information

MDPI

Article

Improving Particle Swarm Optimization Based on Neighborhood and Historical Memory for Training Multi-Layer Perceptron

Wei Li

School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China; liwei@xaut.edu.cn; Tel.: +86-29-8231-2231

Received: 11 November 2017; Accepted: 10 January 2018; Published: 12 January 2018

Abstract: Many optimization problems can be found in scientific and engineering fields. It is a challenge for researchers to design efficient algorithms to solve these optimization problems. The Particle swarm optimization (PSO) algorithm, which is inspired by the social behavior of bird flocks, is a global stochastic method. However, a monotonic and static learning model, which is applied for all particles, limits the exploration ability of PSO. To overcome the shortcomings, we propose an improving particle swarm optimization algorithm based on neighborhood and historical memory (PSONHM). In the proposed algorithm, every particle takes into account the experience of its neighbors and its competitors when updating its position. The crossover operation is employed to enhance the diversity of the population. Furthermore, a historical memory M_w is used to generate new inertia weight with a parameter adaptation mechanism. To verify the effectiveness of the proposed algorithm, experiments are conducted with CEC2014 test problems on 30 dimensions. Finally, two classification problems are employed to investigate the efficiencies of PSONHM in training Multi-Layer Perceptron (MLP). The experimental results indicate that the proposed PSONHM can effectively solve the global optimization problems.

Keywords: neighborhood; particle swarm optimization; historical memory; evolutionary algorithms; classification

1. Introduction

Artificial Neural Networks (ANN) is one of the more significant inventions in the field of soft computing [1]. There are different types of ANNs in which the Feedforward Neural Network (FNN) has been widely used. There are two types of FNN: Single-Layer Perceptron (SLP) [2] and Multi-Layer Perceptron (MLP) [3]. MLP can solve nonlinear problems because it has more than one perceptron. The ANN training process is an optimization process with the aim of finding a set of weights to minimize an error measure [4]. Then, some conventional gradient descent algorithms, such as the Back Propagation (BP) algorithm [5], are used to solve the problem. However, the BP algorithm is prone to getting trapped in local minima because it is highly dependent on the initial values of weights and biases.

To search for the optimal weights of the network, various heuristic optimization methods have been utilized to train FNNs, such as Particle swarm optimization (PSO) [6], Differential evolution (DE) [7], Genetic algorithms (GAs) [8], Ant colony algorithm [9], etc. These evolutionary algorithms (EAs) have been recognized to be effective and efficient for tackling the optimization problems. They have been successfully applied in various scientific and engineering fields, such as optimization, engineering design, neural network training, scheduling, large-scale, constrained, economic problems, multi-objective, forecasting, and clustering [10–17]. However, the EAs are often stuck in a local optimum because of the possible occurrence of premature convergence. It is necessary for the EAs to

address the issue of exploration–exploitation of the search space. To achieve a proper balance between exploration and exploitation during the optimization process, many heuristic algorithms that imitate biological or physical phenomena are proposed. These heuristic algorithms include the derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES) [18], the Simulated Annealing (SA) [19,20], Biogeography-Based Optimizer (BBO) [21], Chemical Reaction Optimization (CRO) [22], Brain Storm Optimization (BSO) [23] and so on. CMA-ES, proposed by Hansen and Ostermeier, adapts the complete covariance matrix of the normal mutation distribution to solve optimization problems. SA, proposed by Kirkpatrick et al., mimics the way that metals cool and anneal. In order to solve the premature convergence, two partial re-initializing solutions strategies are proposed to improve the population diversity in BSO. Inspired by a correspondence between optimization and chemical reaction, CRO is proposed by mimicking what happens to molecules in chemical reactions.

PSO, proposed by Kennedy and Eberhart in 1995 [24], is a simple yet powerful optimization algorithm that imitates the flying and the foraging behavior of birds. The concept of PSO is based on the movement of particles and their personal and best individual experiences [24]. In classical PSO, each particle is attracted by its previous best position (*pbest*) and the global best position (*gbest*). That is to say, the particles adjust their speed and position dynamically by sharing information and experiences of the best particles. Then, the algorithm can converge quickly by using the best solution information in the evolutionary process. However, the information sharing strategy reduces the diversity of the particle swarm, because all particles except itself only share the optimal particle information while ignoring other particles' information. Therefore, the algorithm is prone to premature convergence because of losing diversity too rapidly during the evolutionary process. To improve the performance of PSO, researchers have studied and proposed many improvement strategies based on classical PSO [25–31]. M. Clerc and J. Kennedy proposed PSO with constriction factor (PSOcf) [26] by studying a particle's trajectory as it moves in discrete time. Mendes proposed the fully informed particle swarm (PSOwFIPS) [32], in which the particle uses information from all its neighbors, rather than just the best one. J. J. Liang et al. present the comprehensive learning particle swarm optimizer (CLPSO) utilizing a new learning strategy [33]. T. Krzeszowski et al. propose a modified fuzzy particle swarm optimization method, in which the Takagi–Sugeno fuzzy system is utilized to change the parameters [27]. A. Alfi et al. present an improved fuzzy particle swarm optimization (IFPSO) that uses a fuzzy inertia weight to balance the global and local exploitation abilities [28]. Fuzzy self-turning PSO (FST-PSO), proposed by M. S. Nobile et al. [34], is a novel self-tuning algorithm that exploits fuzzy logic (FL) to calculate the control parameters for each particle. Therefore, FST-PSO realizes a complete settings-free version of PSO.

Obviously, it is impossible to find an algorithm that can solve all the problems. In fact, to develop a new optimization method that can effectively deal with the exploration–exploitation dilemma in some problems during the optimization process remains an important and significant research work.

An innovative element of this work is to propose an improved PSO based on neighborhood and historical memory (PSONHM). Differently from the former PSO, each particle uses the information about the neighborhood and the competitor to update its velocity and position in PSONHM. The inferior particle is recorded as the competitor in the proposed algorithm. Moreover, instead of the same elite (*gbest)* in the former PSO, multiple elites (good solutions) are employed to guide the population toward a promising area. Furthermore, to solve premature convergence, a crossover operator is introduced to make the population disperse. Overall, the main contributions of PSONHM can be described as follows.

(1) Local neighborhood exploration method is introduced to enhance the local exploration ability. With the local neighborhood exploration method, each particle updates its velocity and position with the information of the neighborhood and competitor instead of its own previous information. The method can effectively increase population diversity.

(2) The crossover operator is employed to generate new promising particles and explore new areas of the search space. The multiple elites are employed to guide the evolution of the population instead of *gbest*, and thus avoid the local optima.
(3) Successful parameter settings can reduce the likelihood of being misled and make the particles evolve towards more promising areas. Then, a historical memory M_w, which stores the parameters from previous generations, is used to generate new inertia weights with a parameter adaptation mechanism.
(4) The last contribution of PSONHM is to design a PSONHM-based trainer for MLPs. Classic learning methods, such as Back Propagation (BP), may lead MLPs to local minima rather than the global minimum. Neighborhood method, crossover operator and historical memory can enhance the exploitation and exploration capability of PSONHM. Then, it can help PSONHM find the optimal choice of weights and biases in the ANN and achieve the optimal result.

The remainder of this paper is organized as follows. In Section 2, PSO and its variants are reviewed. Section 3 presents an improving particle swarm optimization based on neighborhood and historical memory (PSONHM). Section 4 reports the experimental results compared with eight well-known EAs on the latest 30 standard benchmark problems listed in the CEC2014 contest. In Section 5, two classification problems are employed to investigate the efficiencies of PSONHM in training MLP. Section 6 gives the conclusions and possible future research.

2. Related Work

PSO is a population-based optimization algorithm that uses interaction between particles to find the optimal solution. In the following, a brief account of basics and improvements of PSO will be given.

2.1. PSO Framework

PSO algorithm, which is inspired by the behaviors of flocks of birds, is a population-based optimization algorithm. Firstly, a randomly population of *NP* particles are generated in a *D*-dimension search space. Each particle is a potential solution. To restrict the change of velocities and control the scope of search, Shi and Eberhart introduced an inertia weight in PSO [35], the corresponding velocity $\mathbf{v}_{i,G}$ and the position $\mathbf{x}_{i,G}$ are updated as follows:

$$\mathbf{v}_{i,G+1} = \omega_G \mathbf{v}_{i,G} + c_1 \mathbf{r}_i (\mathbf{pbest}_{i,G} - \mathbf{x}_{i,G}) + c_2 \mathbf{r}_i (\mathbf{gbest}_G - \mathbf{x}_{i,G}), \tag{1}$$

$$\mathbf{x}_{i,G+1} = \mathbf{x}_{i,G} + \mathbf{v}_{i,G+1}, \tag{2}$$

where $\mathbf{x}_{i,G} = (x^1_{i,G}, x^2_{i,G}, \cdots, x^D_{i,G})$ is the position of the ith particle at generation G. $\mathbf{v}_{i,G} = (v^1_{i,G}, v^2_{i,G}, \cdots, v^D_{i,G})$ is the velocity of particle i. c_1 and c_2 are acceleration coefficients. $\mathbf{r}_i = (r^1_i, r^2_i, \cdots, r^D_i)$ are random numbers generated in the interval [0, 1]. $\mathbf{pbest}_{i,G}$ is the historical best position for ith particle. $\mathbf{gbest}_G$ is the best swarm historical position found so far.

The performance of PSO can be greatly improved by adjusting the inertia weight ω_G. Shi and Eberhart [36] designed a linearly decreasing inertia weight, which is computed as follows:

$$\omega_G = \omega_{max} - \frac{\omega_{max} - \omega_{min}}{MaxGen} G, \tag{3}$$

where ω_{max} and ω_{min} are usually fixed as 0.9 and 0.4. G is the current generation. *MaxGen* is the maximum generation.

