Research Article

An Improved Brain Storm Optimization with Differential Evolution Strategy for Applications of ANNs

Zijian Cao,^{1,2} Xinhong Hei,¹ Lei Wang,¹ Yuhui Shi,³ and Xiaofeng Rong²

¹School of Computer Science and Engineering, Xi'an University of Technology, Xi'an 710048, China

²School of Computer Science and Engineering, Xi'an Technological University, Xi'an 710021, China

³Xi'an Jiaotong-Liverpool University, Suzhou 215123, China

Correspondence should be addressed to Xinhong Hei; heixinhong@xaut.edu.cn

Received 3 May 2015; Revised 28 July 2015; Accepted 29 July 2015

Academic Editor: George S. Dulikravich

Copyright © 2015 Zijian Cao et al. 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.

Brain Storm Optimization (BSO) algorithm is a swarm intelligence algorithm inspired by human being's behavior of brainstorming. The performance of BSO is maintained by the creating process of ideas, but when it cannot find a better solution for some successive iterations, the result will be so inefficient that the population might be trapped into local optima. In this paper, we propose an improved BSO algorithm with differential evolution strategy and new step size method. Firstly, differential evolution strategy is incorporated into the creating operator of ideas to allow BSO jump out of stagnation, owing to its strong searching ability. Secondly, we introduce a new step size control method that can better balance exploration and exploitation at different searching generations. Finally, the proposed algorithm is first tested on 14 benchmark functions of CEC 2005 and then is applied to train artificial neural networks. Comparative experimental results illustrate that the proposed algorithm performs significantly better than the original BSO.

1. Introduction

In the past few decades, swarm intelligence algorithms, which were derived from the concepts, principles, and mechanisms of nature-inspired computation, have attracted more and more attention from researchers. Though swarm intelligence is a new category of stochastic, population-based optimization algorithms compared with evolutionary algorithms (EAs) such as Genetic Algorithms (GA) [1], Evolutionary Programming (EP) [2], Evolutionary Strategies (ES) [3], Genetic Programming (GP) [4], and differential evolution (DE) [5], so far, a lot of swarm intelligence optimization algorithms, such as Particle Swarm Optimization (PSO) [6, 7], have been proposed to tackle many challenging complex computational problems in science and industry.

Brain Storm Optimization (BSO) algorithm [8, 9] is a swarm intelligence algorithm inspired by human being's behavior of brainstorming. Derived from human creative problem-solving process, BSO has achieved more and more attention and has been applied to the function of optimization, optimal satellite formation reconfiguration [10],

the design of DC Brushless Motor [11], economic dispatch considering wind power [12], multiobjective optimization [13, 14], and so forth. Due to the flexibility and scalability of the original BSO algorithm [15], many new versions of BSOs have been designed and implemented. To improve the performance and reduce the computation burden of BSO algorithm, Zhan et al. [16] modified the grouping operator and the creating operator and designed a BSO variant named Modified BSO (MBSO). Cheng et al. [17] used two kinds of partial reinitialization strategies to improve the population diversity in BSO algorithm. Recently, Yang et al. [18] proposed advanced discussion mechanism-based brain storm optimization (ADMBSO), which incorporated intercluster and intracluster discussions into BSO to control global and local searching ability. However, optimization problems have become more and more complex, from simple unimodal functions to hybrid rotated shifted multimodal functions. To the best of our knowledge, all variants of BSO have not been tested on the complex benchmark functions of CEC2005 [19]. Hence, more effective improvement to the original BSO algorithms is always necessary.

Among existing metaheuristics, differential evolution (DE) is a simple yet powerful global optimization technique with successful applications in various areas [5]. To achieve the most satisfactory optimization performance, an improved BSO algorithm with differential evolution strategy and new step size method named BSODE, which can maintain both the exploration ability and the diversity of population, is proposed in the paper. The improved algorithm will be first tested on 14 benchmark functions of CEC 2005 and then will be applied to train artificial neural networks.

The rest of this paper is organized as follows. Section 2 briefly introduces the original BSO algorithm and the basic DE operator. Section 3 describes the improved BSO with differential evolution strategy (BSODE). Section 4 presents the tests on 14 benchmark functions. The applications for artificial neural network are given in Section 5, followed by conclusions in Section 6.

2. Related Work

2.1. Brain Storm Optimization. The BSO algorithm is motivated by the philosophy of brainstorming, and brainstorming is a widely used tool for increasing creativity in organizations which has achieved wide acceptance as a mean of facilitating creative thinking [20]. A potential solution in the solution space represents an idea in BSO. BSO sticks to the rules of interchange of ideas by a team and uses clustering, replacing, and creating operators to produce global optimum generation by generation. In the procedure of BSO, firstly, N ideas are randomly initialized within the solution space, and then each idea is evaluated according to its fitness function. Next m points of cluster center are also randomly selected and initialized like N ideas, where m is less than N . As presented in [8, 9], the rest of the process in BSO can be described as follows.

2.1.1. Clustering Individuals. Clustering is a process of grouping similar objects together, and, during each generation, all the ideas are clustered into m clusters according to idea (or individual) features. The best idea in each center is chosen as its cluster center, and the clustering operation can refine a search area. K -means is a popular algorithm used in clustering; herein it is used in clustering operation.

2.1.2. Disrupting Cluster Center. Cluster center disrupting operation randomly chooses a cluster center and replaces it with a newly generated idea with a probability of p_{replace} , which is also named as the replacing operation. The value of p_{replace} is utilized to control the probability to replace a cluster center by a randomly generated solution. This is used to avoid premature convergence and help individuals “jump out” of the local optima.

