Hindawi Publishing Corporation
Computational Intelligence and Neuroscience
Volume 2015, Article ID 292576, 15 pages
http://dx.doi.org/10.1155/2015/292576

Research Article

Nonlinear Inertia Weighted Teaching-Learning-Based Optimization for Solving Global Optimization Problem

Zong-Sheng Wu,[1] Wei-Ping Fu,[1] and Ru Xue[2]

[1] *School of Mechanical and Precision Instrumental Engineering, Xi'an University of Technology, Xi'an, Shaanxi 710048, China*
[2] *School of Information Engineering, Tibet University for Nationalities, Xianyang, Shaanxi 712082, China*

Correspondence should be addressed to Zong-Sheng Wu; wuzs2005@163.com

Received 28 June 2015; Revised 11 August 2015; Accepted 17 August 2015

Academic Editor: Michael Schmuker

Teaching-learning-based optimization (TLBO) algorithm is proposed in recent years that simulates the teaching-learning phenomenon of a classroom to effectively solve global optimization of multidimensional, linear, and nonlinear problems over continuous spaces. In this paper, an improved teaching-learning-based optimization algorithm is presented, which is called nonlinear inertia weighted teaching-learning-based optimization (NIWTLBO) algorithm. This algorithm introduces a nonlinear inertia weighted factor into the basic TLBO to control the memory rate of learners and uses a dynamic inertia weighted factor to replace the original random number in teacher phase and learner phase. The proposed algorithm is tested on a number of benchmark functions, and its performance comparisons are provided against the basic TLBO and some other well-known optimization algorithms. The experiment results show that the proposed algorithm has a faster convergence rate and better performance than the basic TLBO and some other algorithms as well.

1. Introduction

Most of the swarm intelligent optimization studies and applications have been focused on nature-inspired algorithms. Numerous population-based and nature-inspired optimization algorithms have been presented, such as the Ant Colony Optimization (ACO), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), and Differential Evolution (DE). These optimization algorithms are based on different natural phenomena. ACO works based on the behavior of ant colony searching foods from the source to a destination [1, 2]. GA applies the theory of Darwin based on the survival of the fittest to the optimization problems [3, 4]. PSO emulates the collaborative behavior of birds flocking and fish schooling in searching for foods [5–7]. ABC uses the foraging behavior of a honey bee [8–10]. DE derived from the Genetic Algorithm, which is an efficient global optimizer in the continuous search domain [11, 12]. These algorithms have been applied to many engineering optimization problems and proven effective in solving specific types of problems. However, various algorithms have their own advantages and disadvantages in solving diverse problems. Generally, a good optimization algorithm should possess the three essential conditions. First, the algorithm has the ability of obtaining the true global optima value. Second, the convergence speed of the algorithms should be fast. Third, the program should have a minimum of control parameters so that it will be easy to use. If an optimization algorithm meets the above three conditions at the same time, it would be a great algorithm. Some optimization techniques often achieve global optima results but at the cost of the convergence speed. Those algorithms tend to focus on the quality of computational results rather than the convergence speed. However, the higher calculation accuracy and faster convergence speed are the ultimate aim in the practical applications.