2.2. Improved PSO Based on Neighborhood

To get a proper trade-off between exploration and exploitation, neighborhood, which is an important and efficient method, is widely used in evolutionary algorithms. For example, Das et al. [37] propose two kinds of neighborhood models for DE, namely the local neighborhood model and the

global mutation model that can facilitate the exploration and the exploitation of the search space. Omran et al. [38] employ the index-based neighborhood to enhance the DE mutation scheme. The fully informed PSO, proposed by Mendes [32], uses an index-based neighborhood as the basic structure. The contribution of each neighbor was weighted by the goodness of its previous best. A PSO with a neighborhood operator, proposed by Suganthan, gradually increases the local neighborhood size in the search process [39]. Nasir et al. proposed a dynamic neighborhood learning particle swarm optimizer (DNLPSO) [40]. In DNLPSO, the exemplar particle is selected from a neighborhood and the learner particle can learn from the historical information of its neighborhood. The winner's personal best is used as the exemplar. Each particle in the swarm is known as learner particle. Ouyang et al. proposed an improved global-best-guided particle swarm optimization with learning operation (IGPSO) for solving global optimization problems [41]. In IGPSO, the personal best neighborhood learning strategy is employed to effectively enhance the communication among the historical best swarm.

3. Proposed Modified Optimization Algorithm PSONHM

In this section, the details of the proposed algorithm are described. First, the motivations of this paper are given. Then, the neighborhood exploration strategy, the property of stagnation and the inertia weight assignments based on historical memory are presented. Finally, the pseudo-code of the proposed algorithm is shown.

3.1. Motivations

In the canonical PSO, a particle depends on its personal best and the global best to establish a trajectory along the search space. The global best particle guides the swarm to exploit in the search process. Therefore, similar to other population based algorithms, the algorithm experiences premature convergence because of poor diversity. In PSO, all the particles are attracted by *gbest*, so it is possible that the particles will be easily misguided into unpromising areas. In addition, little attention has been paid to utilize the competitor information, which is helpful for maintaining diversity. To overcome the weakness of PSO, we attempt to use the neighborhood model to prevent the population from getting trapped in local minima. With the neighborhood model, each particle can learn from its neighbors and competitor. In this manner, the possibility of misguidance by the elite may be decreased. Then, the crossover operation is introduced to generate new promising particles. Furthermore, it is widely believed that the inertia weight can significantly influence the performance of PSO. However, there is not much work devoted to discussing or using history information to design the inertia weight. We attempt to use historical memory to guide the selection of future inertia weight.

3.2. Neighborhood Exploration Strategy

It is well known that the traditional PSO includes two types of behaviors: cognitive and social. Generally, *gbest*, which represents the best position, is used in social behaviors. However, the algorithm is easy to drop into the local optimum if *gbest* is not near the global optimum. A main issue in the application of PSO is to implement effective exploration and exploitation. In general, the task of exploration is to find the search space where better solutions are existed. On the other hand, the task of exploitation is to realize a fast convergence to the optimum solution.

Next, we investigate the impact of neighborhood mechanism and archive method, which are used in traditional PSO. They are PSO, PSOwFIPS [32] and PSO_Archive, respectively. In PSOwFIPS, the particle uses information from all its neighbors. In PSO_Archive, the inferior particles are added to the archive at each generation. If the archive size exceeds the population size, then some particles are randomly removed from the archive. Then, we employ as a case study benchmark problem f_4 and f_{17} selected from CEC2014 contest benchmark problems. f_4 is a simple multimodal problem, while f_{17} is a hybrid problem. Each problem is executed for 30 runs. The maximal number of function evaluations (*FES*) for all of the compared algorithms is set to $D \times 10{,}000$ with $D = 30$. To evaluate the performance of each algorithm, the minimum value of the *solution error measure*, which is defined as $f(x) - f(x^*)$ is

recorded, where $f(x)$ is the best fitness value found by an algorithm in a run, and $f(x^*)$ is the real global optimization value of a tested problem.

Figure 1 shows that the neighborhood information can effectively enhance PSO performance because the informed individuals can find better solutions with a higher probability. Hence, we come up with the idea that the neighborhood and the competitor should be utilized when stagnation is happening to PSO.

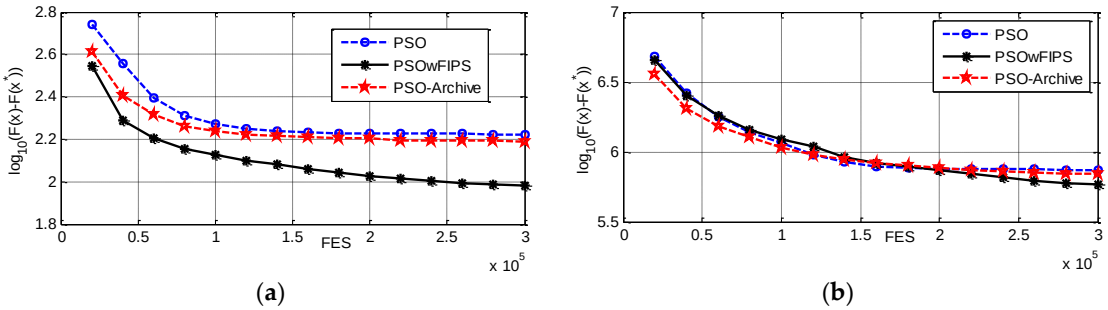

Figure 1. The mean function error values versus the number of FES on test problems. (**a**) f_4; (**b**) f_{17}.

Various neighborhood topologies have been proposed in [26], such as star, wheel, circular, pyramid and 4-clusters. In the proposed algorithm, the ring topology is used because it has better performance compared to other neighborhood topologies. The population is assumed to be organized on a ring topology in connection with their indices. For example, the neighbors of $x_{i,G}$ are $x_{i+1,G}$ and $x_{i-1,G}$. The ring topology used in PSONHM is illustrated in Figure 2.

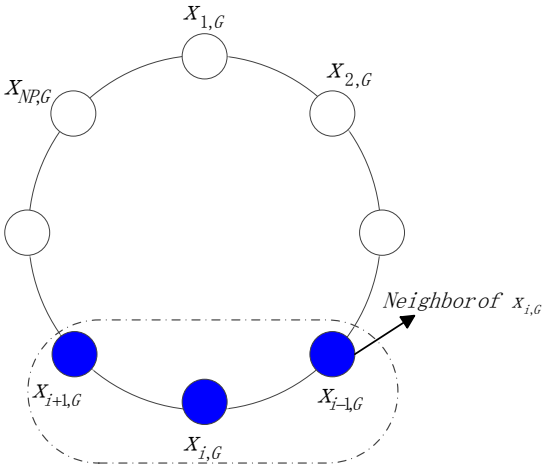

Figure 2. Ring topology of H-neighborhood in PSONHM.

In traditional PSO, all the particles are attracted by *gbest* and the swarm has the tendency to fast converge to the current globally best position. Then, the algorithm may stagnate in the local optimum area because of the rapid convergence. The competitive particles, which may contain some useful information, may be closer to the global minimum. Hence, the difference vector between the personal best position and the competitive particle can be seen as a good direction for exploration. To lessen the influence of *gbest* on the whole population, multiple elites, similar to the better individuals $x^p_{best,g}$ used in DE/current-to-*pbest* [42], are selected to replace *gbest* and instruct updating. In addition, the information of all the neighbors is taken into consideration. The difference vector between the multiple elites and the neighbors can be seen as a good direction for exploitation. Consequently, each particle receives information from its neighbors and competitor, which can

increase the probabilities of generating successful solutions and decreases the probability of premature convergence. The neighborhood exploration strategy is designed as follows:

$$\mathbf{v}_{i,G+1} = \omega_G \mathbf{v}_{i,G} + \mathrm{c}_1 \mathbf{r}_i (\mathbf{pbest}_{i,G} - \widetilde{\mathbf{x}}_{i,G}) + \mathrm{c}_2 \mathbf{r}_i (\mathbf{x}^p_{best,G} - \mathrm{mean}(\mathbf{x}_{n(j,i),G})), \tag{4}$$

where $\widetilde{\mathbf{x}}_{i,G}$ is the competitor of $\mathbf{pbest}_{i,G}$. $\mathbf{pbest}_{i,G}$ denotes the best previously visited position of the ith particle. At G generation, the objective values of ith particle is compared with $\mathbf{pbest}_{i,G}$, The winner is denoted as $\mathbf{pbest}_{i,G+1,}$ while the loser, namely the competitor, is denoted as $\widetilde{\mathbf{x}}_{i,G+1}$. $\mathrm{x}_{n(j,i),G}$ denotes the jth neighbor of the particle i at generation G. Each particle has H neighbors. $\mathbf{gbest}_G$, which is used in Label (1) may result in fast convergence. However, it may also cause premature convergence due to the reduced population diversity. Therefore, $\mathbf{x}^p_{best,G}$, which is randomly chosen as one of the top $100p$% particles in the current population, is used instead of $\mathbf{gbest}_G$. $mean(\bullet)$ denotes the arithmetic mean value function.

After neighborhood exploration, a binomial crossover operation is employed to enhance the diversity of the population. $\mathbf{v}_{i,G+1} = (v^1_{i,G+1}, v^2_{i,G+1}, \cdots, v^D_{i,G+1})$ is updated as follows:

$$v^j_{i,G+1} = \begin{cases} v^j_{i,G+1} \; if \; rand \leq CR \\ v^j_{i,G} \; otherwise \end{cases} \quad j = 1, 2, \cdots, D. \tag{5}$$

The crossover factor CR is calculated as follows:

$$CR = \ln(c_1)(1 + \frac{rand}{2}), \tag{6}$$

where D is the dimension. $rand$ denotes a uniformly selected random number from [0, 1]. $ln(\cdot)$ denotes natural logarithm function. c_1 is the acceleration coefficient.

3.3. Property of Stagnation

It is called stagnation when the algorithm cannot find any better solutions. The property of stagnation can be shown in PSONHM by the $t_{i,G}$ at generation G, which is evaluated to estimate whether the algorithm cannot generate any successful solutions. The $t_{i,G}$ is updated as follows:

$$t_{i,G+1} = \begin{cases} 0 \; if \; f(\mathbf{x}_{i,G}) \leq f(\mathbf{x}_{i,G-1}) \\ t_{i,G} + 1 \; otherwise \end{cases} \quad i = 1, 2, \cdots, NP, \tag{7}$$

where f(*) is the objective function. NP is the population size.

The initial values $t_{i,1}$ are set to zero. It indicates that the algorithm cannot generate any successful solutions for the ith particle if $t_{i,G}$ increases continually. In this moment, it is thought that stagnation happens to the algorithm. In PSONHM, the neighborhood exploration strategy is employed to increase the probabilities of generating successful solutions when $t_{i,G}$ exceeds a fixed threshold value, namely, T.