2.1.3. Creating Individuals. To maintain the diversity of population, a new idea (individual) can be generated based on one idea or two in one cluster or two, respectively. In the creating operation, BSO first randomly chooses one cluster or two according to a probability of p_{one} . Then, in the basis of choosing one cluster or two, an idea of cluster center or

a random idea is selected with a probability of $p_{\text{one_center}}$ and $p_{\text{two_center}}$. The selecting operation is defined below as

$$X_{\text{select}} = \begin{cases} X_i & \text{one cluster} \\ \text{rand} * X_{i1} + (1 - \text{rand}) * X_{i2} & \text{two clusters,} \end{cases} \quad (1)$$

where rand is a random value between 0 and 1. After choosing one idea or two, the selected idea(s) is updated according to

$$X_{\text{new}} = X_{\text{select}} + \xi * \text{normrnd}(0, 1), \quad (2)$$

where normrnd is the Gaussian random value with mean 0 and variance 1 and ξ is an adjusting factor slowing the convergence speed down as the evolution goes, which can be expressed as

$$\xi = \text{rand} * \log \text{sig} \left(\frac{(0.5 * \text{max_iteration} - \text{current_iteration})}{k} \right), \quad (3)$$

where rand is a random value between 0 and 1. The max_iteration and current_iteration denote the maximum number of iterations and current number of iterations, respectively. The logsig is a logarithmic sigmoid transfer function, and such form is beneficial to global search ability at the beginning of the evolution and enhances local search ability when the process is approaching to the end. k is a predefined parameter for changing slopes of the logsig function. The new created idea is evaluated, and if the fitness value is better than the current idea, the old idea will be replaced by the new one.

2.2. Differential Evolution. Differential evolution (DE) [5] is a population-based and stochastic optimization algorithm. Like other EAs, DE begins with an initial population vector containing a number of target individuals. The current generation evolves into the next generation through evolutionary operations repeatedly until the termination condition is attained. For the original DE, the mutation, crossover, and selection operators are defined as follows.

2.2.1. Mutation Operation. The mutation operation of classical DE scheme (DE/rand/bin) can be summarized as follows:

$$V_i^k = X_{r1}^k + F * (X_{r2}^k - X_{r3}^k), \quad (4)$$

where k is the generation number, the indices $r1$, $r2$, and $r3$ are mutually exclusive integers randomly chosen from the range between 1 and N , and F is a mutation scaling factor which affects the differential variation between two individuals.

2.2.2. Crossover Operation. After the process of mutation, crossover operation is used to each pair of the target vector in order to enhance the potential diversity of the population. DE applies a crossover operator on X_i and V_i to generate the

offspring individual U_i at the k th generation. The crossover operation is defined as

$$U_{ij}^k = \begin{cases} V_{ij}^k & \text{if rand} \leq \text{CR} \\ X_{ij}^k & \text{otherwise,} \end{cases} \quad (5)$$

where rand is a random value between 0 and 1 and CR is a parameter of crossover probability.

2.2.3. Selection Operation. The greedy selection is employed by means of comparing a parent and its corresponding offspring. The selection operation at the k th generation is described as

$$X_i^{k+1} = \begin{cases} U_i^k & \text{if } f(U_i^k) \leq f(X_i^k) \\ X_i^k & \text{otherwise,} \end{cases} \quad (6)$$

where $f(X)$ is the objective function value of the trial vector X .

3. The Improved Algorithm: BSODE

In order to achieve better performance, an improved BSO algorithm needs to make use of both the global search information about the search space and the local search information of solutions found so far. The global search information can guide the search for exploring promising areas, while the local search information of solutions can be helpful for exploitation. In this paper, we effectively integrate the mutation and crossover scheme of DE into BSO in the idea creating operator of intracluster and intercluster. The key reason for integrating differential evolution strategy into BSO is that it can take advantage of DE that is mainly based on the distance and direction information and has the advantage of not being biased towards any prior defined guider.

Although the differential strategy has been utilized in [16] to design a BSO variant named Modified BSO (MBSO) and [10] to develop another BSO variant termed Close Loop BSO (CLBSO), our BSODE algorithm has significant differences with MBSO and CLBSO. On one hand, in the MBSO and CLBSO, a new idea Y_i is created by adding the difference of two distinct random ideas X_{r1} and X_{r2} from all the current ideas to a current idea X_i according to

$$Y_i = X_i + \text{rand} * (X_{r1} - X_{r2}), \quad (7)$$

where rand is a random value between 0 and 1 and indices $r1$ and $r2$ are mutually exclusive integers randomly chosen from the range between 1 and N . Even though formula (7) is similar to the mutation formula (4) of DE algorithm, they are obviously different in essence. In formula (4), F , the mutation scaling factor, is a real constant between 0 and 2, and the idea behind F is to control the amplification of the differential variation. However, in formula (7), the factor, rand, represents a random direction scaling, and the scope of distance is less than F in some cases. In mutation operator, our BSODE will be strictly in accordance with formula (4) of DE algorithm. On the other hand, the MBSO and CLBSO do

FIGURE 1: Flowchart showing the working of intracluster differential evolution operator.

not take the crossover operator between ideas into account, and our BSODE uses the crossover operator to enhance the performance of local exploitation according to formula (5).

Furthermore, in our BSODE, we mimic Osborn's rule that different unusual ideas generating from different perspectives and suspending assumptions by participants are welcome in the brainstorming of human being. According to the inspiration, we add the differential evolution strategy into the normal idea to generate more different ideas. According to the above analysis, the differential evolution operators of BSODE algorithm include intracluster differential evolution operator and intercluster differential evolution operator. The detailed differential evolution operators of the BSODE algorithm are described below.

3.1. Differential Evolution Operator

3.1.1. Intracluster Differential Evolution Operator. In creating operator of BSO, if the selected idea is a normal one in one cluster according to the probability of p_one and $(1, p_one_center)$, we will let the idea to learn from the differential value of two random selected ideas and the cluster center. The differential evolution operator of a normal idea is defined as

$$X_{new} = X_{center} + F * (X_{r1} - X_{r2}), \quad (8)$$

where F is the mutation scaling factor which affects the differential variation between two ideas and the indices $r1$ and $r2$ are mutually exclusive integers randomly chosen in the selected cluster. Then, according to formula (5), the crossover operation is used to generate new solutions by shuffling competing vectors and also to increase the diversity of the population.

Figure 1 shows the working process of intracluster differential evolution operator of a normal idea. The search space of the normal idea includes the area between two random ideas X_{r1} and X_{r2} and the cluster center X_{center} , and it can be seen that the local exploitation extends within the cluster space.

