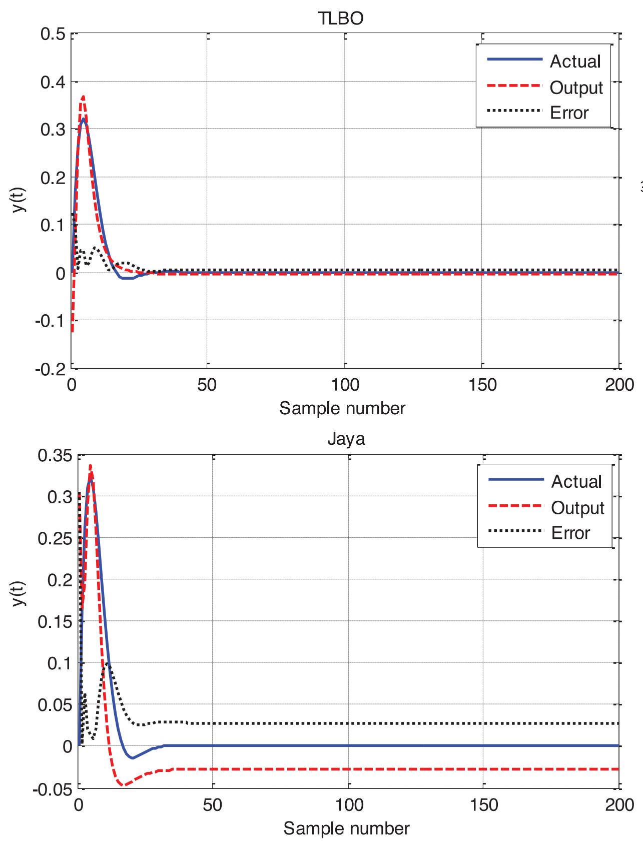


Figure 16 The model and absolute error of different algorithms for SISO system.

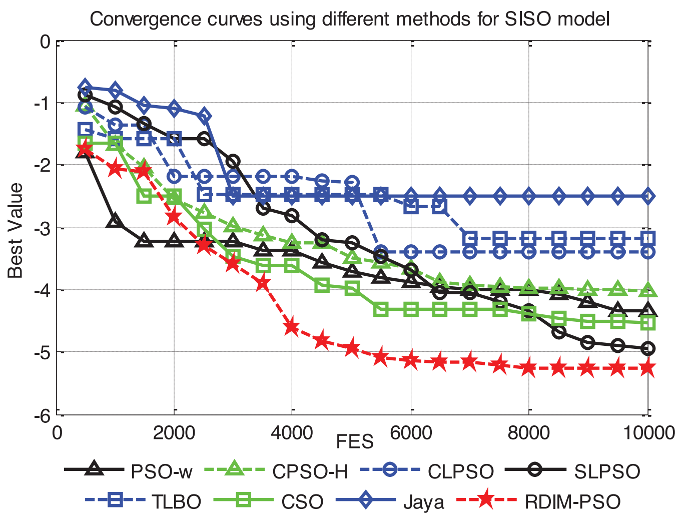


Figure 17 The average fitness curves of eight algorithms for SISO system.

Table 21 shows that RDIM-PSO outperformed the other algorithms. The mean deviation of the training error and testing error obtained by RDIM-PSO is ranked first of the eight algorithms. Concerning the standard deviation of the training error and testing error, CSO performs best and RDIM-PSO ranks second. The training and testing curves for 2000 samples are shown in Figure 18. The convergence process of the average MSE for all algorithms is displayed in Figure 19. The figure shows that the convergence speed of RDIM-PSO is faster than that of the other algorithms for

Table 21 Comparisons between PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on MSE.