Recently, Rao et al. [13, 14] proposed a teaching-learning-based optimization (TLBO) algorithm, inspired by the phenomenon of teaching and learning in a class. The TLBO requires only the common control parameters like population size and numbers of generation and that does not require any algorithm-specific control parameters; that is, it is a parameter-less algorithm [15]. Thus, there is no burden of tuning control parameters in the TLBO algorithm. Hence, the TLBO algorithm is simpler and more effective and involves relatively less computational cost. What it is more important is that the TLBO algorithm has the ability to achieve better results at comparatively faster convergence speed to other algorithms mentioned above. Therefore, the TLBO algorithm has been successfully applied in diverse optimization fields such as mechanical engineering, task scheduling, production planning and control, and vehicle-routing problems in transportation [16–20]. Similar to other swarm intelligent optimization algorithms, the basic TLBO can be improved further and further. In order to improve the performance of TLBO, several variants of the TLBO have been proposed. Rao and Patel presented an elitist TLBO (ETLBO) algorithm [15] to solve complex constrained optimization problems and used a modified version of TLBO algorithm [17] to solve the multiobjective optimization problem of heat exchangers. Sultana and Roy [19] proposed a quasioppositional teaching-learning-based optimization (QOTLBO) methodology in order to find the optimal location of the distributed generator to simultaneously optimize power loss, voltage stability index, and voltage deviation of radial distribution network. Ghasemi et al. [20] used Lévy mutation strategy based on TLBO for optimal settings of optimal power flow problem control variables. Furthermore, some improved TLBO algorithms have been proposed to solve the global function optimization problem [21–24] and the multiobjective optimization problem [17, 25, 26].

In this paper, we propose a novel improved TLBO, which is called nonlinear inertia weighted TLBO (NIWTLBO). A nonlinear inertia weighted factor is introduced into the basic TLBO to control the memory rate of learners, and another dynamic inertia weighted factor is used to replace the original random number in teacher phase and learner phase. So, as a result, the NIWTLBO has faster convergence speed and higher calculation accuracy for most of these optimization problems than the basic TLBO. The performance of NIWTLBO for solving global function optimization problems is compared with basic TLBO and other optimization algorithms. The analysis results show that the proposed algorithm outperforms most of the other algorithms investigated in this paper.

The rest of this paper is organized as follows. Section 2 describes the basic TLBO algorithm in detail. In Section 3, the proposed NIWTLBO algorithm will be introduced. And Section 4 provides numerical experiments and results demonstrating the performance of NIWTLBO in comparison with other optimization algorithms. Finally, our conclusions are mentioned in Section 5.

2. Teaching-Learning-Based Optimization

The basic TLBO algorithm mainly consists of two parts, namely, the teacher phase and the learner phase. In teacher phase, the students can learn from the teacher to make their knowledge level closer to the teacher's. In learner phase, the students can learn from the interaction of other individuals to increase their knowledge. In the TLBO algorithm, a group of learners is considered as a population. Each learner is analogous to an individual of the evolutionary algorithm. The different subjects offered to the learners are considered as design variables of the optimization problem. A learner's result is analogous to the fitness value of the objective function for optimization problems. The best learner (i.e., the best solution in the entire population) is considered as the teacher. The best solution is the best value of the objective function of the given optimization problem. The design variables are the input parameters of the objective function.

The process of basic TLBO algorithm is described below.

2.1. Initialization. The notations used in TLBO are described as follows:

NP is number of learners in a class (i.e., population size).

D is number of subjects offered to the learners (i.e., dimensions of design variables).

MAXITER is maximum number of allowable iterations.

$X_{i,k} = (X_{i,k,1}, X_{i,k,2}, \ldots, X_{i,k,j}, \ldots, X_{i,k,D})$ denotes a learner in class (i.e., the individual in the population) at any iterator i.

$X_{i,k,j}$ denotes the result of jth subject offered to kth learner at ith iterator. $X_{i,\text{teacher}}$ represents the teacher, that is, the best learner in a class at ith iterator.

The population X is randomly initialized by a search space bounded by $NP \times D$ matrix. The values of $X_{i,k,j}$ are assigned randomly using the equation

$$X_{0,k,j} = L_j + \text{rand} \times \left(U_j - L_j\right), \quad (1)$$

where $k = 1, 2, 3, \ldots, NP$ and $j = 1, 2, 3, \ldots, D$. The rand represents a uniformly distributed random variable within the range $[0, 1]$. $L_j \in (L_1, L_2, L_3, \ldots, L_D)$ represents the lower bound of design variable. $U_j \in (U_1, U_2, U_3, \ldots, U_D)$ represents the upper bound of design variable.