3.1.2. Intercluster Differential Evolution Operator. In creating operator of BSO, if the selected ideas are two normal ones in two clusters according to the probability of $(1, p_one)$ and $(1, p_two_center)$, and we will let the idea to learn from the differential value of two randomly selected normal ideas in two clusters and the best idea in all clusters, respectively.

```

(1) Begin
(2)   if rand() <  $p\_one$ 
(3)     Select a cluster according to the probabilities based on number of individuals
(4)     if rand() <  $p\_one\_center$ 
(5)       execute original operator
(6)     else
(7)       execute intra-cluster DE operator to create  $X_{new}$  using formula (8) and (5)
(8)     endif
(9)   else
(10)    Randomly select two clusters
(11)    if rand() <  $p\_two\_center$ 
(12)      execute original operator
(13)    else
(14)      execute inter-cluster DE operator to create  $X_{new}$  using formula (9) and (5)
(15)    endif
(16)  endif
(17) end

```

ALGORITHM 1: Pseudocode of the differential evolution operator().

FIGURE 2: Flowchart showing the working of intercluster differential evolution operator.

The differential evolution operator of two normal ideas is defined as

$$X_{new} = GlobalIdea + F * (X_{r1} - X_{r2}), \quad (9)$$

where F is the mutation scaling factor, $GlobalIdea$ is the best idea in all clusters, and X_{r1} and X_{r2} denote the normal idea of cluster 1 and cluster 2. Then, according to formula (5), the crossover operation is used to generate new solutions.

Figure 2 shows the working process of two normal ideas in intercluster differential evolution operator. The search space of the two normal ideas includes the area between global best idea point and two normal ideas, and it can be seen that the global exploration extends between the two clusters.

According to the above analysis, the pseudocode of differential evolution operator about intracuster and intercluster is summarized in Algorithm 1.

3.2. A New Step Size Method. To adjust the convergence speed as the evolution goes in idea generation, the original BSO algorithm defines an adjusting factor ξ described by formula (3). Figure 3(a) shows the adjusting factor which controls the scale of step. We can observe that at first the adjusting factor

keeps around 1, while when half the number of generations has been reached, it rapidly turns to be near 0. This method to control the size of step can also balance exploration and exploitation at different searching generations. However, it just takes effect only for very short interval. Hence, we introduce a simple new step size method which is shown in Figure 3(b). The dynamic mutation function is described as follows:

$$\xi = rand * \exp \left(1 - \frac{(\max_iteration)}{(\max_iteration - \text{current_iteration} + 1)} \right), \quad (10)$$

where $rand$ is a random value between 0 and 1. The $\max_iteration$ and current_iteration denote the maximum number of iterations and current number of iterations, respectively.

3.3. Flowchart and Pseudocode of the BSO. As previously analyzed, the complete flowchart of the BSO algorithm is shown in Figure 4.

This efficient differential evolution operator in inter-cluster improves the global search capability and avoids convergence to local minima. At the same time, the differential evolution operator in intracuster extends the capability of local exploitation and accelerates the convergence. The pseudocode of the BSO is summarized in Algorithm 2.

4. Benchmark Functions Optimization Problem

The numerical benchmark functions are used for comparing the performance and accuracy of our proposed BSO algorithm with the original BSO algorithm and other variants of DE in this section. For a fair comparison, all the experiments are conducted on the same machine with an Intel 3.4 GHz

FIGURE 3: The comparison chart of adjusting factor between BSO and BSODE.

FIGURE 4: Flow chart of BSODE algorithm.


```

(1) Begin
(2)   Randomly initialize  $N$  ideas and evaluate their fitness
(3)   Initialize  $m$  cluster centers ( $m < N$ )
(4)   while (stopping condition not met)
(5)     Cluster  $N$  ideas into  $m$  clusters           %clustering operation
(6)     Record the best idea in each cluster as the cluster center
(7)     for ( $i = 1$  to  $N$ )                           %creating operation
(8)       Execute Algorithm 1: differential evolution operator()
(9)       Create  $X_{new1}$  using  $X_{new}$  according formula (9)
(10)      Accept  $X_{new1}$  if  $f(X_{new1})$  is better than  $f(X_i)$ 
(11)    endfor
(12)  endwhile
(13) end

```

ALGORITHM 2: Pseudocode of the BSODE().

TABLE 1: Numerical benchmark function definition (CEC2005 $F1-F14$).

Number	Function name	Type	Low range	Up range	Optima
$F1$	Shifted sphere	Unimodal	-100	100	0
$F2$	Shifted Schwefel	Unimodal	-100	100	0
$F3$	Shifted rotated high conditioned elliptic	Unimodal	-100	100	0
$F4$	Shifted Schwefel's problem 1.2 with noise	Unimodal	-100	100	0
$F5$	Schwefel's problem 2.6	Unimodal	-100	100	0
$F6$	Shifted Rosenbrock	Multimodal	-100	100	0
$F7$	Shifted rotated Griewank	Multimodal	0	600	0
$F8$	Shifted rotated Ackley	Multimodal	-32	32	0
$F9$	Shifted Rastrigin	Multimodal	-5	5	0
$F10$	Shifted rotated Rastrigin	Multimodal	-5	5	0
$F11$	Shifted rotated Weierstrass	Multimodal	-0.5	0.5	0
$F12$	Schwefel's problem 2.13	Multimodal	-100	100	0
$F13$	Expanded extended Griewank and Rosenbrock	Expanded	-3	1	0
$F14$	Expanded rotated extended Scaffes	Expanded	-100	100	0

CPU, 4 GB memory. The operating system is Windows 7 with MATLAB 8.0 (R2012b). For the purpose of reducing statistical errors, each function is independently run 25 times.

4.1. Benchmark Functions. We choose 14 widely known rotated or shifted benchmark functions used in CEC2005, which is given in Table 1 [19]. All functions are tested on 30 dimensions. Results obtained are compared with some other variants of DE algorithms defined in the next subsection. The searching range and true optima for all functions are also given in Table 1. Among 14 benchmark functions, $F1$ to $F5$ are shifted unimodal functions, $F6$ to $F12$ are shifted or rotated multimodal functions, and $F13$ to $F14$ are expanded shifted or rotated functions.

4.2. Parameter Settings of Comparative Algorithms. Because the BSODE algorithm combines the original BSO and the differential evolution strategy, we specially compare the performance of BSODE with the DE, the original BSO, and the PSO. Besides, we also compare the performance of BSODE with other variants of DE. We choose two DE variants, that is, CoDE [21] and SaDE [22]. To eliminate influences of statistical errors, each problem function is independently

TABLE 2: Algorithms parameter setting.

