Research Article

A Novel Particle Swarm Optimization Algorithm for Global Optimization

Chun-Feng Wang^{1,2} and Kui Liu¹

¹Department of Mathematics, Henan Normal University, Xinxiang 453007, China

²Henan Engineering Laboratory for Big Data Statistical Analysis and Optimal Control, School of Mathematics and Information Sciences, Henan Normal University, Xinxiang 453007, China

Correspondence should be addressed to Chun-Feng Wang; wangchunfeng10@126.com

Received 17 August 2015; Revised 1 December 2015; Accepted 8 December 2015

Academic Editor: Massimo Panella

Copyright © 2016 C.-F. Wang and K. Liu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Particle Swarm Optimization (PSO) is a recently developed optimization method, which has attracted interest of researchers in various areas due to its simplicity and effectiveness, and many variants have been proposed. In this paper, a novel Particle Swarm Optimization algorithm is presented, in which the information of the best neighbor of each particle and the best particle of the entire population in the current iteration is considered. Meanwhile, to avoid premature, an abandoned mechanism is used. Furthermore, for improving the global convergence speed of our algorithm, a chaotic search is adopted in the best solution of the current iteration. To verify the performance of our algorithm, standard test functions have been employed. The experimental results show that the algorithm is much more robust and efficient than some existing Particle Swarm Optimization algorithms.

1. Introduction

This paper considers the following global optimization problem:

$$\begin{aligned} \min \quad & f(x) \\ \text{s.t.} \quad & x \in X, \end{aligned} \quad (1)$$

where x is a continuous variable vector with domain $X \subset R^n$ defined by the bound constraint $l_j \leq x_j \leq u_j$, $j = 1, \dots, n$. The function $f(x) : X \rightarrow R$ is a continuous real-valued function.

Many real-world problems, such as engineering and related areas, can be reduced to formulation (1). This problem usually has many local optima, so it is difficult to find its global optimum. For solving such problem, researchers have presented many methods during the past years, which can be divided into two groups: deterministic and stochastic algorithms. Most deterministic algorithms usually effective for unimodal functions have one global optimum and need gradient information. However, stochastic algorithms do not require any properties of the objective function. Therefore, more attention has been paid to stochastic algorithms

recently, and many effective algorithms have been presented, including Simulated Annealing (SA) [1], Genetic Algorithm (GA) [2, 3], Differential Evolution (DE) [4], Particle Swarm Optimization (PSO) [5], Ant Colony Optimization (ACO) [6], Artificial Bee Colony (ABC) [7], and Harmony Search (HS) [8].

Among these stochastic algorithms, PSO is a population-based and intelligent method, which is inspired by the emergent motion of a flock of birds searching for food [5]. In PSO, a population of potential solutions is evolved through successive iterations. Since PSO algorithm has a number of desirable properties, including simplicity of implementation, scalability in dimension, and good empirical performance, it has been applied to solve many real-world problems, such as capacitor placement problem [9], short term load forecasting [10], soft sensor [11], the voltage stability of the electric power distribution systems [12, 13], the orbits of discrete chaotic dynamical systems towards desired target region [14], and the permutation flow-shop sequencing problem [15].

Although PSO algorithm has been applied successfully in solving many difficult optimization problems, it also has difficulties in keeping balance between exploration and

```

(1) Initialize a population of  $M$  particles with random positions  $x_i$  in a given search space, and random
    velocities  $v_i$ ; the maximum iteration maxcycle;  $\omega_{\min}$ ,  $\omega_{\max}$ ;  $v_{\min}$ ,  $v_{\max}$ ; the length of chaotic sequence  $k$ .
(2) Set  $p_i = x_i$  ( $i = 1, \dots, SN$ ) and find  $p_g$ .
(3) while  $t \leq \text{maxcycle}$  do
(4)  $\omega = \omega_{\max} - (\omega_{\max} - \omega_{\min}) * t / \text{maxcycle}$ 
(5) for  $i = 1$  to  $SN$  do
(6)   for  $j = 1$  to  $n$  do
(7)     By (3) and (4), update the velocity of each particle.
(8)     By (5), update the position of each particle.
(9)   end for
(10)  if  $f(x_i) < f(p_i)$ 
(11)     $p_i = x_i$ , set  $L_i = 0$ ;
(12)  else
(13)    set  $L_i = L_i + 1$ .
(14)  end if
(15)  if  $f(x_i) < f(p_g)$ 
(16)    set  $p_g = x_i$ ,
(17)  end if
(18) end for
(19) for  $i = 1$  to  $SN$  do
(20)  if  $L_i = L$ 
(21)    By (8), to generate a new position, and replace  $x_i$ .
(22)  end if
(23) end for
(24) By (9)–(12), to chaotic search in  $p_g$ , and update  $p_g$  (if necessary).
(25)  $t = t + 1$ 
(26) end while