2.2. Teacher Phase. In this phase, the algorithm simulates the students learning from teachers. A good teacher can bring his or her learners up to his or her level in terms of knowledge. Hence, the mean result of a class may increase from a low level to the teacher's level. But, in fact, it is impossible that the mean result of a class reaches the teacher's level. Because of the individual differences and the forgetfulness of memory, the learners cannot gain all the knowledge of the teacher. A teacher can increase the mean result of a class to a certain value which depends on the capability of the whole class.

Let $M_{i,j} = (1/NP)(\sum_{k=1}^{k=NP} X_{i,k,j})$ be the mean result of the learners on a particular subject "j" ($j = 1, 2, \ldots, D$) and let $X_{i,\text{teacher}}$ be the teacher at any iteration i. $X_{i,\text{teacher}}$ will try to move mean $M_{i,j}$ towards its own level which is the new mean. Difference_Mean$_{i,k,j}$ is the difference between the existing mean result of each subject and the corresponding result of the teacher for each subject at the iteration i. The solution is updated according to the difference between the existing and the new means given by

$$\text{Difference\_Mean}_{i,k,j} = r_i \left(X_{i,\text{teacher},j} - T_F M_{i,j} \right), \quad (2)$$

$$T_F = \text{round}\left[1 + \text{rand}\left(0,1\right)\{1-2\}\right], \quad (3)$$

$$X_{i,k,j}^{\text{new}} = X_{i,k,j}^{\text{old}} + \text{Difference\_Mean}_{i,k,j}, \quad (4)$$

where $X_{i,\text{teacher},j}$ is the result of the teacher in subject j at the iteration i. r_i is a random number in the range $[0,1]$, and T_F is the teaching factor, which decides the value of mean to be changed. T_F can be either 1 or 2. The values of r_i and T_F are generated randomly in the algorithm and both of these parameters are not supplied as input to the algorithm.

In every iteration, $X_{i,k,j}^{\text{new}}$ is the updated value of $X_{i,k,j}^{\text{old}}$. Because the optimization problem is a minimization problem, our goal is to find the minimum of f. If the new value gives a better function value, then the old value is updated with the new value. The updated formula is given as

$$\begin{aligned} &\text{if} \quad f\left(X_{i,\text{total\_}k}^{\text{new}}\right) < f\left(X_{i,\text{total\_}k}^{\text{old}}\right) \\ &\qquad\qquad X_{i,k,j}^{\text{old}} = X_{i,k,j}^{\text{new}} \\ &\text{end if}, \end{aligned} \quad (5)$$

where $X_{i,\text{total\_}k}^{\text{new}}$ and $X_{i,\text{total\_}k}^{\text{old}}$ represent the new and old total result of kth student at the iteration i, respectively. All the accepted new values at the end of the teacher phase become the input to the learner phase.

2.3. Learner Phase. In learner phase, the algorithm simulates the learning of the learners through interaction among themselves. A learner interacts randomly with other learners to increase his or her knowledge. If a learner has more knowledge than others, the other learners can quickly achieve new knowledge by learning from him or her to increase their level. In this learning process, two learners are randomly selected. One is $X_{i,k}$ and another is $X_{i,q}$, $k \neq q$. The updated formula is given as

$$X_{i,k,j}^{\text{new}} = \begin{cases} X_{i,k,j}^{\text{old}} + r_i\left(X_{i,k,j} - X_{i,q,j}\right) & \text{if } f\left(X_{i,\text{total\_}k}\right) < f\left(X_{i,\text{total\_}q}\right), \\ X_{i,k,j}^{\text{old}} + r_i\left(X_{i,q,j} - X_{i,k,j}\right) & \text{otherwise}, \end{cases} \quad (6)$$

where r_i is a random number in the range $[0,1]$. $X_{i,\text{total\_}k}$ and $X_{i,\text{total\_}q}$ represent the total result of kth student and qth student at the iteration i, respectively. Accept the new value if it improves the value of the objective function. Similarly, use (5) to update the learner.