Number	Algorithm	Parameter setting
1	BSO-DE	$M = 5$, $p_one = 0.8$, $p_one_center = 0.4$, and $p_two_center = 0.5$
2	BSO	$M = 5$, $p_replace = 0.2$, $p_one = 0.8$, $p_one_center = 0.4$, $p_two_center = 0.5$, and $K = 20$
3	PSO	$\omega: 0.9 \sim 0.4$, $c1 = c2 = 1.49445$, and global version
4	DE	$F = 0.5$ and $CR = 0.9$
5	CoDE	$F = [1.0 \ 1.0 \ 0.8]$ and $CR = [0.1 \ 0.9 \ 0.2]$
6	SaDE	$F_i = N_i(0.5, 0.3)$, $CR_i = N_i(0.5, 0.1)$, and $p1 = p2 = 0.5$

run 25 times which is a prescribed evaluation criterion in CEC2005 [19]. For all approaches, the population size is set to 30. The same stopping criterion is used in all algorithms, that is, reaching certain number of iterations or function evaluations (FEs). In our experiments, we have run all

TABLE 3: The results of solution accuracy for new step size.

Function	BSODE		BSO		BSO-newstep		BSODE-nonewstep
F1	4.12E - 20 ± 5.77E - 20	+	2.65E - 03 ± 3.03E - 03	+	2.60E - 16 ± 4.09E - 16	+	1.31E - 11 ± 6.35E - 11
F2	5.19E - 07 ± 3.87E - 07	+	2.76E + 00 ± 8.16E - 01	+	4.08E - 05 ± 1.34E - 05	+	1.58E + 00 ± 5.86E - 01
F3	5.73E + 05 ± 2.37E + 05	+	1.33E + 06 ± 3.58E + 05	+	8.39E + 05 ± 3.03E + 05	+	1.67E + 06 ± 5.61E + 05
F4	2.09E - 01 ± 1.73E - 01	+	1.29E + 04 ± 9.06E + 03	=	1.29E - 01 ± 1.15E - 01	+	1.98E + 03 ± 4.05E + 03
F5	2.92E + 04 ± 3.48E + 02	=	3.12E + 04 ± 6.02E + 02	+	3.02E + 04 ± 5.01E + 02	=	2.93E + 04 ± 2.90E + 02
F6	1.55E + 03 ± 2.64E + 03	=	1.52E + 03 ± 2.71E + 03	+	1.86E + 03 ± 2.66E + 03	+	1.77E + 03 ± 3.78E + 03
F7	8.36E - 03 ± 1.05E - 02	+	5.35E + 03 ± 2.32E + 02	=	5.07E + 03 ± 1.18E + 02	=	5.45E - 02 ± 3.19E - 02
F8	2.02E + 01 ± 6.69E - 02	=	2.05E + 01 ± 1.17E - 01	=	2.02E + 01 ± 2.74E - 02	=	2.04E + 01 ± 1.10E - 01
F9	4.26E + 01 ± 1.37E + 01	+	4.97E + 01 ± 1.02E + 01	-	2.45E + 01 ± 7.20E + 00	+	8.67E + 01 ± 1.23E + 01
F10	3.47E + 01 ± 1.21E + 01	+	4.59E + 01 ± 1.19E + 01	=	3.49E + 01 ± 5.54E + 00	+	8.50E + 01 ± 2.19E + 01
F11	1.40E + 01 ± 2.00E + 00	+	2.43E + 01 ± 2.58E + 00	+	1.60E + 01 ± 1.98E + 00	+	2.69E + 01 ± 2.72E + 00
F12	7.34E + 03 ± 7.32E + 03	+	3.16E + 04 ± 1.70E + 04	=	6.81E + 03 ± 3.50E + 03	+	3.61E + 04 ± 2.33E + 04
F13	3.88E + 00 ± 8.18E - 01	+	5.56E + 00 ± 1.35E + 00	+	4.70E + 00 ± 4.33E - 01	=	5.30E + 00 ± 2.05E + 00
F14	1.29E + 01 ± 4.23E - 01	=	1.32E + 01 ± 3.92E - 01	=	1.26E + 01 ± 4.44E - 01	=	1.33E + 01 ± 3.11E - 01
+/- = /-			10/4/0		7/6/1		9/5/0

algorithms on the benchmark functions using the same FEs of $10000 * D$ for a fair comparison, where D is the size of the problem dimension. The parameter settings of all the algorithms are given in Table 2.

4.3. The Contribution of the New Step Size Method to BSODE.

To verify the rationality and effectivity of the new step size in BSODE and to fully understand the effect of the new step size method, herein, we investigate the contribution of the new step size to BSODE. Considering the consistency of the paper, we compare all 14 benchmark functions with 25 independent runs. We test the BSO, BSO with new step size (BSO-newstep), BSODE, and the BSODE with the original adjusting factor, that is, without new step size (BSODE-nonewstep). The experimental results are presented in Table 3.

Table 3 shows that the BSO-newstep (BSO with new step size) has better performance than the original BSO and BSODE has better performance than the BSODE-nonewstep (BSODE without new step size), and it can be concluded that the new step size plays a significant role in BSODE algorithm. This is in good agreement with our analysis in Section 3.2 that the control of the new step size can balance exploration and exploitation at different searching generations in Section 3.2.

4.4. Comparison of Results

4.4.1. Comparisons on Solution Accuracy. The results of solution accuracy are given in Table 4 in terms of the mean optimum solution and the standard deviation of the solutions obtained in the 25 independent runs by each algorithm over 300,000 FEs on 14 benchmark functions. In all experiments, the dimensions of all problems are 30. In each row of the table, the mean values are listed in the first part, and the standard deviations are listed in the last part, and the two parts are divided with a symbol “±.” The best results among the algorithms are displayed in bold.