3.4. Inertia Weight Assignments Based on Historical Memory

The inertia weight is helpful to balance the local and global search during the evolutionary process [35]. Instead of solely depending on a linearly decreasing inertia weight, the historical memory M_w, which stores a set of inertia weight values that performed well in the past, is used to generate new inertia weight with a parameter adaptation mechanism. PSONHM keeps a historical memory with k entries for inertia weight w. At first, the value of historical memory M_w with k entries at the first generation is all initialized as follows. c_0 is set to 0.5. The index $q \in [1, k]$ determines the inertia weight w_q that is to update in the memory:

$$M_{w,q,G} = \sin\left(1 + Norm(\frac{c_0}{NP \times k}, \frac{c_0}{NP})\right) - \frac{c_0}{MaxFES} \; (G = 1), \tag{8}$$

where $Norm(\bullet)$ is Gaussian distribution.

In each generation, the inertia weight w_r used by each particle is generated with a random index r within the range [1, k]. The w_r used by successful particle is recorded in S_w. At the end of the generation, the memory is updated as follows:

$$M_{w,q,G+1} = \begin{cases} \sin\left(1 + Norm(\frac{c_0}{NP \times k}, \frac{c_0}{NP})\right) - \frac{c_0}{MaxFES}G - mean_w(S_w) \; S_w \neq \varnothing \\ \sin\left(1 + Norm(\frac{c_0}{NP \times k}, \frac{c_0}{NP})\right) - \frac{c_0}{MaxFES}G \; otherwise \end{cases}. \quad (9)$$

At first, q is initialized to 1. When a new inertia weight w is inserted into the memory, q is increased. If $q > k$, q is reset to 1. If the population fails to generate a promising particle, which is better than the parent, the memory is not updated. Otherwise, the qth inertia weight in the memory is updated. c_0 is set to 0.5. *MaxFES* is the maximum number of fitness evaluations. The weighted Lehmer mean $mean_w$ (S_w) is computed as follows, which is proposed in [43]:

$$mean_w(S_w) = \frac{\sum_{i=1}^{|S_w|} \omega_k S_w^2}{\sum_{i=1}^{|S_w|} \omega_k S_w}, \quad (10)$$

$$w_k = \frac{\Delta f_k}{\sum_{i=1}^{|S_w|} \Delta f_k}, \quad (11)$$

where $\Delta f_k = |f(x_{k,G}) - f(x_{k,G\text{-}1})|$ is the amount of fitness improvement, which is used to influence the parameter adaptation.

The pseudo-code of PSONHM is illustrated in Algorithm 1. *NP* is the population size. *H*, which is the number of neighbors, is set to 5. k is the number of inertia weight values in the memory and is set to 5. T, which is the stagnation tolerance value, is set to 3. *CR* is the crossover factor.

Algorithm 1. PSONHM Algorithm.

```
Initialize D(number of dimensions), NP, H, k, T, c0, c1 and c2
Initialize population randomly
Initialize position x_i, velocity v_i, personal best position pbest_i, competitor of pbest_i and global best position gbest of the NP particles (i = 1, 2, ... , NP)
Initialize M_{w,q} according to Equation (8)
Index counter q = 1
while the termination criteria are not met do
  S_w = φ
  for i = 1 to NP do
    r = Select from [1, k] randomly
    w = M_{w,r}
    if t_i > = T
      Compute velocity v_i with neighborhood strategy according to Equation (4)
      Update velocity v_i by crossover operation according to Equations (5) and (6)
    else
      Compute velocity v_i according to Equation (1)
    end if
      Update position x_i according to Equation (2)
      Calculate objective function value f(x_i)
      Calculate t_i for next generation according to Equation (7)
  end for
  Update pbest_i, gbest, and the competitor of pbest_i (i = 1, 2, ... , NP)
  Update M_{w,q} based on S_w according to Equation (9)
  q = q + 1
  if q > k, q is set to 1
end while
Output: the particle with the smallest objective function value in the population.
```

4. Experiments and Discussion

In this section, CEC2014 contest benchmark problems, which are widely adopted in numerical optimizaiton methods, are used to verify the performance of the PSONHM algorithm. 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

(1) *Test Problems and Dimension Setting*: To verify the performance of PSONHM, *CEC2014* [44] contest benchmark problems are used. According to their diverse characteristics, these test problems can be divided into four kinds of optimization problems [44]:

- unimodal problems f_1–f_3,
- simple multimodal problems f_4–f_{16},
- hybrid problems f_{17}–f_{22}, and
- composite problems f_{23}–f_{30}.

The search space is $[-100, 100]^D$ for the optimization problems. D denotes the dimension and is set to 30 in this paper.

(2) *Experimental Platform and Termination Criterion*: All the experiments are run on a PC with a Celeron 3.40 GHz CPU (City, US State abbrev. if applicable, Country) and 4 GB memory. Each problem is executed for 30 runs with the maximal number of function evaluations (*FES*) $D \times 10{,}000$.

(3) *Performance Metrics*: The metrics, such as F_{mean} (mean value), *SD* (standard deviation), *Max* (maximum value) and *Min* (minimum value) of the *solution error measure* [45], 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 error will be recorded as 0 when the value $f(x) - f(x^*)$ is less than 10^{-8}. In view of statistics, the Wilcoxon signed-rank test [46] at the 5% significance level is used to compare PSONHM with other compared algorithms. "$\approx$", "+" and "$-$" are applied to express the performance of PSONHM is similar to, worse than, and better than that of the compared algorithm, respectively.

(4) *Control parameters*: PSONHM is compared with PSO, PSOcf, TLBO (Teaching-Learning-Based Optimization) [47], Jaya [48], GSA (Gravitational Search Algorithm) [49], BBO, CoDE (Differential evolution with composite trial vector generation strategies and control parameters) [46] and FPSO (Fuzzy Adaptive Particle Swarm Optimization) [50]. Default parameters settings for these algorithms are given in Table 1.

For most intelligent algorithms, the size of the population plays a significant role in controlling the convergence rate. Small population sizes may result in faster convergence, but increases the risk of premature convergence. On the contrary, large population sizes tend to explore widely, but reduce the rate of convergence. There are many studies on the population size of the optimization algorithms. Different population size of the same algorithm may result in different performance. Therefore, without loss of generality, the settings used for the competing algorithms are selected on the basis of the original papers. For PSONHM, we make experiments to study the influence of different population size N. The experimental results show that both smaller population size and larger population size are not the best choice for PSONHM. Therefore, the population size of PSONHM is recommended to set as the value 100, which is based on the result of the experiments.

Table 1. Default parameters settings.

Algorithm	Parameter	Value
PSO	*Population size* (N)	40
	Cognitive constant (C_1)	1.49445
	Social Constant (C_2)	1.49445
	Inertia constant (ω)	0.9 to 0.4
	Population size (N)	100
PSOcf	*Cognitive constant* (C_1)	1.49445
	Social Constant (C_2)	1.49445
	Inertia constant (ω)	0.729
TLBO	*Population size* (N)	100
FPSO	*Population size* (N)	80
	Cognitive constant (C_1)	2
	Social Constant (C_2)	2
Jaya	*Population size* (N)	100
GSA	*Population size* (N)	50
	Gravitational constant (G_0)	1
	α	20
BBO	*Population size* (N)	50
	Mutation Probability	0.08
	Number of elites each generation	8
CoDE	*Population size* (N)	100
	Mutation factor (F)	[1.0 1.0 0.8]
	Crossover factor (CR)	[0.1 0.9 0.2]
PSONHM	*Population size* (N)	100
	Cognitive constant (C_1)	1.49445
	Social Constant (C_2)	1.49445
	Memory size	5
	p	0.05

4.2. Comparison with Nine Optimization Algorithms on 30 Dimensions

The statistical results, in terms of F_{mean}, SD, Max and Min obtained in 30 independent runs by each algorithm, are reported in Table 2.

Table 2. Experimental results of PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE, FPSO and PSONHM on f_1–f_3.

F		PSO	PSOcf	TLBO	Jaya	GSA	BBO	CoDE	FPSO	PSONHM
f_1	F_{mean}	8.34×10^{6}	6.44×10^{7}	4.79×10^{5}	7.05×10^{7}	1.32×10^{7}	1.89×10^{7}	2.38×10^{4}	1.14×10^{7}	4.47×10^{5}
	SD	8.22×10^{6}	7.75×10^{7}	3.95×10^{5}	2.00×10^{7}	1.78×10^{6}	1.33×10^{7}	1.85×10^{4}	1.14×10^{7}	2.90×10^{5}
	Max	2.77×10^{7}	3.02×10^{8}	1.58×10^{6}	1.05×10^{8}	1.79×10^{7}	5.44×10^{7}	8.61×10^{4}	6.40×10^{7}	9.92×10^{5}
	Min	8.93×10^{4}	3.89×10^{5}	5.71×10^{4}	3.58×10^{7}	1.02×10^{7}	1.71×10^{6}	4840	1.97×10^{6}	8.93×10^{4}
Compare/rank		$-/4$	$-/8$	$\approx/2$	$-/9$	$-/6$	$-/7$	$+/1$	$-/5$	$\backslash/2$
f_2	F_{mean}	0.172	6.55×10^{9}	22.2	7.05×10^{9}	3.40×10^{9}	4.26×10^{6}	4.88	0.183	9.01×10^{-4}
	SD	0.529	5.33×10^{9}	15.6	9.74×10^{8}	1.86×10^{10}	1.64×10^{6}	2.10	0.664	1.35×10^{-3}
	Max	2.56	1.95×10^{10}	49.1	9.51×10^{9}	1.02×10^{11}	1.01×10^{7}	11.1	3.56	5.98×10^{-3}
	Min	2.78×10^{-6}	6.41×10^{-3}	4.73×10^{-3}	5.31×10^{9}	2.16×10^{3}	1.59×10^{6}	2.48	4.16×10^{6}	5.70×10^{-8}
Compare/rank		$-/2$	$-/8$	$-/5$	$-/9$	$-/7$	$-/6$	$-/4$	$-/3$	$\backslash/1$
f_3	F_{mean}	6.04	2.44×10^{3}	568	7.20×10^{4}	8.29×10^{4}	1.03×10^{4}	1.63×10^{-4}	16.21	0.370
	SD	10.1	3.97×10^{3}	358	1.42×10^{4}	1.52×10^{3}	7.91×10^{3}	9.08×10^{-5}	21.79	0.329
	Max	38.7	1.42×10^{4}	1720	1.03×10^{5}	8.52×10^{4}	3.01×10^{4}	4.15×10^{-4}	89.75	0.968
	Min	3.67×10^{-3}	8.23×10^{-2}	39.8	4.78×10^{4}	7.93×10^{4}	1.66×10^{3}	5.56×10^{-5}	5.80×10^{-2}	7.75×10^{-3}
Compare/rank		$-/3$	$-/4$	$-/3$	$-/6$	$-/7$	$-/5$	$+/1$	$-/4$	$\backslash/2$
$-/\approx/+$		3/0/0	3/0/0	2/1/0	3/0/0	3/0/0	3/0/0	1/0/2	3/0/0	$\backslash$
Avg-rank		3.00	6.67	3.33	8.00	6.67	6.00	2.00	4.00	1.67

(1) *Unimodal problems f_1–f_3*

From the statistical results of Table 2, we can see that PSONHM performs well on f_1–f_3 for 30 dimensions. For f_1–f_3, PSONHM works better than PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE

and FPSO on 3, 3, 2, 3, 3, 3, 1, 3 test problems, respectively. The overall ranking sequences for the test problems are PSONHM, CoDE, PSO, TLBO, FPSO, BBO, PSOcf (GSA) and Jaya in a descending direction. The average rank of PSOcf is the same as that of GSA. For unimodal problems f_1–f_3, the results indicate that the inertia weight, which is updated adaptively, is helpful for PSONHM to find the area where the potential optimal solution existed and converge to the optimal solution quickly.