In each iteration of the TLBO, it is necessary to detect the repeated solution to the entire population. If there is a repeated solution, it needs to remove the repeated solution and generate a new individual randomly. Hence, it will expand the diversity of populations and avoid premature convergence of the algorithm. After a number of generations, the knowledge level of the entire class is smoothly approximated to a point that is considered the teacher, and the algorithm converges to a solution.

2.4. Algorithm Termination. The algorithm is terminated after MAXITER iterations. The details of TLBO algorithm can be referred to in literature [13, 14].

3. Nonlinear Inertia Weighted Teaching-Learning-Based Optimization

The basic TLBO algorithm is based on teaching-learning phenomenon of a classroom. In the teacher phase, the teacher tries to shift the mean of the learners towards himself or herself by teaching. In the learner phase, learners improve their knowledge by interaction among themselves. In the process of the teaching-learning, learners improve their level by accumulating knowledge. In other words, they learn new knowledge based on existing knowledge. In the real world, the teacher tends to wish that his or her students should achieve the knowledge equal to him in fast possible time. But it is impossible for a student because of his or her forgetting characteristics. In fact, a student usually forgets a part of existing knowledge due to the physiological phenomena of the brain. With increasing the iteration numbers of learning, more and more existing knowledge will be remembered. As the learning curve presented by Ebbinghaus, it describes how fast learning knowledge is in learning process. The sharpest increase occurs after the first try and then gradually evens out, meaning that less and less new knowledge is retained after each repetition. Like the forgetting curve, the learning curve is exponential. So it is necessary to add a memory weight to the existing knowledge of the student for simulating this learning scenario. According to this phenomenon, a nonlinear inertia weighted factor w is introduced into (4) and (6) in the basic TLBO, and this factor is considered as memory weighted factor which controls the memory rate of learners. This nonlinear inertia weighted factor will scale the existing knowledge of the learner for computing the new value. In contrast to the basic TLBO, in our algorithm the part of previous value of the learner is decided by a weighted factor w while computing the new learner value.

Accordingly, to meet the characteristic of memory to conform to the learning curve, the *nonlinear inertia weighted factor* w (i.e., memory rate) is nonlinearly increased from $w_{\min}$ to 1.0 over time, whose value is given as

$$w = 1 - \exp\left(\frac{-\text{iter}^2}{2 \times \left(\text{MAXITER}/8\right)^2}\right)\left(1 - w_{\min}\right), \quad (7)$$

where iter is the current iteration number, MAXITER is the maximum number of allowable iterations, and $w_{\min} \in [0.5, 1]$ is the minimum value of *nonlinear inertia weighted factor* w. The value $w_{\min}$ should be above 0.5 (here it is selected 0.6), or the individuals are worse due to remembering too little existing knowledge at first. Hence, if the value $w_{\min}$ is too small, the algorithm could not converge to the true global optimal solution. w curve (i.e., memory rate curve) is shown as Figure 1. The *nonlinear inertia weighted factor* w is applied to the new equations shown as (10) and (11). In this modified TLBO, the individuals try to sample diverse zones of the search space during the early stages of the search. During the later stages, the individuals adjust the movements

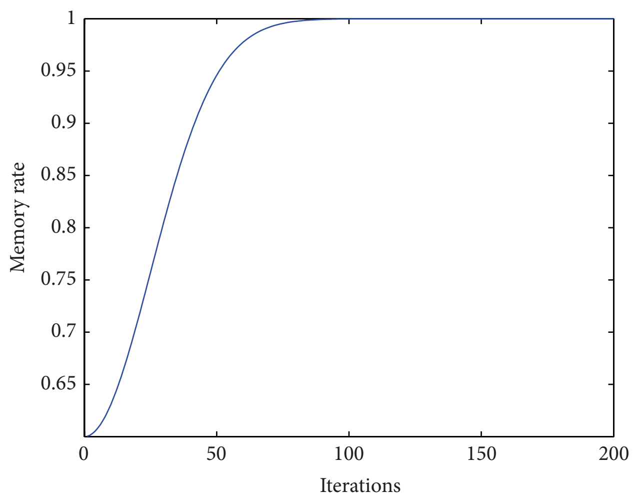


Figure 1: The memory rate curve.