From Table 4, it can be observed that the mean value and the standard deviation value of the original BSO algorithm performs well only for function F6 compared with

the BSODE. The BSODE algorithm performs better than the other algorithms for 4 functions F4, F7, F8, and F10, especially for F4, and the mean value and the standard deviation value are far better than the other five algorithms. PSO outperforms the other algorithms in terms of solution accuracy only for function F12, and DE outperforms the other algorithms in terms of solution accuracy only for functions F3 and F5. Although CoDE performs better for 5 functions F1, F6, F9, F11, and F14 and SaDE performs better for 3 functions F1, F2, and F13, we can conclude that the BSODE algorithm remains still a good performance of the solution accuracy for complex shifted and rotated benchmark functions.

4.4.2. The Comparison Results of Convergence Speed. Figure 5 shows the convergence graphs of the compared 6 algorithms in terms of the mean fitness values of 25 runs. Due to page limitation, only graphs for 3 functions F1, F7, and F14 which include shifted, shifted rotated, and expanded rotated extended functions are shown. From Figure 5, we can observe that BSODE has fast convergence speed compared to the original BSO or similar convergence speed compared to the other four algorithms.

4.4.3. The Comparison Results of t -Tests. To statistically compare BSODE with the other five algorithms, t -test [23, 24] is carried out to show the differences between BSODE and the other five algorithms. The t values on every function by two-tailed test with a significance level of 0.05 between BSODE and other algorithms are given in Table 5. Better, same, and worse give the number of experiments that BSODE performs significantly better than, almost the same as, and significantly worse than the compared algorithm under the same conditions, respectively.

Table 5 gives information about the t value, p value, and h value of 25 independent runs of 6 algorithms over 300,000 FEs on 14 test functions. The best results among the algorithms are shown in bold.

TABLE 4: The results of solution accuracy.

Function	BSODE	BSO	PSO	DE	CoDE	SaDE
F1	4.12E-20 ± 5.77E-20	2.65E-03 ± 3.03E-03	4.67E+03 ± 2.86E+03	5.46E-01 ± 2.59E+00	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00
F2	5.19E-07 ± 3.87E-07	2.76E+00 ± 8.16E-01	6.51E+03 ± 8.15E+03	1.19E-01 ± 5.88E-01	9.11E-06 ± 1.11E-05	5.55E-10 ± 6.99E-10
F3	5.73E+05 ± 2.37E+05	1.33E+06 ± 3.58E+05	2.29E+07 ± 4.93E+07	3.40E+05 ± 1.69E+05	3.65E+05 ± 1.60E+05	3.67E+05 ± 1.82E+05
F4	2.09E-01 ± 1.73E-01	1.29E+04 ± 9.06E+03	4.56E+03 ± 6.54E+03	1.25E+02 ± 5.87E+02	2.01E+01 ± 5.61E+01	4.42E+01 ± 1.25E+02
F5	2.92E+04 ± 3.48E+02	3.12E+04 ± 6.02E+02	3.43E+04 ± 2.28E+03	2.90E+04 ± 2.17E-01	2.93E+04 ± 2.21E-04	2.93E+04 ± 1.09E+02
F6	1.55E+03 ± 2.64E+03	1.52E+03 ± 2.71E+03	8.15E+08 ± 9.30E+08	5.43E+06 ± 1.51E+07	3.45E+00 ± 3.67E+00	3.07E+01 ± 3.24E+01
F7	8.36E-03 ± 1.05E-02	5.35E+03 ± 2.32E+02	5.36E+03 ± 5.34E+02	4.70E+03 ± 9.47E-13	4.70E+03 ± 1.30E-12	4.70E+03 ± 4.15E-13
F8	2.02E+01 ± 6.69E-02	2.05E+01 ± 1.17E-01	2.09E+01 ± 6.56E-02	2.09E+01 ± 7.37E-02	2.05E+01 ± 3.25E-01	2.09E+01 ± 5.36E-02
F9	4.26E+01 ± 1.37E+01	4.97E+01 ± 1.02E+01	6.90E+01 ± 2.28E+01	2.18E+01 ± 5.63E+00	1.34E-12 ± 3.35E-12	1.95E+00 ± 1.27E+00
F10	3.47E+01 ± 1.21E+01	4.59E+01 ± 1.19E+01	1.26E+02 ± 3.82E+01	4.28E+01 ± 1.59E+01	3.98E+01 ± 1.55E+01	4.75E+01 ± 1.13E+01
F11	1.40E+01 ± 2.00E+00	2.43E+01 ± 2.58E+00	2.02E+01 ± 2.59E+00	2.30E+01 ± 6.15E+00	1.05E+01 ± 4.09E+00	1.65E+01 ± 2.85E+00
F12	7.34E+03 ± 7.32E+03	3.16E+04 ± 1.70E+04	4.83E+03 ± 5.14E+03	2.08E+05 ± 8.19E+04	1.31E+05 ± 2.54E+04	9.67E+04 ± 1.47E+04
F13	3.88E+00 ± 8.18E-01	5.56E+00 ± 1.35E+00	3.65E+00 ± 1.14E+00	3.24E+00 ± 5.77E-01	4.02E+00 ± 1.22E+00	2.76E+00 ± 3.60E-01
F14	1.29E+01 ± 4.23E-01	1.32E+01 ± 3.92E-01	1.21E+01 ± 5.27E-01	1.30E+01 ± 2.66E-01	1.25E+01 ± 3.82E-01	1.27E+01 ± 1.83E-01

FIGURE 5: Convergence performance of the 6 algorithms on parts of functions.

For example, comparing BSODE and DE, the former significantly outperformed the latter in 7 functions (*F2*, *F4*, *F7*, *F8*, *F10*, *F11*, and *F12*), does the same as the latter in 3 functions (*F6*, *F13*, and *F14*), and does worse in 4 function (*F1*, *F3*, *F5*, and *F9*), yielding a “General Merit” value $7 - 4 = 3$, indicating that the BSODE generally outperforms the DE. Although it performed slightly weaker in some functions, the BSODE in general offered much improved performance than other compared algorithms, as confirmed in Table 5.

5. Applications of ANN Using BSODE

Artificial neural network (ANN) [25] simulates the learning pattern of human brain for solving many difficult problems

in the field of applications such as prediction of principal ground-motion parameters [26] and prediction of peak ground acceleration [27]. ANN is a flexible mathematical scheme which is capable of identifying complex nonlinear relationships between input data and output data. The task to train ANN needs vigorous optimization techniques because the search space is highly dimensional and multimodal which suffers from noises and data in general. The usually used training method is Back-Propagation (BP) algorithm [28]. However, BP algorithm has some potential problems. On one hand, for the complex pattern classification problems [29], BP algorithm may lightly get trapped into local minima. On the other hand, the training time of the BP algorithm is very long for complex nonlinear problems.