```

ALGORITHM 1: Framework of NPSO. Remark: in our algorithm, for simplicity, in (8), we can set $\tau = 1 - \omega$, which can meet our needs.

exploitation when solving complex multimodal problems. In order to get a better performance for PSO algorithm, many variants of PSO have been developed. For example, by using random value of inertia weight, Eberhart and Shi proposed a modified PSO, which can track the optima in a dynamic environment [16]. By utilizing the success rate of the particles, a new adaptive inertia weight strategy was presented [17]. Through using Cauchy mutation, a hybrid PSO (HPSO) was developed [18]. In [19], a hybrid PSO with a wavelet mutation (HWPSO) was given, in which the mutation incorporates with a wavelet function. To avoid premature convergence, a novel parameter automation strategy is proposed [20]. Valdez et al. introduced an improved FPSO + FGA hybrid method by combining the advantages of PSO and GA [21]. Based on mutation operator and different local search techniques, a superior solution guided PSO (SSG-PSO) was developed [22]. By using the second personal best and the second global best particle, two modified PSO algorithms are presented [23]. In order to avoid being trapped in local optima in the convergence process, some other improved PSO have been proposed, such as crossover [24], orthogonal learning strategy [25], chaos [26], and elitist learning strategy [27].

In this paper, by utilizing the information of the the best neighbor of each particle and the best particle of the entire population in the current iteration, a new Particle Swarm Optimization algorithm is proposed, which is named NPSO. To avoid premature, an abandoned mechanism is presented in our algorithm. Furthermore, for improving the global

convergence speed, a chaotic search is implemented in the best solution of each iteration.

The remainder of this paper is organized as follows. Section 2 describes the original PSO. Then our proposed modifications to PSO are described in Section 3. Numerical results and discussions are presented in Section 4. Finally, some concluding remarks are provided in Section 5.

2. Particle Swarm Optimization (PSO)

Assume that the search space is n -dimensional; M denotes the size of the swarm population. In PSO, each particle i ($i = 1, \dots, M$) has a position $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n})$ in the search space and a velocity $v_i = (v_{i,1}, v_{i,2}, \dots, v_{i,n})$ to indicate its current state. A position x_i denotes a feasible solution. The position x_i and the velocity v_i are updated by the best position $p_i = (p_{i,1}, p_{i,2}, \dots, p_{i,n})$ encountered by the particle so far and the best position $p_g = (p_{g,1}, p_{g,2}, \dots, p_{g,n})$ found by the entire population of particles according to the following equations:

$$\begin{aligned}
 v_{i,d}(t+1) &= \omega v_{i,d}(t) + c_1 r_1 (p_{i,d}(t) - x_{i,d}(t)) \\
 &\quad + c_2 r_2 (p_{g,d}(t) - x_{i,d}(t)), \\
 x_{i,d}(t+1) &= x_{i,d}(t) + v_{i,d}(t+1),
 \end{aligned} \tag{2}$$

where c_1 and c_2 are two learning factors which control the influence of the social and cognitive components, r_i ($i = 1, 2$) are random numbers in the range $[0, 1]$, and ω is the inertia

TABLE 1: Benchmark functions used in experiments.