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

Considering the simple multimodal problems f_4–f_{16} in Table 3, PSONHM outperforms other algorithms on $f_5, f_6, f_7, f_9, f_{12}, f_{13}$ and f_{15}. CoDE performs well on f_4. BBO performs well on f_8, f_{10}, f_{11} and f_{16}. TLBO performs well on and f_{14}. PSONHM performs better PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE and FPSO on 12, 12, 11, 13, 13, 6, 11 and 12 test problems, respectively. The overall ranking sequences for the test problems are PSONHM, BBO, PSO, CoDE (FPSO), TLBO, PSOcf, Jaya and GSA in a descending direction. The average rank of CoDE is the same as that of FPSO. The results indicate that PSONHM generally offered better performance in most of the simple multimodal problems, though it worked slightly worse on several problems. Due to the neighborhood mechanism, the population makes full use of the information from its neighbors and the competitor, and guides the evolution process successfully toward more promising solutions.

Table 3. Experimental results of PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE, FPSO and PSONHM on f_4–f_{16}.

F		PSO	PSOcf	TLBO	Jaya	GSA	BBO	CoDE	FPSO	PSONHM
f_4	f_{mean}	167	332	54.9	443	1800	119	19.5	157	107
	SD	26.2	238	41.2	88.9	6040	30.1	22.3	60.7	31.6
	Max	238	920	137	744	2.53×10^4	180	70.3	270	145
	Min	124	68.5	1.56×10^{-2}	330	166	72.3	1.46	9.01	31.8
Compare/rank		−/6	−/7	+/2	−/8	−/9	≈/3	+/1	−/5	\/3
f_5	f_{mean}	20.7	20.2	20.9	20.9	20.9	20.1	20.6	20.8	20
	SD	0.149	0.25	7.02×10^{-2}	4.75×10^{-2}	4.29×10^{-2}	4.17×10^{-2}	4.13×10^{-2}	6.40×10^{-2}	0.204
	Max	20.9	20.8	21	21	21	20.2	20.6	20.9	20.8
	Min	20.4	20	20.6	20.8	20.8	20.1	20.5	20.7	20
Compare/rank		−/5	−/3	−/8	−/7	−/9	−/2	−/4	−/6	\/1
f_6	f_{mean}	13.1	17.8	11.5	35.1	34.5	14	20.4	14.7	9.19
	SD	2.78	4.52	2.62	1.80	4.90	1.78	2.88	3.64	20.1
	Max	20	28.1	16.1	38.1	41.2	17.2	25.4	22.7	11.4
	Min	7.78	8.58	5.77	30.6	22.5	10.5	9.50	7.53	3.63
Compare/rank		−/3	−/6	−/2	−/9	−/8	−/4	−/7	−/5	\/1
f_7	f_{mean}	1.48×10^{-2}	79.7	1.34×10^{-2}	21	159	1.03	6.77×10^{-5}	1.06×10^{-2}	0
	SD	1.36×10^{-2}	43.9	1.99×10^{-2}	4.81	261	1.93×10^{-2}	5.44×10^{-5}	1.19×10^{-2}	0
	Max	6.64×10^{-2}	205	7.57×10^{-2}	31.8	1050	1.07	3.05×10^{-4}	4.40×10^{-2}	0
	Min	0	24.3	0	13.7	11.6	0.981	1.22×10^{-5}	0	0
Compare/rank		−/5	−/8	−/4	−/7	−/9	−/6	−/2	−/3	\/1
f_8	f_{mean}	29.4	75.5	58.5	224	179	0.609	18.5	38.2	15
	SD	7.01	23.1	11.7	12.8	52.8	0.244	1.93	13.6	3.15
	Max	44.7	131	81.5	254	448	1.39	22.2	80.5	18.9
	Min	16.9	31	39.7	204	140	0.204	13.8	15.9	5.96
Compare/rank		−/4	−/7	−/6	−/9	−/8	+/1	−/3	−/5	\/2
f_9	f_{mean}	77.1	123	61.3	262	214	51.1	139	82.9	50.5
	SD	14.4	36.1	14.9	13.9	65.9	10.3	9.41	23.4	7.97
	Max	101	216	96.5	291	445	70.3	154	139	59.6
	Min	46.7	59.6	38.8	223	166	32.8	112	41.7	32.8
Compare/rank		−/4	−/6	−/3	−/9	−/8	≈/1	−/7	−/5	\/1
f_{10}	f_{mean}	886	2370	1200	5630	4050	3.43	762	1.05×10^3	522
	SD	328	679	526	379	287	1.24	129	382	146
	Max	1600	3650	2400	6220	4400	7.10	991	1.61×10^3	708
	Min	265	1330	6.87	4800	3290	1.13	535	279	139
Compare/rank		−/4	−/7	−/6	−/9	−/8	+/1	−/3	−/5	\/2
f_{11}	f_{mean}	2890	3440	6490	6880	4390	1810	4800	3.30×10^3	2250
	SD	661	760	352	367	140	250	208	1.05×10^3	304
	Max	4300	4850	7150	7480	4620	2420	5230	6.85×10^3	2630
	Min	1440	1740	5510	5990	4050	1180	4290	1.89×10^3	1370

Table 3. *Cont.*

F		PSO	PSOcf	TLBO	Jaya	GSA	BBO	CoDE	FPSO	PSONHM
Compare/rank		−/3	−/5	−/8	−/9	−/6	+/1	−/7	−/4	\/2
f_{12}	f_{mean}	0.59	0.253	2.46	2.41	2.82	0.214	1.02	0.8201	0.198
	SD	0.268	9.85×10^{-2}	0.241	0.331	0.352	5.40×10^{-2}	0.110	0.633	4.75×10^{-2}
	Max	1.67	0.468	2.98	2.98	3.37	0.334	1.23	2.56	0.251
	Min	0.273	0.103	2.07	1.66	1.98	0.127	0.794	0.139	6.94×10^{-2}
Compare/rank		−/4	≈/1	−/8	−/7	−/9	≈/1	−/6	−/5	\/1
f_{13}	f_{mean}	0.395	1.53	0.418	1.59	8.81	0.513	0.464	0.4214	0.339
	SD	0.103	1.09	9.97×10^{-2}	0.323	0.938	0.103	6.44×10^{-2}	0.105	8.47×10^{-2}
	Max	0.594	4.33	0.619	2.48	10.2	0.691	0.546	0.701	0.555
	Min	0.186	0.556	0.262	0.982	6.74	0.264	0.325	0.277	0.197
Compare/rank		−/2	−/7	−/3	−/8	−/9	−/6	−/5	−/4	\/1
f_{14}	f_{mean}	0.308	22.1	0.275	9.61	139	0.402	0.284	0.313	0.361
	SD	0.124	16.5	5.30×10^{-2}	1.83	115	0.177	3.46×10^{-2}	0.127	0.181
	Max	0.842	77.1	0.391	13.1	403	0.982	0.363	0.820	0.714
	Min	0.189	0.89	0.151	4.96	15.7	0.23	0.201	0.180	0.199
Compare/rank		≈/1	−/8	≈/1	−/7	−/9	−/6	≈/1	≈/1	\/1
f_{15}	f_{mean}	7.43	4290	9.28	56.5	43.6	14.1	13.4	7.34	5.84
	SD	2.43	1.04×10^4	3.73	49.1	14.5	3.01	0.865	2.34	2.01
	Max	15.4	4.23×10^4	17.2	278	94.5	21.6	15.1	10.81	10.9
	Min	4.16	3.64	3.14	34.2	26.3	10.2	11.9	2.62	2.95
Compare/rank		−/3	−/9	−/4	−/8	−/7	−/6	−/5	−/2	\/1
f_{16}	f_{mean}	10.9	11.1	11.8	12.9	13.7	9.72	11.6	11.67	10.7
	SD	0.599	0.62	0.311	0.173	0.238	0.681	0.230	0.515	0.619
	Max	12	12.5	12.5	13.3	14.1	11.3	11.9	12.42	11.9
	Min	9.69	9.55	11.2	12.6	13.2	8.81	11.1	10.08	9.81
Compare/rank		−/3	−/4	−/7	−/8	−/9	≈/1	−/5	−/6	\/1
−/≈/+		12/1/0	12/1/0	11/1/1	13/0/0	13/0/0	6/4/3	11/1/1	12/1/0	\
Avg-rank		3.62	6.00	4.77	8.08	8.31	3.00	4.31	4.31	1.38

(3) *Hybrid problems* f_{17}–f_{22}

The results in Table 4 show that PSONHM performs better than other compared algorithms except CoDE. The overall ranking sequences for the test problems are CoDE, PSONHM, TLBO, PSO, FPSO, BBO, PSOcf, GSA and Jaya in a descending direction. Because of the crossover operation, which is utilized to enhance the potential diversity of the population, PSONHM can avoid premature convergence with a higher probability and show better performance than most compared algorithms on these hybrid problems.