of trial solutions finely so that they can explore the interior of a relative small space.

In the teacher phase, in order to obtain a new set of better learners, the difference between the existing mean result and the corresponding result of the teacher is added to the existing population of learners. Similarly, to obtain a new set of better learners in the learner phase, two learners are selected randomly, and the difference between their result of each corresponding subject is added to the existing learner. As (2) and (6) shown, the difference value added to the existing learner is formed from the difference of result and the random number r_i. Therefore, in the teacher and learner phases, the difference value is decided by the random number r_i to a large extent. In our proposed method, we modify the random number r_i as follows:

$$r_i' = \lambda + (1 - \lambda)\,\text{rand}\,(0, 1)\,, \tag{8}$$

where rand(0, 1) is a uniformly distributed random number within the range [0, 1]. The value $\lambda \in (0, 1)$ should be neither too big nor too small. Here, λ is selected to be 0.5, which conforms to the dynamic inertia weight proposed by Eberhart and Shi [28]. So (8) is modified as

$$r_i' = 0.5 + \frac{\text{rand}\,(0, 1)}{2}. \tag{9}$$

Equation (9) generates a random number in the range [0.5, 1] which is similar to the method proposed by Satapathy and Naik [23]. We call r_i' *dynamic inertia weighted factor*. Therefore, the mean value of the random number r_i is raised from 0.5 to 0.75. This increases the probability of stochastic variations and enlarges the difference value added to the existing learners, so as to improve population diversity, avoid prematurity in the search process, and increase the ability of the basic TLBO to escape from local optima. On the multimodal function surface, the original random weighed factor leads to most of the populations clustering near a local optimum point. However, the population with new dynamic inertia weight has more chances to jump out of the local optima and continuously move towards the global optimum point until a true global optimum is reached.

With the nonlinear inertia weighted factor and the dynamic inertia weighted factor, the new set of improved learners can be expressed by using equation in the teacher phase

$$X_{i,k,j}^{\text{new}} = wX_{i,k,j}^{\text{old}} + r_i' \left(X_{i,\text{teacher},j} - T_F M_{i,j}\right) \tag{10}$$

and the new set of improved learners can be expressed by using equation in the learner phase

$$X_{i,k,j}^{\text{new}} = \begin{cases} wX_{i,k,j}^{\text{old}} + r_i' \left(X_{i,k,j} - X_{i,q,j}\right) & \text{if } f\left(X_{i,\text{total\_}k}\right) < f\left(X_{i,\text{total\_}q}\right), \\ wX_{i,k,j}^{\text{old}} + r_i' \left(X_{i,q,j} - X_{i,k,j}\right) & \text{otherwise}, \end{cases} \tag{11}$$

where w is given by (7) and r_i' is given by (9).

4. Experiments on Benchmark Functions

In this section, NIWTLBO is applied on several benchmark functions to evaluate its performance with different dimensions and search space, comparing with the basic TLBO algorithm and with other optimization algorithms available in the literature. All tests are evaluated on a laptop having Intel core i5 2.67 GHz processor and 2 GB RAM. The algorithm is coded using the MATLAB programming language and run in MATLAB 2012a environment. This section provides the results obtained by the NIWTLBO algorithm compared to the basic TLBO and other intelligent optimization algorithms. The details of the 24 benchmark functions with different characteristics like unimodality/multimodality and separability/nonseparability are shown in Table 1. "C" denotes the characteristic of function; "D" is the dimensions of function; "range" of each function is the difference between the lower and upper bounds of the variables; "$f_{\min}$" is the theoretical global minimum solution.