Functions	Dimension	C	Range	Optimal value
$f_1 = \sum_{i=1}^n x_i^2$	30	US	$[-100, 100]$	0
$f_2 = \sum_{i=1}^n ix_i^2$	30	US	$[-10, 10]$	0
$f_3 = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	UN	$[-30, 30]$	0
$f_4 = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$	30	MS	$[-5.12, 5.12]$	0
$f_5 = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	30	MS	$[-600, 600]$	0
$f_6 = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	UN	$[-10, 10]$	0
$f_7 = -20 \exp\left(-0.2 * \sqrt{\frac{\sum_{i=1}^n z_i^2}{n}}\right) - \exp\left(\sum_{i=1}^n \cos\left(\frac{2\pi z_i}{n}\right)\right) + 20 + e$	30	MN	$[-32, 32]$	0
$f_8 = \frac{1}{4000} \sum_{i=1}^n z_i^2 - \prod_{i=1}^n \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$	30	MS	$[-600, 600]$	0
$f_9 = \sum_{i=1}^n z_i^2$	30	US	$[-100, 100]$	0
$f_{10} = \sum_{i=1}^n \left(\sum_{j=1}^i z_j\right)^2$	30	UN	$[-100, 100]$	0
$f_{11} = \sum_{i=1}^n (z_i^2 - 10 \cos(2\pi z_i) + 10)$	30	MS	$[-5.12, 5.12]$	0
$f_{12} = \sum_{i=1}^n \left(\sum_{j=1}^i z_j\right)^2 - 450$	30	UN	$[-100, 100]$	-450
$f_{13} = \sum_{i=1}^n (z_i^2 - 10 \cos(2\pi z_i) + 10) - 330$	30	MS	$[-5.12, 5.12]$	-330
$f_{14} = \sum_{i=1}^n \left(\sum_{k=0}^{k_{\max}} [a^k \cos(2 * \pi b^k (z_i + 0.5))] \right) - n \sum_{k=0}^{k_{\max}} [a^k \cos(2\pi b^k 0.5)] + 90, a = 0.5, b = 3, k_{\max} = 20$	30	US	$[-0.5, 0.5]$	90

C: characteristic, U: unimodal, M: multimodal, N: nonseparable, and S: separable.

weight, which ensures the convergence of the PSO algorithm and is decreased linearly.

3. The Novel PSO Algorithm (NPSO)

In the original PSO, since each particle moves in the search space guided only by its historical best solution p_i and the global best solution p_g , it may get trapped in a local optimal solution when current global best solution in a local optimum and not easy for the particle escapes from it. To solve such problem, in this section, three improvement strategies are proposed.

3.1. The First Improvement Strategy. From (2), we observe that only the information of the historical best position p_i of each particle and the global best position p_g of all particles is utilized. As a matter of fact, the information of the best neighbor $p_{N_i}^{\text{best}}$ of the particle i may provide a better guidance than p_i . The details are given as follows.

Firstly, we explain how to define the neighbors and to determine the best neighbor $p_{N_i}^{\text{best}}$ of the particle i . In order to define appropriate neighbors, different approaches could

be used. In this paper, the neighbors of x_i are defined by using the mean Euclidean distance between x_i and the rest of solutions. Let $d(i, j)$ be the Euclidean distance between x_i and x_j and let md_i be the mean Euclidean distance for x_i . Then md_i can be computed as follows:

$$\text{md}_i = \frac{\sum_{j=1}^M d(i, j)}{M - 1}. \quad (3)$$

By (3), if $d(i, j) < \text{md}_i$, then x_j could be accepted as a neighbor of x_i .

In addition, we can also use a more general and flexible definition to determine a neighbor of x_i :

$$\begin{aligned} &\text{if } d(i, j) \\ &\leq r \\ &\times \text{md}_i, \text{ then } x_j \text{ is a neighbor of } x_i; \text{ else it is not.} \end{aligned} \quad (4)$$

If (4) is used, then a new parameter r ($r \geq 0$), which refers to the ‘‘neighbourhood radius,’’ will be added to the parameters of PSO. If $r = 0$, it turns to the standard PSO. With the

TABLE 2: NPSO performance comparison with PSO.