Table 4. Experimental results of PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE, FPSO and PSONHM on f_{17}–f_{22}.

F		PSO	PSOcf	TLBO	Jaya	GSA	BBO	CoDE	FPSO	PSONHM
f_{17}	F_{mean}	7.41×10^5	1.17×10^6	2.10×10^5	4.41×10^6	1.65×10^6	1.64×10^6	1.47×10^3	7.88×10^5	1.07×10^5
	SD	7.71×10^5	1.46×10^6	1.66×10^5	1.62×10^6	1.60×10^5	1.02×10^6	235	9.02×10^5	8.24×10^4
	Max	3.05×10^6	5.69×10^6	7.79×10^5	8.20×10^6	1.94×10^6	5.10×10^6	1.87×10^3	4.51×10^6	2.93×10^5
	Min	1.92×10^4	3.45×10^4	4.36×10^4	8.17×10^5	1.26×10^6	3.24×10^5	831	7.62×10^4	4010
Compare/rank		−/4	−/6	−/3	−/9	−/8	−/7	+/1	−/5	\/2
f_{18}	F_{mean}	5.77×10^3	5.09×10^7	2480	2.66×10^7	286	3010	49.1	2.81×10^5	1310
	SD	6.10×10^3	1.39×10^8	4530	4.32×10^7	65	2570	6.05	1.42×10^6	1020
	Max	2.75×10^4	5.03×10^8	2.25×10^4	1.71×10^8	556	1.13×10^4	60.3	7.80×10^6	3020
	Min	251	437	77.8	5.03×10^6	230	289	36.2	248	136
Compare/rank		−/6	−/9	−/4	−/8	+/2	−/5	+/1	−/7	\/3
f_{19}	F_{mean}	15.1	25.9	12	37.1	217	29.7	7.15	15.6	6.88
	SD	20.4	28	11	23.8	138	33.1	0.689	21.3	0.922
	Max	76.9	140	69.4	120	753	115	8.43	87.9	7.99
	Min	4.52	8.53	4.78	23	43.3	6.97	5.86	4.38	4.82
Compare/rank		−/4	−/6	−/3	−/8	−/9	−/7	−/2	−/5	\/1
f_{20}	F_{mean}	368	1740	814	1.05×10^4	2.31×10^5	8020	30.4	537	257
	SD	229	1800	388	3.62×10^3	4.48×10^4	6190	4.04	311	57.9
	Max	1330	7440	2020	2.06×10^4	3.21×10^5	2.80×10^4	39.2	1.46×10^3	340
	Min	890	223	381	3.58×10^3	1.30×10^5	648	24.6	189	152
Compare/rank		−/3	−/6	−/5	−/8	−/9	−/7	+/1	−/4	\/2
f_{21}	F_{mean}	6.60×10^4	4.41×10^5	6.78×10^4	9.02×10^5	9.77×10^5	7.51×10^5	772	1.44×10^5	2.20×10^4
	SD	7.53×10^4	5.04×10^5	3.98×10^4	3.83×10^5	2.02×10^5	6.24×10^5	112	1.53×10^5	1.08×10^4
	Max	3.38×10^5	1.91×10^6	1.60×10^5	1.54×10^6	1.55×10^6	2.50×10^6	982	6.74×10^5	4.13×10^4
	Min	1720	1.12×10^4	1.92×10^4	3.95×10^5	6.47×10^5	3.05×10^4	583	4.66×10^3	6200

Table 4. *Cont.*

F		PSO	PSOcf	TLBO	Jaya	GSA	BBO	CoDE	FPSO	PSONHM
Compare/rank		−/3	−/6	−/4	−/8	−/9	−/7	+/1	−/5	\/2
f_{22}	F_{mean}	310	600	239	628	922	478	271	347	232
	SD	136	218	106	113	161	200	153	173	82.4
	Max	620	1060	415	822	1270	896	627	777	330
	Min	22.5	204	40.2	349	736	35.2	25.8	20.8	411
Compare/rank		−/4	−/7	≈/1	−/8	−/9	−/6	−/3	−/5	\/1
−/≈/+		6/0/0	6/0/0	5/1/0	6/0/0	5/0/1	6/0/0	2/0/4	6/0/0	\
Avg-rank		4.00	6.67	3.33	8.17	7.67	6.50	1.50	5.17	1.83

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

The composite problems are very complex and time-consuming because they combine multiple test problems into a complex landscape. Thus, it is difficult for optimization algorithms to achieve better solutions. Table 5 shows that the overall ranking sequences for the test problems are GSA, PSONHM, CoDE (TLBO), BBO, PSO, FPSO, Jaya and PSOcf in a descending direction. The average rank of CoDE is the same as that of TLBO. The results indicate that the neighborhood mechanism can not effectively solve the composite problems.

Table 5. Experimental results of PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE, FPSO and PSONHM on f_{23}–f_{30}.

F		PSO	PSOcf	TLBO	Jaya	GSA	BBO	CoDE	FPSO	PSONHM
f_{23}	F_{mean}	315	353	315	349	246	316	315	315	315
	SD	0.214	22	1.71	6.11	13.7	0.854	7.14×10^{-7}	0.203	0.195
	Max	316	416	315	364	269	318	315	316	316
	Min	315	325	315	338	220	315	315	315	315
Compare/rank		≈/3	−/9	+/2	−/8	+/1	−/7	≈/3	−/6	\/3
f_{24}	F_{mean}	235	257	200	252	207	233	249	236	230
	SD	8.15	25.8	1.55×10^{-3}	12.5	0.327	4.61	16.8	6.41	5.60
	Max	250	331	200	266	208	246	297	247	243
	Min	224	226	200	212	207	228	225	223	224
Compare/rank		−/5	−/9	+/1	−/8	+/2	−/4	−/7	−/6	\/3
f_{25}	F_{mean}	210	214	200	220	201	207	202	212	210
	SD	3.02	9.42	0.621	4.91	4.30×10^{-2}	1.58	0.139	4.15	2.55
	Max	218	241	203	229	201	210	203	221	216
	Min	206	204	200	210	200	205	202	206	206
Compare/rank		−/6	−/8	+/1	−/9	+/2	+/4	+/3	−/7	\/5
f_{26}	F_{mean}	128	115	107	101	171	100	100	103	100
	SD	55.9	33.7	25.2	0.411	37.7	0.114	0.529	18.3	0.411
	Max	332	200	200	103	200	100	100	200	100
	Min	100	100	100	100	108	100	100	100	100
Compare/rank		−/8	−/7	−/6	−/4	−/9	−/2	−/3	−/5	\/1
f_{27}	F_{mean}	636	798	512	1130	210	570	400	622	427
	SD	148	236	138	87.4	1.31	124	2.24	159	38.8
	Max	932	1090	844	1210	213	722	401	853	523
	Min	401	432	401	722	206	405	400	401	401
Compare/rank		−/7	−/8	−/4	−/9	+/1	−/5	≈/2	−/6	\/2
f_{28}	F_{mean}	1234	1570	1080	1208	213	977	1035	1.42×10^{3}	985
	SD	378	324	175	205	3.11	160	126	448	42.9
	Max	2400	2330	1700	1960	221	1630	1225	2.46×10^{3}	1040
	Min	906	1130	887	1050	208	803	890	918	897
Compare/rank		−/7	−/9	−/5	−/6	+/1	≈/2	−/4	−/8	\/2
f_{29}	F_{mean}	2.14×10^{6}	6.22×10^{6}	1.44×10^{6}	2.12×10^{6}	244	1830	564	1.29×10^{6}	1140
	SD	6.71×10^{6}	5.50×10^{6}	3.28×10^{6}	3.56×10^{6}	8.55	504	206	4.91×10^{6}	136
	Max	2.56×10^{7}	1.71×10^{7}	9.16×10^{6}	1.02×10^{7}	258	2850	733	1.96×10^{7}	1290
	Min	1010	5.15×10^{4}	1130	6.24×10^{4}	229	1150	261	779	828
Compare/rank		−/8	−/9	−/6	−/7	+/1	−/4	+/2	−/5	\/3
f_{30}	F_{mean}	4660	9.51×10^{4}	3870	1.72×10^{4}	251	6170	1.11×10^{3}	9.50×10^{3}	3040
	SD	2260	7.58×10^{4}	2870	1.46×10^{4}	7.86	2670	179	9.06×10^{3}	836
	Max	1.06×10^{4}	2.35×10^{5}	1.37×10^{4}	6.91×10^{4}	266	1.17×10^{4}	1.45×10^{3}	4.42×10^{4}	4090
	Min	966	1860	994	7420	235	2090	772	1.20×10^{3}	1440
Compare/rank		−/5	−/9	≈/3	−/8	+/1	−/6	+/2	−/7	\/3
−/≈/+		7/1/0	8/0/0	4/1/3	8/0/0	1/0/7	6/1/1	3/2/3	8/0/0	\
Avg-rank		6.13	8.50	3.50	7.38	2.25	4.25	3.25	6.25	2.75

All in all, PSONHM performs better than the compared algorithms when considering f_1–f_{30} on 30 dimensions. Table 6 indicates PSONHM is competitive on CEC2014 test problems. PSONHM outperforms other algorithms on $f_2, f_5, f_6, f_7, f_9, f_{12}, f_{13}, f_{15}, f_{19}, f_{22}$ and f_{26}. CoDE performs well on $f_1, f_3, f_4, f_{17}, f_{18}, f_{20}$ and f_{21}. BBO performs well on f_8, f_{10}, f_{11} and f_{16}. TLBO performs well on and f_{14}, f_{24} and f_{25}. GSA outperforms other algorithms on $f_{23}, f_{27}, f_{28}, f_{29}$ and f_{30}. The total ranking orders on 30 test problems are PSONHM, CoDE, TLBO, PSO, BBO, FPSO, GSA, PSOcf and Jaya in a descending direction. Figure 3 shows the convergence curves for sixteen of the 30-dimensional CEC2014 benchmark problems. The curves illustrate mean errors (in logarithmic scale) in 30 independent simulations of PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE, FPSO and PSONHM. As mentioned above, the curves indicate that, in most problems, PSONHM either achieves a fast convergence or behaves similarly to it.