4.1. Experiment 1: NIWTLBO versus PSO, ABC, DE, and TLBO. This experiment is aimed at identifying the performance of the NIWTLBO algorithm to achieve the global optimum value comparing with PSO, ABC, DE, and the basic TLBO. To be fair, each algorithm uses the same values of common control parameters such as population size and maximum evaluation number. Population size is 40 and the maximum fitness function evaluation number is 80,000 for all benchmark functions in Table 1. The other specific parameters of algorithms are given below.

PSO Setting. Cognitive attraction $C_1 = 2$, social attraction $C_2 = 2$, and inertia weight $w = 0.9$. As mentioned in [5], a recommended choice for constant C_1 and C_2 is integer 2, since it on average makes the weights for "social" and "cognition" parts be 1. When w is in the range of [0.8, 1.2], the PSO will have the best chance to find the global optimum and takes a moderate number of iterations [29].

Table 1: List of benchmark functions which have been used in experiments.

Number	Function	C	D	Range	Formulation	$f_{\min}$
f_1	Sphere	US	30	[−100, 100]	$f(x) = \sum_{i=1}^{D} x_i^2$	$f_{\min} = 0$
f_2	SumSquares	US	30	[−100, 100]	$f(x) = \sum_{i=1}^{D} ix_i^2$	$f_{\min} = 0$
f_3	Tablet	US	30	[−100, 100]	$f(x) = 10^6 x_1^2 + \sum_{i=1}^{D} x^2$	$f_{\min} = 0$
f_4	Quartic	US	30	[−1.28 1.28]	$f(x) = \sum_{i=1}^{D} ix_i^4 + \text{random}(0, 1)$	$f_{\min} = 0$
f_5	Schwefel 1.2	UN	30	[−100, 100]	$f(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	$f_{\min} = 0$
f_6	Schwefel 2.22	UN	30	[−10, 10]	$f(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$f_{\min} = 0$
f_7	Schwefel 2.21	UN	30	[−100, 100]	$f(x) = \max_i^D \{\lvert x_i \rvert\}, \quad 1 \le i \le D$	$f_{\min} = 0$
f_8	Zakharov	UN	30	[−5, 10]	$f(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5ix_i \right)^2 + \left(\sum_{i=1}^{D} 0.5ix_i \right)^4$	$f_{\min} = 0$
f_9	Rosenbrock	US	30	[−4, 4]	$f(x) = \sum_{i=1}^{D-1} \left[100 \left(x_{i+1} - x_i^2 \right)^2 + (1 - x_i)^2 \right]$	$f_{\min} = 0$
f_{10}	Schaffer	MN	2	[−10, 10]	$f(x) = \frac{\sin^2 \left(\sqrt{x_1^2 + x_2^2} \right) - 0.5}{\left(1 + 0.001 \left(x_1^2 + x_2^2\right)\right)^2} - 0.5$	$f_{\min} = -1$
f_{11}	Dropwave	MN	2	[−2, 2]	$f(x) = -\frac{1 + \cos \left(12 \sqrt{x_1^2 + x_2^2} \right)}{0.5 \left(x_1^2 + x_2^2\right) + 2}$	$f_{\min} = -1$
f_{12}	Bohachevsky1	MN	2	[−100, 100]	$f(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7$	$f_{\min} = 0$
f_{13}	Bohachevsky2	MN	2	[−100, 100]	$f(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) * \cos(4\pi x_2) + 0.3$	$f_{\min} = 0$
f_{14}	Bohachevsky3	MN	2	[−100, 100]	$f(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1 + 4\pi x_2) + 0.3$	$f_{\min} = 0$
f_{15}	Six-Hump Camel Back	MN	2	[−5, 5]	$f(x) = 4x_1^2 - 2.1x_1^4 - \frac{1}{3}x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	$f_{\min} = -1.03163$
f_{16}	Branin	MS	2	[−5, 15]	$f(x) = \left(x_2 - \frac{5.1}{4\pi^2} x_1^2 + \frac{5}{\pi} x_1 - 6 \right)^2 + 10 \left(1 - \frac{1}{8\pi} \right) \cos x_1 + 10$	$f_{\min} = 0.398$
f_{17}	Goldstein-Price	MN	2	[−2, 2]	$f(x) = \left[1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2\right) \right] \times \left[30 + (2x_1 - 3x_2)^2 \times \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2\right) \right]$	$f_{\min} = 3$
f_{18}	Ackley	MN	30	[−32, 32]	$f(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^{D} x_i^2} \right) - \exp \left(\frac{1}{D} \sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$	$f_{\min} = 0$