Function	Max iteration	Algorithm	Mean	SD	Min
f_1	1000	PSO	$5.06e - 003$	$1.26e - 003$	$4.10e - 003$
		NPSO	$5.66e - 141$	$9.80e - 141$	$1.97e - 198$
f_2	1000	PSO	$1.81e - 001$	$1.65e - 001$	$3.08e - 002$
		NPSO	$2.41e - 129$	$4.17e - 129$	$2.91e - 201$
f_3	1000	PSO	$2.95e + 001$	$3.55e - 001$	$2.91e + 001$
		NPSO	$2.217e - 005$	$3.48e - 005$	$6.81e - 013$
f_4	1000	PSO	$2.90e + 001$	0	$2.90e + 001$
		NPSO	0	0	0
f_5	1000	PSO	$4.97e - 004$	$4.65e - 004$	$1.91e - 004$
		NPSO	0	0	0
f_6	1000	PSO	$4.55e - 096$	$3.06e - 096$	$1.12e - 096$
		NPSO	$4.49e - 104$	$7.78e - 104$	$7.40e - 117$
f_7	1000	PSO	$2.03e + 001$	$1.22e - 002$	$2.02e + 001$
		NPSO	$3.29e - 007$	$3.76e - 007$	$1.75e - 008$
f_8	1000	PSO	$1.054e + 002$	$9.24e - 002$	$1.053e + 002$
		NPSO	$3.68e - 009$	$6.27e - 009$	$6.00e - 012$
f_9	1000	PSO	$2.32e + 004$	$3.03e + 002$	$2.30e + 004$
		NPSO	$6.30e - 012$	$1.04e - 011$	$1.29e - 014$
f_{10}	1000	PSO	$3.84e + 007$	$7.18e + 004$	$3.83e + 007$
		NPSO	$3.79e - 009$	$6.41e - 009$	$3.29e - 012$
f_{11}	1000	PSO	$2.44e + 001$	$1.25e + 000$	$2.33e + 001$
		NPSO	$1.66e - 004$	$2.03e - 004$	$1.25e - 009$
f_{12}	1000	PSO	$8.34e + 002$	$1.05e + 003$	$1.22e + 002$
		NPSO	$-4.499e + 002$	$4.25e - 004$	-450
f_{13}	1000	PSO	$-2.59e + 002$	$1.25e + 001$	$-2.69e + 002$
		NPSO	-330	0	-330
f_{14}	1000	PSO	$1.24e + 002$	$1.02e + 000$	$1.23e + 002$
		NPSO	$9.001e + 001$	$2.23e - 002$	90

value of r increasing, the neighborhood of x_i enlarges or its neighborhood shrinks as the value of r decreases. Once the neighbors are determined, we select the best position among the neighbors of x_i as the best neighbor $p_{N_i}^{\text{best}}$.

After determining the the best neighbor $p_{N_i}^{\text{best}}$, we give a new way moving for each particle. If $f(p_{N_i}^{\text{best}}) < f(p_i)$, then set

$$v_{i,d}(t+1) = \omega v_{i,d}(t) + c_1 r_1 (p_{N_i,d}^{\text{best}}(t) - x_{i,d}(t)) + c_2 r_2 (p_{g,d}(t) - x_{i,d}(t)). \quad (5)$$

If $f(p_{N_i}^{\text{best}}) > f(p_i)$, then set

$$v_{i,d}(t+1) = \omega v_{i,d}(t) + c_1 r_1 (p_{i,d}(t) - x_{i,d}(t)) + c_2 r_2 (p_{g,d}(t) - x_{i,d}(t)). \quad (6)$$

After that, x_i moves to a new position by the following equation:

$$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1). \quad (7)$$

In our algorithm, it is obvious that before each particle moves, it first watches the region which is centered by itself,

selects the best neighbour, and then uses (5)–(7) to generate the next position.

3.2. The Second Improvement Strategy. To avoid premature, in our algorithm, an abandoned mechanism is proposed.

Assume that “ L ” is a predetermined number, which will be used to determine whether the position x_i of the particle i should be abandoned. At the beginning, set $l(i) = 0$ ($i = 1, \dots, M$). With the implementation of the algorithm, if the position x_i was not improved, then set $l(i) = l(i) + 1$; else set $l(i) = 0$. If the position x_i can not be improved anymore when L is achieved, that is, $l(i) = L$, then the position will be abandoned and a new position x_{new} will replace it, which is generated by using the following equation:

$$x_{\text{new}} = \tau * p_g + \phi * (p_g - x_i), \quad (8)$$

where ϕ is a random number in the range $[-1, 1]$ and τ is the inertia weight that controls the impact of the optimal solution at current iteration, which is increased linearly.

From (8), it can be seen that, in the early stage of the algorithm, x_{new} will be generated with a large randomness and in the late stage, it will be generated near the global best position p_g .