Algorithm	Training Error		Testing Error	
	Mean	Std	Mean	Std
PSO-w	4.5861e-004	3.4699e-004	4.7360e-004	3.6247e-004
CPSO-H	1.2946e-002	1.8885e-002	1.3313e-002	1.8970e-002
CLPSO	5.2244e-003	2.1412e-003	5.3556e-003	2.1398e-003
SLPSO	2.8744e-004	2.6103e-004	3.0806e-004	2.7996e-004
TLBO	5.0261e-003	3.0655e-003	5.1453e-003	3.1560e-003
CSO	1.9086e-004	1.5304e-004	2.0688e-004	1.6889e-004
Jaya	1.3970e-002	6.4265e-003	1.4490e-002	6.8396e-003
RDIM-PSO	1.6213e-004	1.7993e-004	1.6839e-004	1.8640e-004

Lorenz system. The experimental results indicate that the approximate accuracy of RDIM-PSO is high.

5.3. Box–Jenkins Chaotic Time Series

The third problem is the Box–Jenkins gas furnace (Box–Jenkins) prediction. The Box–Jenkins gas furnace dataset was recorded from a combustion process of a methane–air mixture. There are originally 296 data points $(x(t), u(t))$, from $t = 1$ to $t = 296$. $x(t)$ is the output CO_2 concentration and $u(t)$ is the input gas flowing rate.

Our task of this study is to use the earlier points $u(t-4)$ and $x(t-1)$ to predict the value of the time series at the point $x(t+1)$. Our experimental aim is to build the single-step-ahead prediction model of chaotic time series, shown as followings [57,58]:

$$\hat{x}(t) = f(u(t-4), x(t-1), x(t)) \tag{22}$$

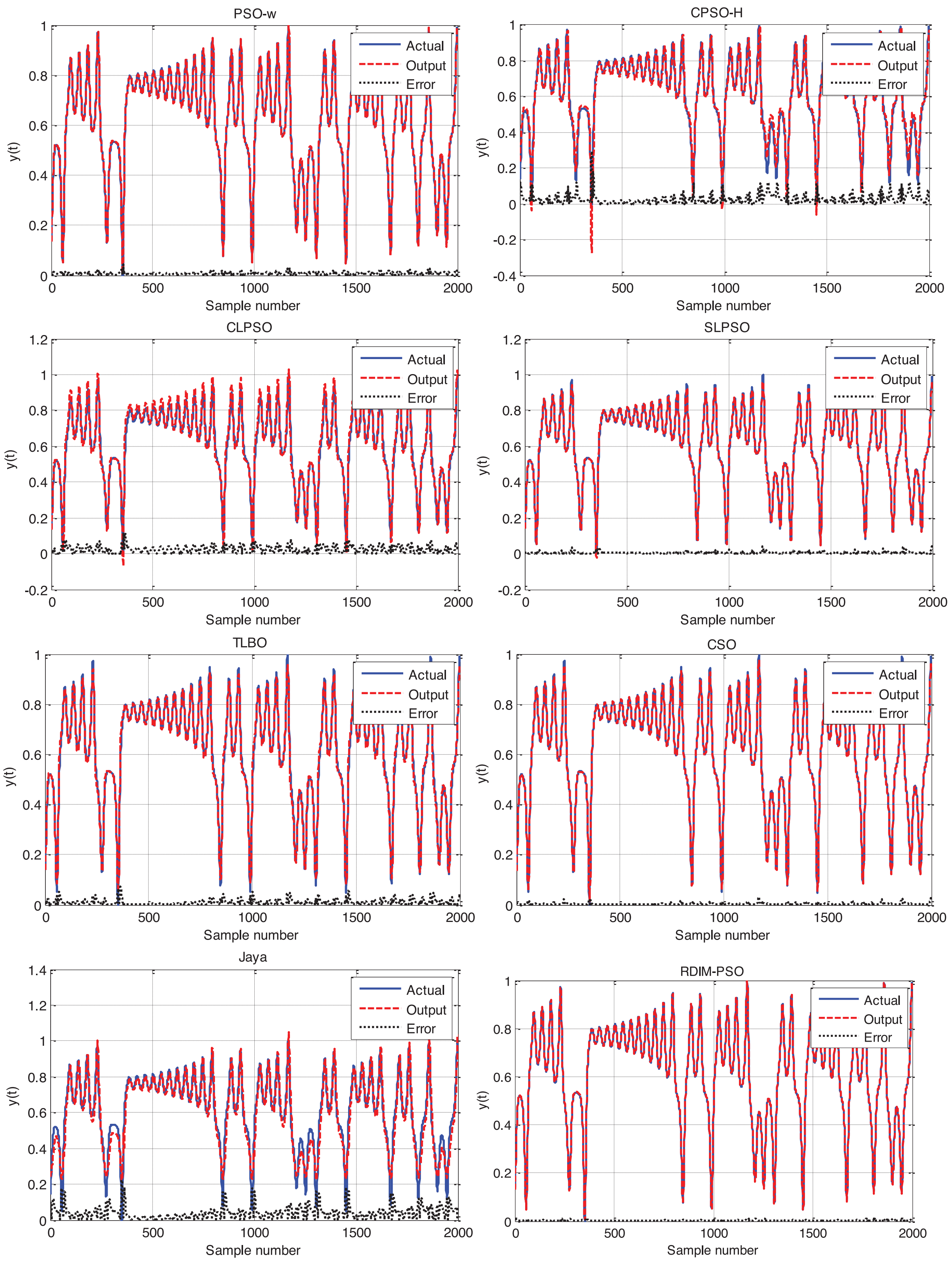


Figure 18 The model and absolute error of different algorithms for Lorenz system.

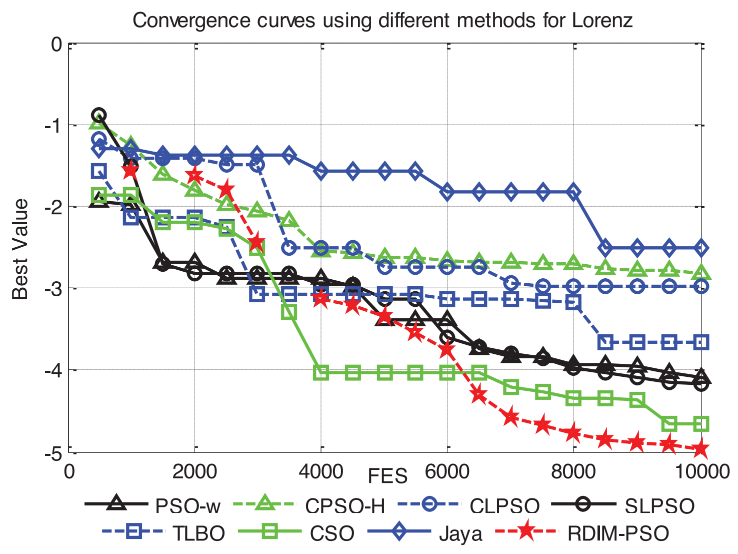


Figure 19 The average fitness curves of eight algorithms for Lorenz system.

Table 22 Comparisons between PSO-w, CPSO-H, CLPSO, SLPSO, TLBO, CSO, Jaya and RDIM-PSO on MSE.

Algorithm	**Training Error**		**Testing Error**	
	Mean	**Std**	**Mean**	**Std**
PSO-w	2.7440e-003	2.6438e-003	4.2470e-003	3.9781e-003
CPSO-H	7.9915e-002	7.8607e-002	1.2020e-001	1.1165e-001
CLPSO	4.3867e-002	2.8976e-002	4.4900e-002	3.0854e-002
SLPSO	7.8569e-004	5.8067e-004	1.7505e-003	1.4053e-003
TLBO	3.1100e-003	2.2400e-003	4.0338e-003	2.8770e-003
CSO	1.2168e-003	9.2267e-004	2.4789e-003	1.2699e-003
Jaya	1.2345e-001	8.9826e-002	1.3974e-001	1.2677e-001
RDIM-PSO	6.6152e-004	5.0912e-004	1.4904e-003	1.0095e-003