TABLE 5: Comparisons between BSODE and other algorithms on t -tests.

BSODE versus		BSO	PSO	DE	CoDE	SaDE
F1	t value	6.18	11.53	1.49	-6.49	-6.49
	p value	0.00007	0	0.29712	0.00003	0.00003
	h value	1	1	0	1	1
F2	t value	23.91	5.65	1.43	5.69	-9.07
	p value	0	0.00022	0.31701	0.002	0
	h value	1	1	1	1	1
F3	t value	10.64	3.20	-7.38	-6.92	6.62
	p value	0	0.02842	0	0	0.00002
	h value	1	1	1	1	0
F4	t value	10.04	4.93	1.50	2.51	2.50
	p value	0	0.00105	0.0291	0.03137	0.0233
	h value	1	1	1	1	1
F5	t value	21.75	15.77	-7.99	7.99	2.49
	p value	0	0	0	0	0.08411
	h value	1	1	1	0	0
F6	t value	-0.32	6.20	2.54	-4.64	-4.56
	p value	0.08210	0.00006	0.0790	0.00194	0.00227
	h value	0	1	0	1	1
F7	t value	162.80	70.98	3165211.38	3165211.38	3165211.38
	p value	0	0	0	0	0
	h value	1	1	1	1	1
F8	t value	11.70	43.94	43.35	6.73	50.07
	p value	0	0	0	0.00002	0
	h value	1	1	1	1	1
F9	t value	8.59	10.28	-8.30	-28.45	-26.50
	p value	0	0	0	0	0
	h value	1	1	1	1	1
F10	t value	5.10	16.34	3.05	1.96	5.98
	p value	0.00074	0	0.03618	0.01729	0.0001
	h value	1	1	1	1	1
F11	t value	20.95	12.61	9.62	-5.38	4.73
	p value	0	0	0	0.0004	0.00162
	h value	1	1	1	1	1
F12	t value	4.61	-2.53	16.25	24.62	21.93
	p value	0.00205	0.08008	0	0	0
	h value	1	0	1	1	1
F13	t value	10.15	1.45	-1.02	3.25	-5.51
	p value	0	0.31127	0.47521	0.02607	0.00031
	h value	1	0	0	1	1
F14	t value	4.35	-7.05	2.73	3.53	1.01
	p value	0.00344	0.00001	0.05952	0.01618	0.47943
	h value	1	1	0	0	0
Better		13	11	7	7	6
Same		0	2	3	2	3
Worse		1	1	4	5	5

In recent decades, in order to deal with the complicated training problem of the ANN, many metaheuristic optimization algorithms, such as Simulated Annealing (SA) [26, 27], GA [30], and PSO [31], have been utilized to optimize the

weights and biases of ANN. Besides, the recently proposed EvoNN [32, 33] algorithm, which utilizes the multiobjective optimization technique in the training process of a feed-forward neural network, ensures correct neural training by

FIGURE 6: BSOE-based ANN training model.

working out a Pareto tradeoff between the accuracy of the training and the complexity of the network. In this section, we use our BSOE algorithm to adjust connection weights and biases of ANN. To validate the usefulness and the efficiency of our BSOE algorithm, two types of applications including time series prediction and System Identification are illustrated.

5.1. Training ANN with BSOE Algorithm. The structure of training ANN using our BSOE algorithm is a three-layer feed-forward network. The basic structure of the proposed scheme is given in Figure 6.

In our experiments, activation function used in the input layer is a linear combination of input variables, activation function used in the hidden layer is the sigma function, and activation function used in the output layer is a linear combination of output variables of hidden layer. In training ANN, the aim is to find a set of weights and biases with the smallest error value. Herein, the objective function is the mean sum of squared errors (MSE) which is given as follows:

$$\text{MSE} = \frac{1}{Q * K} \sum_{i=1}^Q \sum_{j=1}^K \frac{1}{2} (d_{ij} - y_{ij})^2, \quad (11)$$

where Q is the number of training data sets, K is the number of output units, d_{ij} is desired output, and y_{ij} is output inferred from ANN.

5.2. Time Series Prediction of Lorenz Chaotic System. The first application is the time series prediction of Lorenz chaotic system described by the following equation [34]:

$$\dot{x}(t) = a(y(t) - x(t)) \quad (12a)$$

$$\dot{y}(t) = bx(t) - x(t)z(t) - y(t) \quad (12b)$$

$$\dot{z}(t) = x(t)y(t) - cz(t), \quad (12c)$$

where a , b , and c are parameters of the Lorenz chaotic system. The system is chaotic when $a = 10$, $b = 28$, and $c = 8/3$. The differential equations were solved numerically using 4th order Runge-Kutta integration with a step size 0.01 and initial values $x(0) = 0$, $y(0) = -1$, and $z(0) = 0$. The x -coordinate of the Lorenz time series system is chosen for prediction. The goal is to construct the prediction model of chaotic time series according to the following formula:

$$y(t+1) = f(y(t-18), y(t-12), \dots, y(t-6), y(t)). \quad (13)$$

In order to train the ANN, we choose 10000 data sets according to the real model, where the first 8000 data sets are discarded; the rest normalized 2000 data sets are considered as experiment data. The former 1500 points are selected as the training data points and the rest 500 points are selected as the testing data points to test the validity of the model.

In this experiment, the parameter settings of the six algorithms are the same as those in Section 4.2, and the maximal generation used as ended condition of algorithm is 100 for BSOE, BSO, PSO, DE, CoDE, and SaDE. The results are shown in Table 6 in terms of the mean optimum solution, the standard deviation of the solutions, and run time independent runs by each algorithm.

As is seen in Table 6, SaDE has the worst MSE in both training and testing cases. BSOE performs better than BSO, PSO, DE, CoDE, and SaDE in the mean of the MSE. The model trained by six algorithms follows the chaotic behavior of the Lorenz chaotic time series which is demonstrated in Figure 7. It can be concluded that BSOE is the most effective learning algorithm for Lorenz chaotic time series even though BSOE needs more running time than PSO algorithm.