FIGURE 1: Convergence rates on test functions.

TABLE 3: Benchmark functions used in experiments.

Functions	Dimension (n)	C	Range	Optimal value
$f_1 = \sum_{i=1}^n x_i^2$	30	US	$[-100, 100]$	0
$f_2 = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	UN	$[-10, 10]$	0
$f_3 = \sum_{i=1}^n \left(\sum_{j=1}^i x_j \right)^2$	30	UN	$[-100, 100]$	0
$f_4 = \sum_{i=1}^n i x_i^4 + \text{random}[0, 1]$	30	MS	$[-1.28, 1.28]$	0
$f_5 = \sum_{i=1}^{n-1} \left[100 (x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \right]$	30	UN	$[-30, 30]$	0
$f_6 = \sum_{i=1}^n (\lfloor x_i + 0.5 \rfloor)^2$	30	US	$[-1.28, 1.28]$	0
$f_7 = \sum_{i=1}^n (x_i^2 - 10 \cos(2\pi x_i) + 10)$	30	MS	$[-5.12, 5.12]$	0
$f_8 = \sum_{i=1}^n (y_i^2 - 10 \cos(2\pi y_i) + 10),$ if $ x_i < \frac{1}{2}$, $y_i = x_i$; else $y_i = \frac{\text{round}(2x_i)}{2}$	30	US	$[-5.12, 5.12]$	0
$f_9 = -20 \exp \left(-0.2 * \sqrt{\frac{\sum_{i=1}^n x_i^2}{n}} \right) - \exp \left(\sum_{i=1}^n \cos \left(\frac{2\pi x_i}{n} \right) \right) + 20 + e$	30	MN	$[-32, 32]$	0
$f_{10} = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos \left(\frac{x_i}{\sqrt{i}} \right) + 1$	30	MS	$[-600, 600]$	0
$f_{11} = \sum_{i=1}^n -x_i * \sin(\sqrt{ x_i })$	30	MS	$[-500, 500]$	$418.98288 * n$
$f_{12} = \sum_{i=1}^n x_i^2$	30	US	$[-10, 190]$	0

C: characteristic, U: unimodal, M: multimodal, N: nonseparable, and S: separable.

3.3. The Third Improvement Strategy. To improve the global convergence of NPSO, a chaotic search operator is adopted. Next, we give the details.

Let p_g be the best solution of the current iteration. Firstly, utilize the following equation (9) to generate chaotic variable ch_i :

$$ch_{i+1} = 4 * ch_i * (1 - ch_i), \quad 1 \leq i \leq K, \quad (9)$$

where K is the length of chaotic sequence and $ch_0 \in (0, 1)$ is a random number. Then map ch_i to a chaotic vector CH_i in the interval $[l, u]$:

$$CH_i = l + ch_i * (u - l), \quad i = 1, \dots, K, \quad (10)$$

where l and u are the lower bound and upper bound of variable x , respectively. Finally, a new candidate solution $\hat{x}_i$ is obtained by the following equation:

$$\hat{x}_i = (1 - \lambda) * p_g + \lambda * CH_i, \quad i = 1, \dots, K, \quad (11)$$

where λ is a shrinking factor, which is defined as follows:

$$\lambda = \frac{\text{maxcycle} - t + 1}{\text{maxcycle}}, \quad (12)$$

where maxcycle is the maximum number of iterations and t is the number of iterations.

By (11) and (12), it can be seen that λ will become smaller with the increase of evolutionary generation; that is, the local search range will become smaller with the process of evolution.

Based on the abovementioned explanation, the pseudo-code of the NPSO algorithm is given in Algorithm 1.

4. Experimental Results and Discussion

4.1. Experiments 1. In this subsection, the performance of NPSO algorithm is compared to PSO algorithm by evaluating convergence and best solution found for 14 benchmark functions, where f_7 - f_{11} are shifted functions and f_{12} - f_{14} are rotated functions. The characteristics, dimensions, initial range, and formulations of these functions are listed in Table 1. z is $\mathbf{x} - \mathbf{o}$ for f_7 - f_{11} and $(\mathbf{x} - \mathbf{o}) * \mathbf{M}$ for f_{12} - f_{14} ; $\mathbf{o} = (o_1, o_2, \dots, o_n)$ is the shifted global optimum; $\mathbf{M}$ is the $n \times n$ orthogonal matrix.