To predict the Box–Jenkins model, there are two input units, five hidden units and one output units in the three-layer feed-forward ANN. 292 data pairs are selected from the real model. The data are divided into two groups. One group has 142 data pairs which act as the training samples. The other group has 150 data pairs which act as the testing samples. Like the experiment of Section 5.1, $NP = 50$, $FES = 5000$. The merits in terms of the mean optimum solution (Mean) and the standard deviation of the solutions (Std) in the training and testing process are shown in Table 16. These results come from 30 independent runs of each of the eight algorithms.

From Table 22, it is observed that RDIM-PSO outperformed other seven algorithms in the mean MSE and std MSE in both training and testing cases. Figures 20 and 21 represent the training and testing curves and the convergence process of the average MSE for all algorithms, respectively. The figure shows that the convergence speed of RDIM-PSO is faster than that of the other algorithms for Box–Jenkins system. Then, it can be concluded that RDIM-PSO is the most effective learning algorithm for training the model.

6. CONCLUSIONS

This paper presents an enhanced PSO based on reference direction and inverse model. In the proposed algorithm, the reference particles are first selected according to their fitness and diversity contribution with non-dominated sorting. The dynamic neighborhood strategy is employed to divide the population into several sub-swarms with the reference directions. Next, Gaussian process-based inverse model are introduced to generate equilibrium particles by sampling the objective space. Finally, the velocity update strategy is utilized to improve the convergence of the algorithm. Since fitness and diversity information is simultaneously considered to select the best particle in the dynamic neighborhood, a good balance between exploration and exploitation can be ensured. Based on the results of the eight algorithms on CEC2014 test problems and chaotic time series prediction problems, it can be concluded that RDIM-PSO significantly improves the performance of the original PSO algorithm when compared with seven other optimization algorithms.

Future work will extend the application domain of RDIM-PSO, such as Radial basis function networks, Kohonen networks, etc. In addition, we will focus on making full use of the problem landscape information to design a stronger exploration strategy because RDIM-PSO cannot find the global optimum for some composite problems. Finally, RDIM-PSO will be expected to solve the multiobjective optimization problems and constrained optimization problems.

CONFLICT OF INTERESTS

The authors declare they have no conflicts of interest.