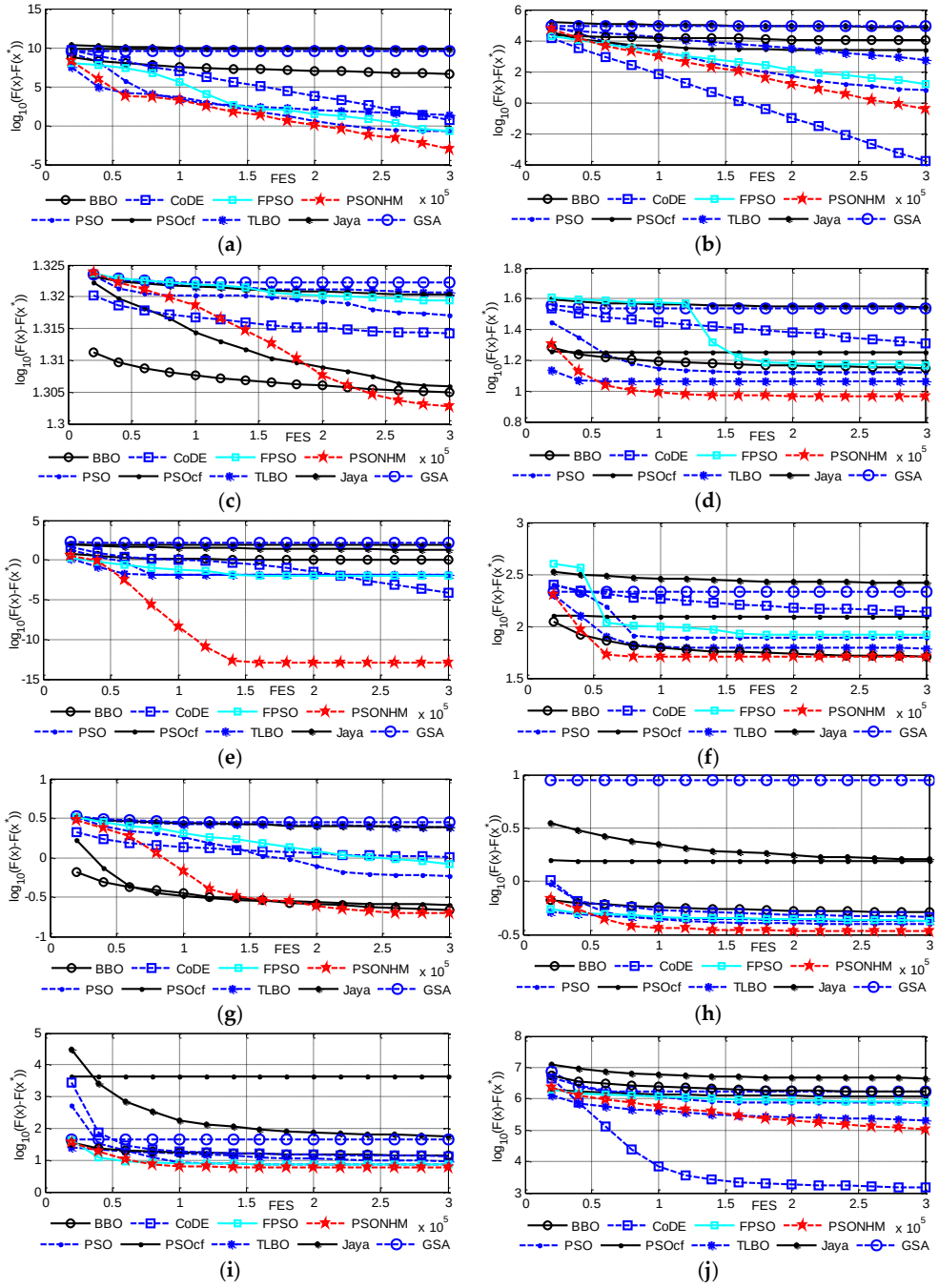

Figure 3. *Cont.*

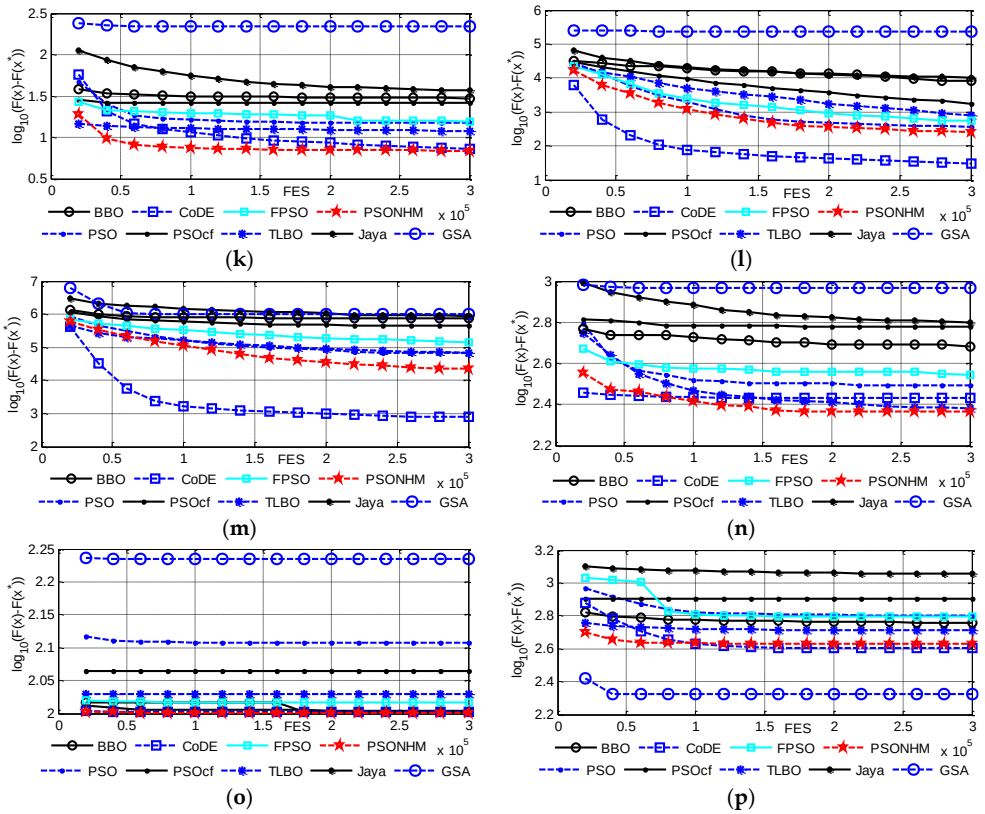

Figure 3. Evolution of the mean function error values derived from PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE, FPSO and PSONHM versus the number of *FES* on sixteen test problems with $D = 30$. (**a**) f_2; (**b**) f_3; (**c**) f_5; (**d**) f_6; (**e**) f_7; (**f**) f_9; (**g**) f_{12}; (**h**) f_{13}; (**i**) f_{15}; (**j**) f_{17}; (**k**) f_{19}; (**l**) f_{20}; (**m**) f_{21}; (**n**) f_{22}; (**o**) f_{26}; (**p**) f_{27}.

Table 6. Comparison of PSONHM with PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE and FPSO on the CEC2014 benchmarks (D = 30 dimensions).

	D	PSO	PSOcf	TLBO	Jaya	GSA	BBO	CoDE	FPSO	PSONHM
30	−/≈/+	28/2/0	29/1/0	22/4/4	30/0/0	22/0/8	21/4/5	17/3/10	29/1/0	\
	Avg-rank	4.30	6.87	4.00	7.90	6.40	4.33	3.23	4.97	1.87

The experiment results reveal that PSONHM can work well for most test problems because of the use of neighborhood mechanism and the adaptive inertia weight assignments based on historical memory. With the interaction of the best particles, the neighborhood particles and the competitors, the neighborhood exploration mechanism is designed to guide the search better in the next generation. It is helpful for PSONHM to explore and find the area where the potential optimal solution is existed. The risk of premature convergence is decreased as much as possible. After neighborhood exploration, PSONHM utilizes the crossover operation to enhance the potential diversity of the population. The convergence rate of algorithm is obviously improved because of learning from previous experience. In addition, the inertia weight is adaptively adjusted based on the historical memory, where the better inertia weight preserved in each generation. Then, the probability of finding better solutions is greater and this is helpful for improving the performance of the proposed algorithm. Thus, the exploration and exploitation are done during the optimization process. Accordingly, PSONHM can not only improve the convergence rate of algorithm but can also decrease the risk of premature convergence as much as possible.

5. PSONHM for Training an MLP

In this section, the proposed PSONHM algorithm is applied to solve two classification problems by training an MLP. The basic structure of the proposed scheme is depicted in Figure 4.

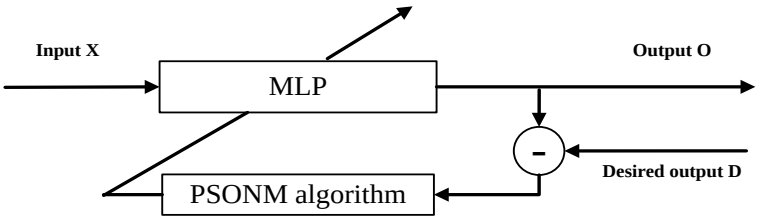

Figure 4. PSONHM-based MLP.

5.1. Multi-Layer Perceptron

In this section, the proposed algorithm is used for training MLPs. There are three layers in MLP: input layer, hidden layer and output layer. Figure 5 shows a multi-layer perceptron for neural network architecture. These layers are interconnected by links called weights. The outputs of hidden nodes (S_j, $j = 1, \ldots, H$) are calculated by an activation function, which is defined as follows:

$$S_j = \frac{1}{(1 + exp(-(\sum_{i=1}^{n}(W_{ij}X_i) - \theta_j)))}, \tag{12}$$

where n is the number of the input nodes, W_{ij} is the connection weight from the ith node in the input layer to the jth node in the hidden layer. X_i shows the ith input node. θ_j denotes the threshold.

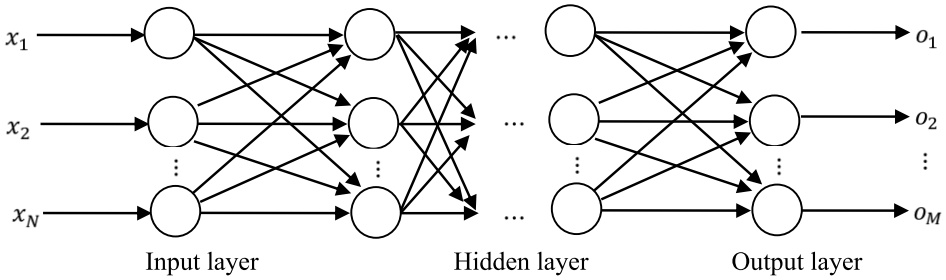

Figure 5. Multi-layer perceptron for neural network architecture.