The proposed algorithm NPSO and PSO are coded in Matlab 7.0, and the experiments' platform is a personal computer with Pentium 4, 3.06 GHz CPU, 512 M memory, and Windows XP.

The parameters of algorithms are given as follows. The common parameters are the dimension $n = 30$, the population size $M = 30$, the maximum number of iterations maxcycle = 2000, $\omega_{\min} = 0.4$, $\omega_{\max} = 0.9$, $v_{\min} = 0.4$, and $v_{\max} = 0.9$. NPSO settings are as follows: the length K of

TABLE 4: The mean and standard deviation of the best solutions of six PSO variants on 12 test problems in 200,000 function evaluations.

	f_1	f_2	f_3
CPSO	$5.146e - 013$ ($7.7588e - 025$)	$1.2534e - 007$ ($1.1791e - 014$)	$1.8889e + 003$ ($9.9106e + 006$)
CLPSO	$4.894e - 039$ ($6.7814e - 078$)	$8.8677e - 024$ ($7.9008e - 049$)	$1.9217e + 002$ ($3.8433e + 003$)
FIPS	$4.588e - 027$ ($1.9577e - 053$)	$2.3239e - 016$ ($1.1406e - 032$)	$9.4634e + 000$ ($2.5976e + 001$)
Frankenstein	$2.409e - 016$ ($2.0047e - 031$)	$1.5804e - 011$ ($1.0301e - 022$)	$1.7315e + 002$ ($9.1577e + 003$)
AIWPSO	$3.370e - 134$ ($5.1722e - 267$)	$1.6534e - 062$ ($7.7348e - 123$)	$1.9570e - 010$ ($1.2012e - 019$)
NPSO	$6.597e - 217$ ($1.632e - 235$)	$2.5016e - 109$ ($4.3331e - 109$)	$5.4552e - 193$ ($9.2401e - 198$)
	f_4	f_5	f_6
CPSO	$1.0764e - 002$ ($2.7698e - 005$)	$8.2648e - 001$ ($2.3449e + 00$)	0.0 (0.0)
CLPSO	$4.0642e - 003$ ($9.6184e - 007$)	$1.3217e + 001$ ($2.1480e + 002$)	0.0 (0.0)
FIPS	$3.3047e - 003$ ($8.6680e - 007$)	$2.6714e + 001$ ($2.0025e + 002$)	0.0 (0.0)
Frankenstein	$4.1690e - 003$ ($2.4012e - 006$)	$2.8156e + 001$ ($2.3132e + 002$)	0.0 (0.0)
AIWPSO	$5.5241e - 003$ ($1.5358e - 005$)	$2.5003e + 000$ ($1.5978e + 001$)	0.0 (0.0)
NPSO	$8.4591e - 006$ ($8.4932e - 006$)	$3.0621e - 006$ ($4.0283e - 006$)	0.0 (0.0)
	f_7	f_8	f_9
CPSO	$3.6007e - 013$ ($1.5035e - 024$)	$5.3717e - 013$ ($5.9437e - 024$)	$1.6091e - 007$ ($7.8608e - 014$)
CLPSO	0.0 (0.0)	$1.3333e - 001$ ($1.1954e - 001$)	$9.2371e - 015$ ($6.6156e - 030$)
FIPS	$5.8502e + 001$ ($1.9185e + 002$)	$6.1883e + 001$ ($1.4013e + 002$)	$1.3856e - 014$ ($2.3227e - 029$)
Frankenstein	$7.3836e + 001$ ($3.7055e + 002$)	$7.0347e + 001$ ($2.9600e + 002$)	$2.1792e - 009$ ($1.7187e - 018$)
AIWPSO	$1.6583e - 001$ ($2.1051e - 001$)	$1.1842e - 016$ ($4.2073e - 031$)	$6.9870e - 015$ ($4.2073e - 031$)
NPSO	0.0 (0.0)	$4.0407e - 215$ ($1.5575e - 220$)	$-8.8817e - 016$ (0.0)
	f_{10}	f_{11}	f_{12}
CPSO	$2.1245e - 002$ ($6.3144e - 004$)	$-1.2127e + 004$ ($3.3795e + 004$)	$5.4282e - 014$ ($8.2868e - 027$)
CLPSO	0.0 (0.0)	$-1.2546e + 004$ ($4.2567e + 003$)	$9.9748e - 039$ ($3.7661e - 084$)
FIPS	$2.4776e - 004$ ($1.8266e - 006$)	$-1.1052e + 004$ ($9.4421e + 005$)	$2.6033e + 002$ ($2.1785e + 004$)
Frankenstein	$1.4736e - 003$ ($1.2846e - 005$)	$-1.1221e + 004$ ($2.7708e + 005$)	$5.1953e + 004$ ($1.1136e + 009$)
AIWPSO	$2.8524e - 002$ ($7.6640e - 004$)	$-1.2569e + 004$ ($1.1409e - 025$)	$1.8317e - 137$ ($3.4534e - 273$)
NPSO	0.0 (0.0)	$-1.2569e + 004$ ($2.5299e - 007$)	$2.2740e - 198$ ($2.6952e - 186$)