5.3. Nonlinear System Identification. The second application is to build a three-layer feed-forward ANN with three input units, five hidden units, and one output unit. The ANN is used to model the curve of a nonlinear function which is given by the following equation:

$$y(t+1) = \frac{y(t)y(t-1)[y(t)+2.5]}{1+y^2(t)+y^2(t-1)} + u(t) \quad (14)$$

$$u(t) = \sin\left(\frac{2\pi t}{25}\right).$$

The simulation target is to construct the approximation model of the nonlinear function according to the following formula:

$$y(t+1) = f(u(t), y(t-1), y(t)). \quad (15)$$

To train the ANN, we choose 200 pairs of data from the real model. In addition, to further evaluate the performance of BSOE algorithm, we add 20 db additive white Gaussian noise into the 200 pairs of data.

In this experiment, population size of each algorithm is 30, and the maximal generation used as termination

TABLE 6: Comparisons of MSE on Lorenz chaotic time series over 25 runs.

Performance	BSODE	BSO	PSO	DE	CoDE	SaDE
Training error	3.33E-03 ± 2.02E-03	3.81E-03 ± 1.70E-03(+)	4.01E-03 ± 2.30E-03(+)	1.03E-02 ± 5.95E-03(+)	4.84E-03 ± 1.97E-03(+)	9.97E-02 ± 1.11E-01(+)
Testing error	3.48E-03 ± 2.15E-02	3.99E-03 ± 1.78E-03(+)	4.21E-03 ± 3.20E-03(+)	1.06E-02 ± 6.08E-03(+)	4.99E-03 ± 2.07E-03(+)	9.74E-02 ± 1.13E-01(+)
Run time(s)	7.39E+01 ± 8.30E-01	7.41E+01 ± 6.60E-01(+)	7.11E+01 ± 3.07E+00 (-)	7.35E+01 ± 4.85E-01(-)	2.13E+02 ± 3.05E+00(+)	7.14E+01 ± 4.99E-01(-)

(a) The prediction results using BSODE

(b) The prediction results using BSO

(c) The prediction results using PSO

(d) The prediction results using DE

(e) The prediction results using CoDE

(f) The prediction results using SaDE

FIGURE 7: Comparison of the performance curves for Lorenz chaotic time series prediction.

TABLE 7: Comparisons between BSODe and other algorithms on MSE over 25 runs.

App.	BSODE	BSO	PSO	DE	CoDE	SaDE
Without noise	$6.69E - 03 \pm 3.89E - 03$	$7.78E - 03 \pm 3.87E - 03(+)$	$2.90E - 03 \pm 1.69E - 03(-)$	$9.89E - 03 \pm 1.69E - 02(+)$	$3.69E - 03 \pm 6.36E - 03(-)$	$1.68E - 03 \pm 1.41E - 03(-)$
With noise	$3.20E - 02 \pm 3.55E - 03$	$4.08E - 02 \pm 6.24E - 03(+)$	$3.28E - 02 \pm 3.66E - 03(+)$	$3.27E - 02 \pm 1.55E - 02(+)$	$3.63E - 02 \pm 4.14E - 03(+)$	$3.51E - 02 \pm 4.40E - 03(+)$

(a) System Identification without noise using BSODE

(b) System Identification with noise using BSODE

(c) System Identification without noise using BSO

(d) System Identification with noise using BSO

(e) System Identification without noise using PSO

(f) System Identification with noise using PSO

FIGURE 8: Continued.

(g) System Identification without noise using DE

(h) System Identification with noise using DE

(i) System Identification without noise using CoDE

(j) System Identification with noise using CoDE

(k) System Identification without noise using SaDE

(l) System Identification with noise using SaDE

FIGURE 8: Comparison of the performance curves for System Identification.

condition of each algorithm is 1000 for BSO, PSO, DE, CoDE, and SaDE. The parameter settings of the six algorithms are also the same as in Section 4.2. The results are given in Table 7 in terms of the mean MSE and the standard deviation obtained in the 25 independent runs for six algorithms.

From Table 7, in case without noise, we observed that BSO performs worse than SaDE, CoDE, and PSO, but BSO performs better than BSO and DE in terms of the mean MSE. However, in case of 20 db additive white Gaussian noise, BSO performs better than all other five algorithms. Figure 8 shows that the output of the identified system tracks the target output for three algorithms. We can conclude that BSO has the best performance in case without noise and a better performance compared with BSO and DE in building the approximation model.

6. Conclusion

In this paper, an improved BSO with differential evolution strategy and new step size method named BSO is proposed for solving complex optimization problems and training artificial neural networks. Experiments on the 14 widely used benchmark functions of CEC 2005 and the two real-world applications of artificial neural network are carried out to compare the proposed BSO with five state-of-the-art algorithms in this paper. From these results, we can draw the conclusion that the BSO significantly improves the performance of the original BSO. Moreover, the BSO gives good performance on the training of artificial neural networks compared with other algorithms. We expect that BSO will be used to solve more real-world global optimization problems. Further work includes research into neighbor search to make the algorithm more efficient. In the near future, we also plan to apply the algorithm to large scale optimization domains.

Conflict of Interests

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

Acknowledgments

This research was partially supported by National Natural Science Foundation of China (under Grants nos. 61272283, 61403304, 61304082, and 61273367), the Project of Department of Education Science Research of Shaanxi Province, China (under Grants nos. 14JK1360 and 14JK1511), and the Project of Science Research Plan in Xi'an of China (nos. CXY1437 (4) and CXY1439 (5)).