After calculating the outputs of hidden nodes, the final outputs (O_k, $k = 1, \ldots, m$) are calculated by applying a sigmoid function:

$$O_k = \frac{1}{(1 + exp(-\left(\sum_{j=1}^{H}(W_{jk}S_j) - \theta_k\right)))}, \tag{13}$$

where θ_k denotes the threshold of the kth output node. W_{jk} is the connection weight from the jth node in the hidden layer to the kth node in the output layer.

The aim of training MLPs is to find a set of weights with the smallest error measure. The objective function is the mean sum of squared errors (MSE) over all training patterns, which is shown as follows:

$$MSE = \sum_{i=1}^{Q} \frac{\sum_{j=1}^{k}(d_{ij} - o_{ij})^2}{Q}, \tag{14}$$

where Q is the number of training data set, K is the number of output units, d_{ij} is desired output and o_{ij} is actual output of the ith input node.

Finally, the objective function uses MSE to evaluate the fitness of the individuals in each optimization algorithm. The fitness of the ith training sample is calculated by:

$$\text{Fitness } (X_i) = MSE\ (X_i). \tag{15}$$

5.2. Classification Problems

In this section, PSONHM is evaluated on the classification datasets Iris and Balloon, and the two datasets are obtained from the University of California at Irvine (UCI) Machine Learning Repository [51]. The metrics, such as MSE_{mean} (mean value), MSE_{std} (standard deviation), MSE_{max} (maximum value) MSE_{min} (minimum value) of MSE, and Classification rate are used to appraise the performance of each algorithm.

To provide a fair comparison, all algorithms were terminated when a maximum number of fitness evaluations ($MaxFES$ = 50,000) were reached. The mean trained MLP with 10 runs is chosen and used to classify the test set. The performance of the different algorithms for Iris problem and Balloon problem is presented in Tables 7 and 8.

5.2.1. Iris Flower Classification

The Iris flower classification is a three-class problem with 150 samples. Each sample consists of four features, namely sepal length, sepal width, petal length, and petal width. Iris flower classification was solved using MLPs with four inputs, nine hidden nodes and three output nodes. The experimental results are shown in Table 7 and Figure 6.

The MSE results show that PSONHM manages to solve the Iris flower classification with high accuracy compared PSO, PSOcf, TLBO, Jaya, GSA, BBO, CoDE and FPSO. The overall ranking sequences for MSE_{mean} are PSONHM, TLBO, PSO, BBO, CoDE, PSOcf, FPSO, Jaya and GSA in a descending direction. The overall ranking sequences for classification rate are PSONHM, TLBO, PSOcf, PSO, FPSO, BBO, Jaya, CoDE and GSA in a descending direction. The classification rate is 93.40% for PSONHM, more than all the other algorithms. Figure 6 shows the convergence curves of nine algorithms. It can be seen that the convergence rate of PSONHM is much faster than all other algorithms.

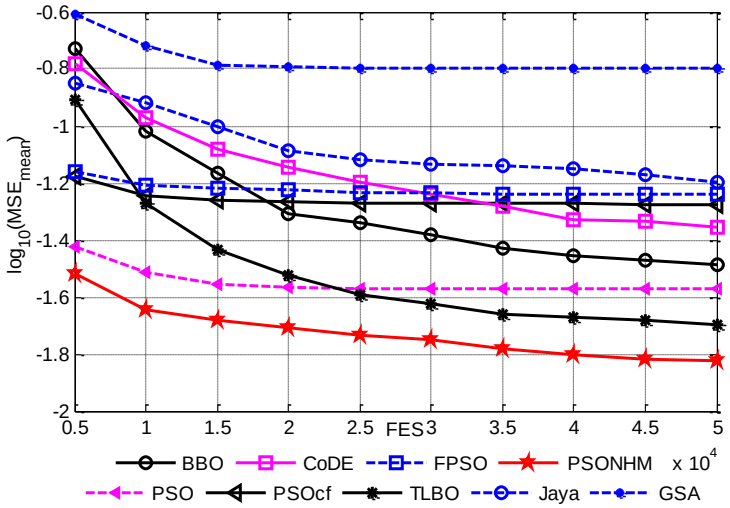

Figure 6. Convergence curves of nine algorithms for the Iris dataset.

Table 7. Experimental results for the Iris dataset.

Algorithm	MSE_{mean}	MSE_{std}	MSE_{max}	MSE_{min}	Classification Rate (%)
PSO	2.67×10^{-2}	1.92×10^{-3}	2.26×10^{-2}	2.97×10^{-2}	84.80
PSOcf	5.32×10^{-2}	1.00×10^{-1}	1.60×10^{-2}	3.40×10^{-1}	86.20
TLBO	2.01×10^{-2}	5.07×10^{-3}	1.45×10^{-2}	3.14×10^{-2}	90.80
Jaya	6.31×10^{-2}	1.36×10^{-2}	4.95×10^{-2}	9.21×10^{-2}	80.93
GSA	1.60×10^{-1}	2.45×10^{-2}	0.127	1.99×10^{-1}	0.00
BBO	3.26×10^{-2}	4.63×10^{-3}	2.63×10^{-2}	3.90×10^{-2}	83.00
CoDE	4.41×10^{-2}	5.82×10^{-3}	5.37×10^{-2}	3.48×10^{-2}	67.06
FPSO	5.75×10^{-2}	9.97×10^{-2}	3.41×10^{-1}	2.46×10^{-2}	84.73
PSONHM	1.49×10^{-2}	3.80×10^{-3}	7.11×10^{-3}	2.12×10^{-2}	93.40

5.2.2. Balloon Classification

The Balloon dataset has 16 samples that are divided into two classes: inflated or not. All samples have four attributes such as color, size, act, and age. This dataset is solved by employing MLP with four input nodes, nine hidden nodes, and one output node. The experimental results are shown in Table 8 and Figure 7.

The MSE results show that the classification rate is 100% for all the algorithms except GSA. The classification rate of GSA is 49.50%. As shown in Table 8, the overall ranking sequences for MSE_{mean} are PSONHM, TLBO, PSOcf, BBO, FPSO, PSO, Jaya, CoDE and GSA in a descending direction. PSONHM achieves the minimum error on the balloon classification problem. In addition, it can be seen that the convergence of PSONHM is faster than other compared algorithms.

Table 8. Experimental results for the Balloon dataset.

Algorithm	MSE_{mean}	MSE_{std}	MSE_{max}	MSE_{min}	Classification Rate (%)
PSO	8.34×10^{-12}	2.56×10^{-11}	5.67×10^{-20}	8.14×10^{-11}	100
PSOcf	7.07×10^{-19}	1.03×10^{-18}	2.94×10^{-25}	3.06×10^{-18}	100
TLBO	5.02×10^{-20}	1.58×10^{-19}	4.49×10^{-31}	5.01×10^{-19}	100
Jaya	1.43×10^{-11}	2.48×10^{-11}	3.10×10^{-15}	8.11×10^{-11}	100
GSA	1.41×10^{-2}	3.20×10^{-2}	4.85×10^{-5}	1.04×10^{-1}	49.50
BBO	2.99×10^{-15}	6.51×10^{-15}	1.02×10^{-20}	2.02×10^{-14}	100
CoDE	6.98×10^{-11}	7.39×10^{-11}	2.35×10^{-10}	7.84×10^{-14}	100
FPSO	1.45×10^{-13}	1.88×10^{-13}	5.46×10^{-13}	4.32×10^{-16}	100
PSONHM	9.27×10^{-26}	1.72×10^{-25}	1.05×10^{-33}	4.47×10^{-25}	100

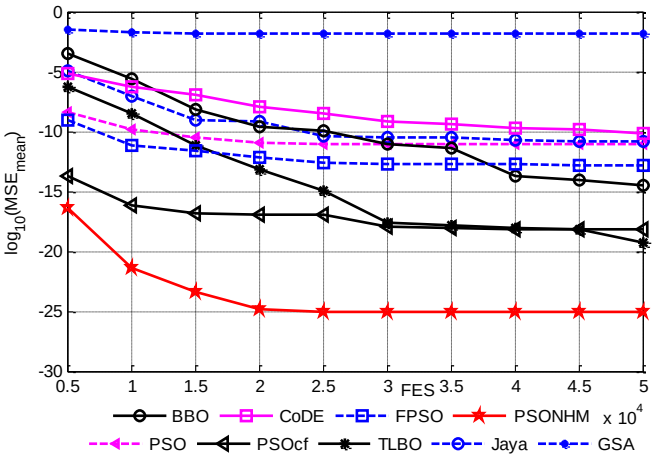

Figure 7. Convergence curves of nine algorithms for the Balloon dataset.

6. Conclusions

PSO shows better performance in exploitation but worse performance in exploration. It is difficult for the particles to jump out of the local optimal region because all the particles are attracted by the same global best particle, which limits the exploration ability of PSO. In this paper, we proposed an improving particle swarm optimization algorithm based on neighborhood and historical memory (PSONHM). In the proposed algorithm, the experience of its neighbors and its competitors is considered to decrease the risk of premature convergence as much as possible. Furthermore, the several best particles are selected instead of *gbest* to guide the swarm towards a new better space in the search process. Finally, the parameter adaptation mechanism is designed to generate new inertia weight. By comparing the experimental results with those from other algorithms on CEC2014 test problems and two benchmark datasets (Iris and Balloon), it can be concluded that the PSONHM algorithm significantly improves the performance of the original PSO algorithm.

In the future, the efficiency of PSONHM in training other types of ANN is worthy of study (e.g., Radia basis function networks, Kohonen networks, etc.). In addition, employing PSONHM to design an evolutionary neural network method is worth investigating. Finally, PSONHM will be expected to solve the multi-objective optimization problems and constrained optimization problems.

Acknowledgments: This research is supported by the support of the Doctoral Foundation of Xi'an University of Technology (112-451116017), and the National Natural Science Foundation of China under Project Code (61773314).

Conflicts of Interest: The author declares no conflict of interest.