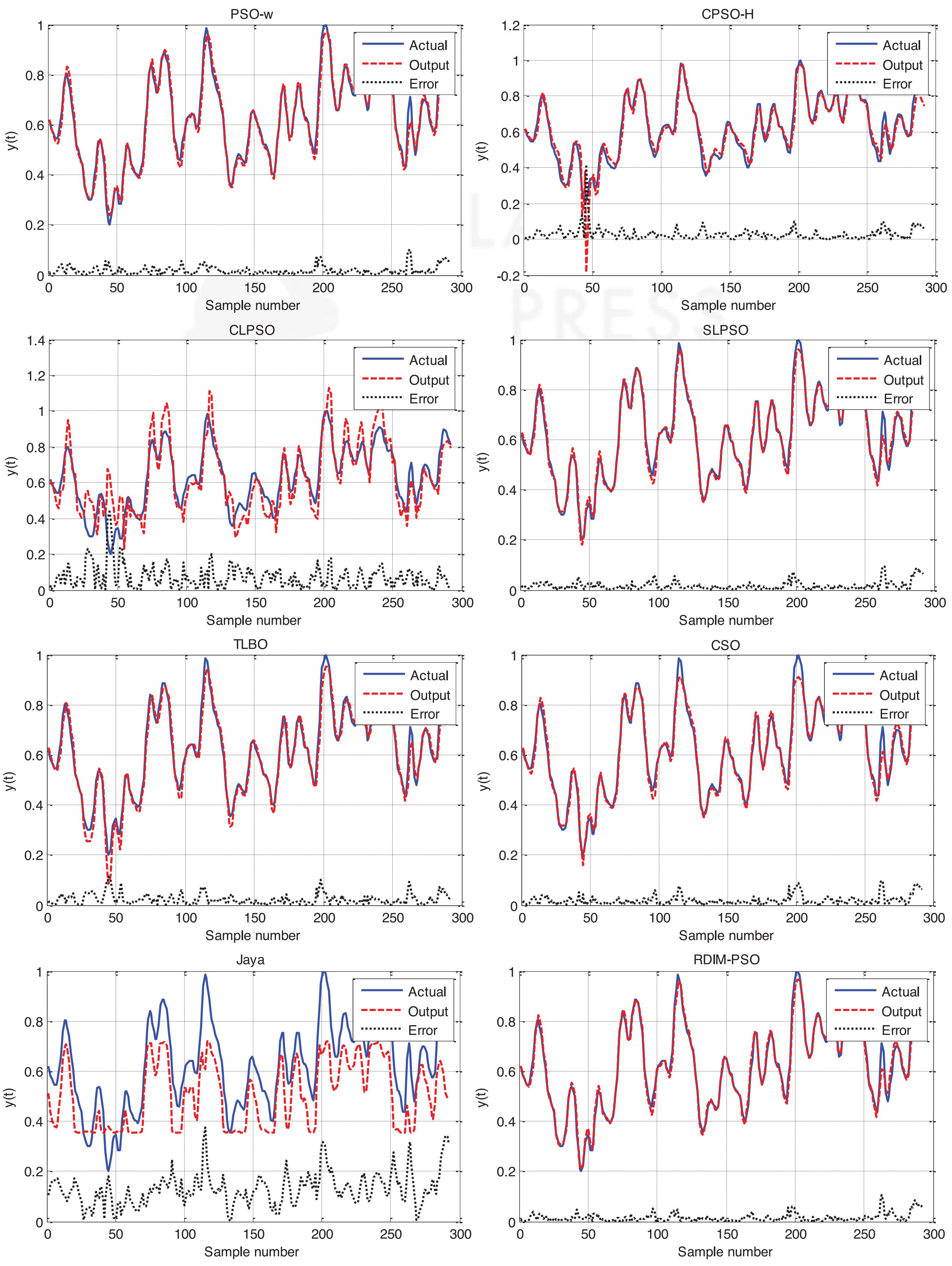


Figure 20 The model and absolute error of different algorithms for Box–Jenkins system.

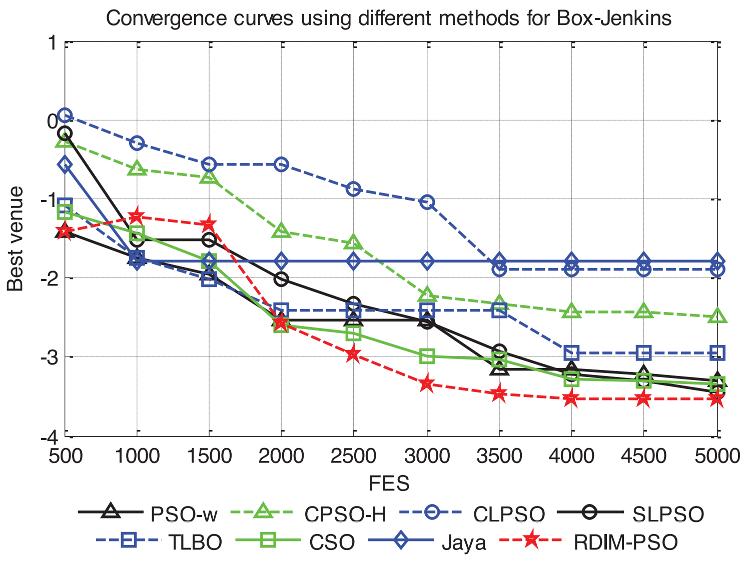


Figure 21 The average fitness curves of eight algorithms for Box–Jenkins system.