chaotic sequence and the limit L of a position abandoned are set to 5; $r = 0.01$. Each experiment is repeated 30 times independently. The global minimums (Min), maximum number of iterations (Max iteration), mean best values (Mean), and standard deviations (SD) are given in Table 2. To show the convergence speed of PSO and NPSO more clearly, the convergence graphs of PSO and NPSO are shown in Figure 1.

From Table 2, it can be seen that the NPSO performs better than PSO on all test functions. From Figure 1, it can be seen that the convergence speed of NPSO is more fast than PSO.

4.2. Experiments 2. In this subsection, to further test the efficiency of NPSO, it is compared with other five algorithms, that is, CPSO [28], CLPSO [29], FIPS [30], Frankenstein [31], and AIWPSO [17].

Twelve benchmark functions are used for the comparison. The characteristics, dimensions, initial range, and formulations of these functions are listed in Table 3.

In order to make a fair comparison, the maximum number of function evaluations (maxFEs) is set to $2e5$ for all algorithms; the population size is 20. The other parameters for NPSO are set as in Experiments 1. The comparison results

are presented in Table 4. For the sake of convenience and reliability, except for the NPSO algorithm, the rest of results reported here are taken directly from the literature [17].

From Table 4, it can be seen that NPSO is significantly better than the other five algorithms on almost all the test functions, except for the function f_{11} .

5. Conclusion

In this paper, by utilizing the information of the best neighbor of each particle and the best solution in the current iteration, we presented a new move equation. After that, based on the other two improvement strategies, a novel Particle Swarm Optimization algorithm NPSO was proposed. The performance of NPSO was compared with the standard PSO and other five variants of PSO. The results showed that NPSO presents promising results for considered problems.

In the future, the adaption of the parameters in NPSO can be studied to improve its performance.

Conflict of Interests

The authors declare that there is no conflict of interest regarding the publication of this paper.

Acknowledgments

The research was supported by NSFC (U1404105, 11171094); the Key Scientific and Technological Project of Henan Province (142102210058); the Doctoral Scientific Research Foundation of Henan Normal University (qd12103); the Youth Science Foundation of Henan Normal University (2013qk02); Henan Normal University National Research Project to Cultivate the Funded Projects (01016400105); the Henan Normal University Youth Backbone Teacher Training.