References

- [1] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.
- [2] L. J. Fogel, "Evolutionary programming in perspective: the top-down view," in *Computational Intelligence: Imitating Life*, pp. 135–146, IEEE Press, 1994.
- [3] I. Rechenberg, "Evolution strategy," in *Computational Intelligence: Imitating Life*, J. M. Zurada, R. J. Marks II, and C. J. Robinson, Eds., IEEE Press, Piscataway, NJ, USA, 1994.
- [4] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*, MIT Press, Cambridge, Mass, USA, 1992.
- [5] R. Storn and K. Price, "Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.
- [6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.
- [7] Y. H. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, Anchorage, Alaska, USA, May 1998.
- [8] Y. H. Shi, "Brain storm optimization algorithm," in *Advances in Swarm Intelligence: Second International Conference, ICSI 2011, Chongqing, China, June 12–15, 2011, Proceedings, Part I*, vol. 6728 of *Lecture Notes in Computer Science*, pp. 303–309, Springer, Berlin, Germany, 2011.
- [9] Y. Shi, "An optimization algorithm based on brainstorming process," *International Journal of Swarm Intelligence Research*, vol. 2, no. 4, pp. 35–62, 2011.
- [10] C. Sun, H. Duan, and Y. Shi, "Optimal satellite formation reconfiguration based on closed-loop brain storm optimization," *IEEE Computational Intelligence Magazine*, vol. 8, no. 4, pp. 39–51, 2013.
- [11] H. B. Duan, S. T. Li, and Y. H. Shi, "Predator-prey based brain storm optimization for DC brushless motor," *IEEE Transactions on Magnetics*, vol. 49, no. 10, pp. 5336–5340, 2013.
- [12] H. T. Jadhav, U. Sharma, J. Patel, and R. Roy, "Brain storm optimization algorithm based economic dispatch considering wind power," in *Proceedings of the IEEE International Conference on Power and Energy (PECon '12)*, pp. 588–593, December 2012.
- [13] Y. Shi, J. Xue, and Y. Wu, "Multi-objective optimization based on brain storm optimization algorithm," *International Journal of Swarm Intelligence Research*, vol. 4, no. 3, pp. 1–21, 2013.
- [14] J. Q. Xue, Y. L. Wui, Y. H. Shi, and S. Cheng, "Brain storm optimization algorithm for multi-objective optimization problems," in *Advances in Swarm Intelligence: Third International Conference, ICSI 2012, Shenzhen, China, June 17–20, 2012 Proceedings, Part I*, vol. 7331 of *Lecture Notes in Computer Science*, pp. 513–519, Springer, Berlin, Germany, 2012.
- [15] Z. H. Zhan, J. Zhang, Y. H. Shi et al., "Parameter investigation in brain storm optimization," in *Proceedings of the IEEE Symposium Series on Computational Intelligence*, pp. 103–110, April 2013.
- [16] Z. H. Zhan, J. Zhang, Y. H. Shi, and H. L. Liu, "A modified brain storm optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1–8, Brisbane, Australia, June 2012.
- [17] S. Cheng, Y. H. Shi, Q. D. Qin, T. O. Ting, and R. Bai, "Maintaining population diversity in brain storm optimization algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 3230–3237, July 2014.
- [18] Y. T. Yang, Y. H. Shi, and S. R. Xia, "Advanced discussion mechanism-based brain storm optimization algorithm," *Soft Computing*, 2014.
- [19] P. N. Suganthan, N. Hansen, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC 2005 special

- session on real-parameter optimization,” Tech. Rep., Nanyang Technological University, Singapore, 2005.
- [20] A. F. Osborn, *Applied Imagination: Principles and Procedures of Creative Problem Solving*, Charles Scribner’s Son, New York, NY, USA, 1963.
 - [21] Y. Wang, Z. Cai, and Q. Zhang, “Differential evolution with composite trial vector generation strategies and control parameters,” *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 55–66, 2011.
 - [22] A. K. Qin, V. L. Huang, and P. N. Suganthan, “Differential evolution algorithm with strategy adaptation for global numerical optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.
 - [23] J. D. Gibbons and S. Chakraborti, *Nonparametric Statistical Inference*, Chapman & Hall, CRC Press, Taylor & Francis, Boca Raton, Fla, USA, 5th edition, 2011.
 - [24] S. García, D. Molina, M. Lozano, and F. Herrera, “A study on the use of non-parametric tests for analyzing the evolutionary algorithms’ behaviour: a case study on the CEC’2005 Special Session on Real Parameter Optimization,” *Journal of Heuristics*, vol. 15, no. 6, pp. 617–644, 2009.
 - [25] K. Mehrotra, C. K. Mohan, and S. Ranka, *Artificial Neural Networks*, The MIT Press, 1997.
 - [26] A. H. Alavi and A. H. Gandomi, “Prediction of principal ground-motion parameters using a hybrid method coupling artificial neural networks and simulated annealing,” *Computers and Structures*, vol. 89, no. 23–24, pp. 2176–2194, 2011.
 - [27] M. Gandomia, M. Soltanpoura, M. R. Zolfaghari, and A. H. Gandomi, “Prediction of peak ground acceleration of Iran’s tectonic regions using a hybrid soft computing technique,” *Geoscience Frontiers*, 2014.
 - [28] M. Negnevitsky, *Artificial Intelligence: A Guide to Intelligent Systems*, Pearson Education, 2005.
 - [29] M. Gori and A. Tesi, “On the problem of local minima in backpropagation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 1, pp. 76–86, 1992.
 - [30] D. J. Montana and L. Davis, “Training feedforward neural networks using genetic algorithms,” in *Proceedings of the 11th International Joint Conference on Artificial Intelligence (IJCAI ’89)*, pp. 762–767, Detroit, Mich, USA, August 1989.
 - [31] J. Yu, S. Wang, and L. Xi, “Evolving artificial neural networks using an improved PSO and DPSO,” *Neurocomputing*, vol. 71, no. 4–6, pp. 1054–1060, 2008.
 - [32] F. Pettersson, N. Chakraborti, and H. Saxén, “A genetic algorithms based multi-objective neural net applied to noisy blast furnace data,” *Applied Soft Computing Journal*, vol. 7, no. 1, pp. 387–397, 2007.
 - [33] R. Jha, F. Pettersson, G. S. Dulikravich, H. Saxen, and N. Chakraborti, “Evolutionary design of nickel-based superalloys using data-driven genetic algorithms and related strategies,” *Materials and Manufacturing Processes*, vol. 30, no. 4, pp. 488–510, 2014.
 - [34] L. Wang, F. Zou, X. Hei et al., “A hybridization of teaching–learning-based optimization and differential evolution for chaotic time series prediction,” *Neural Computing and Applications*, vol. 25, no. 6, pp. 1407–1422, 2014.