DATA AVAILABILITY STATEMENT

The experimental data used to support the findings of this study are included within the article.

AUTHORS' CONTRIBUTIONS

All authors contributed to the work. All authors read and approved the final manuscript.

ACKNOWLEDGMENTS

This research is partly supported by the Doctoral Foundation of Xi'an University of Technology (112-451116017), National Natural Science Foundation of China under Project Code (61803301, 61773314) and the Scientific Research Foundation of the National University of Defense Technology (grant no. ZK18-03-43).

REFERENCES

[1] J.O. Kephart, Learning from nature, Science. 331 (2011), 682–683.

[2] W. Banzhaf, G. Beslon, S. Christensen, J.A. Foster, F. Képès, W. Lefort, J.F. Miller, M. Radman, J.J. Ramsden, From artificial evolution to computational evolution: a research agenda, Nature. 7 (2006), 729–735.

[3] S. Mirjalili, A. Lewis, The whale optimization algorithm, Adv. Eng. Softw. 95 (2016), 51–67.

[4] J.H. Holland, Adaptation in Natural and Artificial Systems, University of Michigan Press, The MIT Press, London, 1975.

[5] J.R. Koza, Genetic Programming: On the Programming of Computers by Means of Natural Selection, MIT Press, Cambridge, MA, 1992, https://dl.acm.org/doi/book/10.5555/138936.

[6] R. Storn, K.V. Price, Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces, J. Global Optim. 11 (1997), 341–359.

[7] N. Hansen, A. Ostermeier, Completely derandomized self-adaptation in evolution strategies, Evol. Comput. 9 (2001), 159–195.

[8] S. Kirkpatrick, C.D. Gelatt Jr, M.P. Vecchi, Optimization by simulated annealing, Science. 220 (1983), 671–680.

[9] A.Y.S. Lam, V.O.K. Li, Chemical-reaction-inspired metaheuristic for optimization, IEEE Trans. Evol. Comput. 14 (2010), 381–399.

[10] Y. Tan, Y. Zhu, Fireworks algorithm for optimization, in Proceedings of the 2nd International Conference on Swarm Intelligence, Lecture Notes in Computer Science, Beijing, China, 2010, vol. 6145, pp. 355–364.

[11] Y.H. Shi, Brain storm optimization algorithm, in Advances in Swarm Intelligence, Series Lecture Notes in Computer Science, Chongqing, China, 2011, vol. 6728, 303–309.

[12] J. Kennedy, R. Eberhart, Particle swarm optimization, in Proceedings of International Conference on Neural Networks, Perth, WA, Australia, 1995, pp. 1942–1948.

[13] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. 39 (2007), 459–471.

[14] R.V. Rao, V.J. Savsani, D.P. Vakharia, Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems, Inf. Sci. 183 (2012), 1–15.

[15] M.R. Tanweer, S. Suresh, N. Sundararajan, Self-regulating particle swarm optimization algorithm, Inf. Sci. 294 (2015), 182–202.

[16] F. van den Bergh, A.P. Engelbrecht, A cooperative approach to particle swarm optimization, IEEE Trans. Evol. Comput. 8 (2004), 225–239.

[17] R. Cheng, Y. Jin, A social learning particle swarm optimization algorithm for scalable optimization, Inf. Sci. 291 (2015), 43–60.

[18] R. Cheng, Y. Jin, K. Narukawa, B. Sendhoff, A multiobjective evolutionary algorithm using Gaussian process based inverse modeling, IEEE Trans. Evol. Comput. 19 (2015), 838–856.

[19] S. Jiang, S. Yang, A strength Pareto evolutionary algorithm based on reference direction for multi-objective and many-objective optimization, IEEE Trans. Evol. Comput. 21 (2017), 329–346.