References

- [1] N. Gabere, *Simulated annealing driven pattern search algorithms for global optimization [M.S. thesis]*, University of the Witwatersrand, Johannesburg, South Africa, 2007.
- [2] K. Deep and M. Thakur, "A new mutation operator for real coded genetic algorithms," *Applied Mathematics and Computation*, vol. 193, no. 1, pp. 211–230, 2007.
- [3] P. Kaelo and M. M. Ali, "Integrated crossover rules in real coded genetic algorithms," *European Journal of Operational Research*, vol. 176, no. 1, pp. 60–76, 2007.
- [4] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.
- [5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, Perth, Australia, November–December 1995.
- [6] M. Dorigo and T. Stutzle, *Ant Colony Optimization*, MIT Press, Cambridge, Mass, USA, 2004.
- [7] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Erciyes University, Kayseri, Turkey, 2005.
- [8] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001.
- [9] N. W. Oo, "A comparison study on particle swarm and evolutionary particle swarm optimization using capacitor placement problem," in *Proceedings of the 2nd IEEE International Conference on Power and Energy Conference*, pp. 1208–1211, Johor Bahru, Malaysia, December 2008.
- [10] B. Wang, N.-L. Tai, H.-Q. Zhai, J. Ye, J.-D. Zhu, and L.-B. Qi, "A new ARMAX model based on evolutionary algorithm and particle swarm optimization for short-term load forecasting," *Electric Power Systems Research*, vol. 78, no. 10, pp. 1679–1685, 2008.
- [11] H. Wang and F. Qian, "An improved particle swarm optimizer with behavior-distance models and its application in soft-sensor," in *Proceedings of the 7th World Congress on Intelligent Control and Automation (WCICA '08)*, pp. 4473–4478, IEEE, Chongqing, China, June 2008.
- [12] S. Naka, T. Genji, T. Yura, and Y. Fukuyama, "A hybrid particle swarm optimization for distribution state estimation," *IEEE Transactions on Power Systems*, vol. 18, no. 1, pp. 60–68, 2003.
- [13] A. A. El-Dib, H. K. M. Youssef, M. M. El-Metwally, and Z. Osman, "Maximum loadability of power systems using hybrid particle swarm optimization," *Electric Power Systems Research*, vol. 76, no. 6–7, pp. 485–492, 2006.
- [14] B. Liu, L. Wang, Y.-H. Jin, F. Tang, and D.-X. Huang, "Directing orbits of chaotic systems by particle swarm optimization," *Chaos, Solitons and Fractals*, vol. 29, no. 2, pp. 454–461, 2006.
- [15] M. F. Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, "A particle swarm optimization algorithm for makespan and total flowtime minimization in the permutation flowshop sequencing problem," *European Journal of Operational Research*, vol. 177, no. 3, pp. 1930–1947, 2007.
- [16] R. C. Eberhart and Y. H. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 1, pp. 94–100, Seoul, South Korea, May 2001.
- [17] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11, no. 4, pp. 3658–3670, 2011.
- [18] H. Wang, C. Li, Y. Liu, and S. Zeng, "A hybrid particle swarm algorithm with cauchy mutation," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '07)*, pp. 356–360, IEEE, Honolulu, Hawaii, USA, April 2007.
- [19] S. H. Ling, H. H. C. Iu, K. Y. Chan, H. K. Lam, B. C. W. Yeung, and F. H. Leung, "Hybrid particle swarm optimization with wavelet mutation and its industrial applications," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 38, no. 3, pp. 743–763, 2008.
- [20] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 240–255, 2004.
- [21] F. Valdez, P. Melin, and O. Castillo, "Modular Neural Networks architecture optimization with a new nature inspired method using a fuzzy combination of Particle Swarm Optimization and Genetic Algorithms," *Information Sciences*, vol. 270, pp. 143–153, 2014.
- [22] G. H. Wu, D. S. Qiu, Y. Yu, W. Pedrycz, M. Ma, and H. Li, "Superior solution guided particle swarm optimization combined with local search techniques," *Expert Systems with Applications*, vol. 41, no. 16, pp. 7536–7548, 2014.
- [23] Y.-B. Shin and E. Kita, "Search performance improvement of particle swarm optimization by second best particle information," *Applied Mathematics and Computation*, vol. 246, pp. 346–354, 2014.
- [24] Y.-P. Chen, W.-C. Peng, and M.-C. Jian, "Particle swarm optimization with recombination and dynamic linkage discovery," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 37, no. 6, pp. 1460–1470, 2007.
- [25] Z.-H. Zhan, J. Zhang, Y. Li, and Y.-H. Shi, "Orthogonal learning particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 6, pp. 832–847, 2011.
- [26] B. Alatas, E. Akin, and A. B. Ozer, "Chaos embedded particle swarm optimization algorithms," *Chaos, Solitons and Fractals*, vol. 40, no. 4, pp. 1715–1734, 2009.
- [27] Z.-H. Zhan, J. Zhang, Y. Li, and H. S.-H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.
- [28] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

- [29] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.
- [30] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.
- [31] M. A. M. de Oca, T. Stützle, M. Birattari, and M. Dorigo, "Frankenstein's PSO: a composite particle swarm optimization algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.