References

1. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Let A Biogeography-Based Optimizer Train Your Multi-Layer Perceptron. *Inf. Sci.* **2014**, *268*, 188–209. [CrossRef]
2. Rosenblatt, F. *The Perceptron, a Perceiving and Recognizing Automaton Project Para*; Cornell Aeronautical Laboratory: Buffalo, NY, USA, 1957.
3. Werbos, P. Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences. Ph.D. Thesis, Harvard University, Cambridge, MA, USA, 1974.
4. Askarzadeh, A.; Rezazadeh, A. Artificial neural network training using a new efficient optimization algorithm. *Appl. Soft Comput.* **2013**, *13*, 1206–1213. [CrossRef]
5. Gori, M.; Tesi, A. On the problem of local minima in backpropagation. *IEEE Trans. Pattern Anal. Mach. Intell.* **1992**, *14*, 76–86. [CrossRef]
6. Mendes, R.; Cortez, P.; Rocha, M.; Neves, J. Particle swarms for feedforward neural network training. In Proceedings of the 2002 International Joint Conference on Neural Networks, Honolulu, HI, USA, 12–17 May 2002; pp. 1895–1899.
7. Demertzis, K.; Iliadis, L. Adaptive Elitist Differential Evolution Extreme Learning Machines on Big Data: Intelligent Recognition of Invasive Species. *Adv. Intell. Syst. Comput.* **2017**, *529*, 1–13.
8. Seiffert, U. Multiple layer perceptron training using genetic algorithms. In Proceedings of the European Symposium on Artificial Neural Networks, Bruges, Belgium, 25–27 April 2001; pp. 159–164.
9. Blum, C.; Socha, K. 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, 6–9 December 2005; pp. 233–238.
10. Tian, G.D.; Ren, Y.P.; Zhou, M.C. Dual-Objective Scheduling of Rescue Vehicles to Distinguish Forest Fires via Differential Evolution and Particle Swarm Optimization Combined Algorithm. *IEEE Trans. Intell. Transp. Syst.* **2016**, *99*, 1–13. [CrossRef]
11. Guedria, N.B. Improved accelerated PSO algorithm for mechanical engineering optimization problems. *Appl. Soft Comput.* **2016**, *40*, 455–467. [CrossRef]
12. Segura, C.; CoelloCoello, C.A.; Hernández-Díaz, A.G. Improving the vector generation strategy of Differential Evolution for large-scale optimization. *Inf. Sci.* **2015**, *323*, 106–129. [CrossRef]
13. Liu, B.; Zhang, Q.F.; Fernandez, F.V.; Gielen, G.G.E. An Efficient Evolutionary Algorithm for Chance-Constrained Bi-Objective Stochastic Optimization. *IEEE Trans. Evol. Comput.* **2013**, *17*, 786–796. [CrossRef]

14. Zaman, M.F.; Elsayed, S.M.; Ray, T.; Sarker, R.A. Evolutionary Algorithms for Dynamic Economic Dispatch Problems. *IEEE Trans. Power Syst.* **2016**, *31*, 1486–1495. [CrossRef]
15. CarrenoJara, E. Multi-Objective Optimization by Using Evolutionary Algorithms: The *p*-Optimality Criteria. *IEEE Trans. Evol. Comput.* **2014**, *18*, 167–179.
16. Cheng, S.H.; Chen, S.M.; Jian, W.S. Fuzzy time series forecasting based on fuzzy logical relationships and similarity measures. *Inf. Sci.* **2016**, *327*, 272–287. [CrossRef]
17. Das, S.; Abraham, A.; Konar, A. Automatic clustering using an improved differential evolution algorithm. *IEEE Trans. Syst. Man Cybern. Part A* **2008**, *38*, 218–236. [CrossRef]
18. Hansen, N.; Kern, S. Evaluating the CMA evolution strategy on multimodal test functions. In *Parallel Problem Solving from Nature (PPSN), Proceedings of the 8th International Conference, Birmingham, UK, 18–22 September 2004*; Springer International Publishing: Basel, Switzerland, 2004; pp. 282–291.
19. Kirkpatrick, S.; Gelatt, C.D., Jr.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef] [PubMed]
20. Černý, V. Thermo dynamical approach to the traveling salesman problem: An efficient simulation algorithm. *J. Optim. Theory Appl.* **1985**, *45*, 41–51. [CrossRef]
21. Simon, D. Biogeography-based optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
22. Lam, A.Y.S.; Li, V.O.K. Chemical-Reaction-Inspired Metaheuristic for Optimization. *IEEE Trans. Evol. Comput.* **2010**, *14*, 381–399. [CrossRef]
23. Shi, Y.H. *Brain Storm Optimization Algorithm*; Advances in Swarm Intelligence, Series Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6728, pp. 303–309.
24. Kennedy, J.; Eberhart, K. Particle swarm optimization. In Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, 27 November–1 December 1995; pp. 1942–1948.
25. Bergh, F.V.D. An Analysis of Particle Swarm Optimizers. Ph.D. Thesis, University of Pretoria, Pretoria, South Africa, 2002.
26. Clerc, M.; Kennedy, J. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evol. Comput.* **2002**, *6*, 58–73. [CrossRef]
27. Krzeszowski, T.; Wiktorowicz, K. Evaluation of selected fuzzy particle swarm optimization algorithms. In Proceedings of the Federated Conference on Computer Science and Information Systems (FedCSIS), Gdansk, Poland, 11–14 September 2016; pp. 571–575.
28. Alfi, A.; Fateh, M.M. Intelligent identification and control using improved fuzzy particle swarm optimization. *Expert Syst. Appl.* **2011**, *38*, 12312–12317. [CrossRef]
29. Kwolek, B.; Krzeszowski, T.; Gagalowicz, A.; Wojciechowski, K.; Josinski, H. Real-Time Multi-view Human Motion Tracking Using Particle Swarm Optimization with Resampling. In *Articulated Motion and Deformable Objects*; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7378, pp. 92–101.
30. Sharifi, A.; Harati, A.; Vahedian, A. Marker-based human pose tracking using adaptive annealed particle swarm optimization with search space partitioning. *Image Vis. Comput.* **2017**, *62*, 28–38. [CrossRef]
31. Gandomi, A.H.; Yun, G.J.; Yang, X.S.; Talatahari, S. Chaos-enhanced accelerated particle swarm optimization. *Commun. Nonlinear Sci. Numer. Simul.* **2013**, *18*, 327–340. [CrossRef]
32. Mendes, R.; Kennedy, J.; Neves, J. The fully informed particle swarm simpler, maybe better. *IEEE Trans. Evol. Comput.* **2004**, *8*, 204–210. [CrossRef]
33. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]
34. Nobile, M.S.; Cazzaniga, P.; Besozzi, D.; Colombo, R. Fuzzy self-turning PSO: A settings-free algorithm for global optimization. *Swarm Evol. Comput.* **2017**. [CrossRef]
35. Shi, Y.H.; Eberhart, R. A modified particle swarm optimizer. In Proceedings of the IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.
36. Shi, Y.H.; Eberhart, R.C. Empirical study of particle swarm optimizaiton. In Proceedings of the IEEE Congress on Evolutionary Computation, Washington, DC, USA, 6–9 July 1999; pp. 1950–1955.
37. Das, S.; Abraham, A.; Chakraborty, U.K.; Konar, A. Differential evolution using a neighborhood-based mutation operator. *IEEE Trans. Evol. Comput.* **2009**, *13*, 526–553. [CrossRef]
38. Omran, M.G.; Engelbrecht, A.P.; Salman, A. Bare bones differential evolution. *Eur. J. Oper. Res.* **2009**, *196*, 128–139. [CrossRef]

39. Suganthan, P.N. Particle swarm optimiser with neighbourhood operator. In Proceedings of the IEEE Congress on Evolutionary Computation, Washington, DC, USA, 6–9 July 1999; Volume 3, pp. 1958–1962.
40. Nasir, M.; Das, S.; Maity, D.; Sengupta, S.; Halder, U. A dynamic neighborhood learning based particle swarm optimizer for global numerical optimization. *Inf. Sci.* **2012**, *209*, 16–36. [CrossRef]
41. Ouyang, H.B.; Gao, L.Q.; Li, S.; Kong, X.Y. Improved global-best-guided particle swarm optimization with learning operation for global optimization problems. *Appl. Soft Comput.* **2017**, *52*, 987–1008. [CrossRef]
42. Zhang, J.Q.; Sanderson, A.C. JADE: Adaptive Differential Evolution with Optional External Archive. *IEEE Trans. Evol. Comput.* **2009**, *13*, 945–957. [CrossRef]
43. Tanabe, R.; Fukunaga, A. Success-history based parameter adaptation for differential evolution. In Proceedings of the IEEE Congress on Evolutionary Computation, Cancún, Mexico, 20–23 June 2013; pp. 71–78.
44. Liang, J.J.; Qu, B.Y.; Suganthan, P.N. *Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*; Technical Report; Zhengzhou University and Nanyang Technological University: Singapore, 2013.
45. Suganthan, P.N.; Hansen, N.; Liang, J.J.; Deb, K.; Chen, Y.P.; Auger, A.; Tiwari, S. Problem Definitions and Evaluation Criteria for the CEC2005 Special Session on Real-Parameter Optimization. 2005. Available online: http://www.ntu.edu.sg/home/EPNSugan (accessed on 9 January 2017).
46. Wang, Y.; Cai, Z.X.; Zhang, Q.F. Differential evolution with composite trial vector generation strategies and control parameters. *IEEE Trans. Evol. Comput.* **2011**, *15*, 55–66. [CrossRef]
47. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput. Aided Des.* **2011**, *43*, 303–315. [CrossRef]
48. Rao, R.V. Jaya: A simple and new optimization algorithm for solving constrained and unconstrained optimization problems. *Int. J. Ind. Eng. Comput.* **2016**, *7*, 19–34.
49. Rashedi, E.; Nezamabadi, S.; Saryazdi, S. GSA: A gravitational search algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
50. Shi, Y.H.; Eberhart, R.C. Fuzzy adaptive particle swarm optimization. In Proceedings of the Congress on Evolutionary Computation, Seoul, Korea, 27–30 May 2001; Volume 1, pp. 101–106.
51. Bache, K.; Lichman, M. UCI Machine Learning Repository. 2013. Available online: http://archive.ics.uci.edu/ml (accessed on 9 January 2017).