[20] Y.H. Shi, R.C. Eberhart, A modified particle swarm optimizer, in Proceedings of the IEEE World Congress on Computational Intelligence, Anchorage, AK, 1998, pp. 69–73.
[21] M. Clerc, J. Kennedy, The particle swarm-explosion, stability, and convergence in a multidimensional complex space, IEEE Trans. Evol. Comput. 6 (2002), 58–73.
[22] M.S. Nobile, P. Cazzaniga, D. Besozzi, R. Colombo, Fuzzy self-turning PSO: a settings-free algorithm for global optimization, Swarm Evol. Comput. 39 (2018), 70–85.
[23] M. Hu, T.F. Wu, J.D. Weir, An adaptive particle swarm optimization with multiple adaptive methods, IEEE Trans. Evol. Comput. 17 (2013), 705–720.
[24] K. Chen, F.Y. Zhou, A. Liu, Chaotic dynamic weight particle swarm optimization for numerical function optimization, Knowl. Based Syst. 139 (2018), 23–40.
[25] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Trans. Evol. Comput. 10 (2006), 281–295.
[26] Z.H. Zhan, J. Zhang, Y. Li, Y.H. Shi, Orthogonal learning particle swarm optimization, IEEE Trans. Evol. Comput. 15 (2011), 832–847.
[27] H.B. Quyang, L.Q. Gao, S. Li, X.Y. Kong, Improved global best guided particle swarm optimization with learning operation for global optimization problems, Appl. Soft Comput. 52 (2017), 987–1008.
[28] C.L. Sun, Y.C. Jin, R. Cheng, J.L. Ding, J.C. Zeng, Surrogate-assisted cooperative swarm optimization of high-dimensional expensive problems, IEEE Trans. Evol. Comput. 21 (2017), 644–660.
[29] W. Sun, A.P. Lin, H.S. Yu, Q.K. Liang, G.H. Wu, All-dimension neighborhood based particle swarm optimization with randomly selected neighbors, Inf. Sci. 405 (2017), 141–156.
[30] L. Wang, B. Yang, J. Orchard, Particle swarm optimization using dynamic tournament topology, Appl. Soft Comput. 48 (2016), 584–596.
[31] R. Jensi, G.W. Jiji, An enhanced particle swarm optimization with levy flight for global optimization, Appl. Soft Comput. 43 (2016), 248–261.
[32] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm simpler, maybe better, IEEE Trans. Evol. Comput. 8 (2004), 204–210.
[33] P.N. Suganthan, Particle swarm optimiser with neighbourhood operator, in Proceedings of the IEEE Congress on Evolutionary Computation, Washington, DC, 1999, vol. 3, pp. 1958–1962.
[34] M. Nasir, S. Das, D. Maity, S. Sengupta, U. Halder, P.N. Suganthan, A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization, Inf. Sci. 209 (2012), 16–36.
[35] W.X. Ye, W.Y. Feng, S.H. Fan, A novel multi-swarm particle swarm optimization with dynamic learning strategy, Appl. Soft Comput. 61 (2017), 832–843.
[36] I. Maruta, T.H. Kim, D. Song, T. Sugie, Synthesis of fixed-structure robust controllers using a constrained particle swarm optimizer with cyclic neighborhood topology, Expert Syst. Appl. 40 (2013), 3595–3605.
[37] X.C. Zhao, W.Q. Lin, J.L. Hao, X.Q. Zuo, J.H. Yuan, Clustering and pattern search for enhancing particle swarm optimization with Euclidean spatial neighborhood search, Neurocomputing. 171 (2016), 966–981.
[38] H. Wang, H. Sun, C.H. Li, S. Rahnamayan, J. Pan, Diversity enhanced particle swarm optimization with neighborhood search, Inf. Sci. 223 (2013), 119–135.
[39] W.D. Chang, A modified particle swarm optimization with multiple subpopulations for multimodal function optimization problems, Appl. Soft Comput. 33 (2015), 170–182.
[40] S. Das, A. Abraham, U.K. Chakraborty, A. Konar, Differential evolution using a neighborhood-based mutation operator, IEEE Trans. Evol. Comput. 13 (2009), 526–553.
[41] L.Z. Cui, G.H. Li, Q.Z. Lin, J.Y. Chen, N. Lu, Adaptive differential evolution algorithm with novel mutation strategies in multiple sub-populations, Comput. Oper. Res. 67 (2016), 155–173.
[42] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: NSGA-II, IEEE Trans. Evol. Comput. 6 (2002), 182–197.
[43] J.H. Wang, J.J. Liao, Y. Zhou, Y.Q. Cai, Differential evolution enhanced with multiobjective sorting-based mutation operators, IEEE Trans. Cybern. 44 (2014), 2792–2805.
[44] R. Cheng, Y. Jin, A competitive swarm optimizer for large scale optimization, IEEE Trans. Cybern. 45 (2015), 191–204.
[45] R.V. Rao, Jaya: a simple and new optimization algorithm for solving constrained and unconstrained optimization problems, Int. J. Ind. Eng. Comp. 7 (2016), 19–34.
[46] J. Cheng, G. Zhang, F. Neri, Enhancing distributed differential evolution with multicultural migration for global numerical optimization, Inf. Sci. 247 (2013), 72–93.
[47] P.N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y.P. Chen, A. Auger, S. Tiwari, Problem definitions and evaluation criteria for the CEC2005 special session on real-parameter optimization, 2005. http://www.ntu.edu.sg/home/EPNSugan
[48] Y. Wang, Z.X. Cai, Q.F. Zhang, Differential evolution with composite trial vector generation strategies and control parameters, IEEE Trans. Evol. Comput. 15 (2011), 55–66.
[49] J. Alcalá-Fdez, L. Sánchez, S. García, M.J. del Jesus, S. Ventura, J.M. Garrell, J. Otero, C. Romero, J. Bacardit, V.M. Rivas, J.C. Fernández, F. Herrera, KEEL: a software tool to assess evolutionary algorithms to data mining problems, Soft Comput. 13 (2009), 307–318.
[50] R. Mendes, P. Cortez, M. Rocha, J. Neves, Particle swarms for feedforward neural network training, in Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, HI, 2002, pp. 1895–1899.
[51] K. Demertzis, L. Iliadis, Adaptive elitist differential evolution extreme learning machines on big data: intelligent recognition of invasive species, Adv. Intell. Syst. Comput. 529 (2017), 1–13.
[52] U. Seiffert, Multiple layer perceptron training using genetic algorithms, in Proceedings of the European Symposium on Artificial Neural Networks, Bruges, Belgium, 2001, pp. 159–164.
[53] C. Blum, K. Socha, Training feed-forward neural networks with ant colony optimization: an application to pattern classification, in Proceedings of the International Conference on Hybrid Intelligent System, Rio de Janeiro, Brazil, 2005, pp. 233–238.
[54] S.A. Mirjalili, S.Z.M. Hashim, S.H. Moradian, Training feedforward neural networks using hybrid particle swarm optimization and gravitational search algorithm, Appl. Math. Comput. 218 (2012), 11125–11137.
[55] D. Simon, Biogeography-based optimization, IEEE Trans. Evol. Comput. 12 (2008), 702–713.

[56] V.A. Gromov, A.N. Shulga, Chaotic time series prediction with employment of ant colony optimization, Expert Syst. Appl. 39 (2012), 8474–8478.

[57] L. Zhao, Y. Yang, PSO-based single multiplicative neuron model for time series prediction, Expert Syst. Appl. 36 (2009), 2805–2812.

[58] B. Samanta, Prediction of chaotic time series using computational intelligence, Expert Syst. Appl. 38 (2011), 11406–11411.