Table 1: Continued.

Number	Function	C	D	Range	Formulation	$f_{\min}$
f_{19}	Rastrigin	MN	30	[−5.12, 5.12]	$f(x) = \sum_{i=1}^{D}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right)$	$f_{\min} = 0$
f_{20}	Griewank	MN	30	[−600, 600]	$f(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$f_{\min} = 0$
f_{21}	Schwefel 2.26	MN	30	[−500, 500]	$f(x) = -\sum_{i=1}^{D}\left(x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	$f_{\min} = -837.9658$
f_{22}	Multimod	MN	30	[−10, 10]	$f(x) = \sum_{i=1}^{D}\lvert x_i\rvert\prod_{i=1}^{D}\lvert x_i\rvert$	$f_{\min} = 0$
f_{23}	Noncontinuous Rastrigin	MS	30	[−5.12, 5.12]	$f(x) = \sum_{i=1}^{D}\left(y_i^2 - 10\cos(2\pi y_i) + 10\right)$, where $y_i = \begin{cases} x_i & \lvert x_i\rvert < 0.5 \\ 0.5\text{round}(2x_i) & \lvert x_i\rvert \geq 0.5 \end{cases}$	$f_{\min} = 0$
f_{24}	Weierstrass	MS	30	[−0.5, 0.5]	$f(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{km}\left[a^k\cos\left(2\pi b^k(x_i + 0.5)\right)\right]\right) - D\sum_{k=0}^{km}\left[a^k\cos\left(2\pi b^k 0.5\right)\right]$ where $a = 0.5,\ b = 3,\ km = 20$	$f_{\min} = 0$

C: characteristic; D: dimension; U: unimodal; M: multimodal; S: separable; N: nonseparable.

ABC Setting. For ABC there are no other specific parameters to set.

DE Setting. In DE, F is a real constant which affects the differential variation between two solutions and R is crossover rate. Set $F = 0.5$ and $R = 0.4$. The configuration parameters for DE are decided on the results of experiments using different parameter values. We choose the parameter values which make the DE algorithms get the best result.

TLBO Settings. For TLBO there are no other specific parameters to set.

NIWTLBO Settings. In NIWTLBO, there are no other specific parameters too.

In this section, each benchmark function is independently experimented 30 times with PSO, ABC, DE, TLBO, and NIWTLBO. Each algorithm was terminated after running for 80,000FEs or when it reached the global minimum value before completely running for 80,000FEs. The mean and standard deviation of fitness value obtained through 30 experiments on each benchmark function are recorded in Table 2. Meanwhile, the mean value and standard deviations of the number of function fitness evaluations produced by the experiments are reported in Table 3. In order to analyze the performance whether there is significance between the results of the NIWTLBO and other algorithms, we carried out t-test on pairs of algorithms which is very popular in evolutionary computing [12]. The statistical significance levels of difference of the means of PSO and NIWTLBO algorithm, ABC and NIWTLBO algorithm, DE and NIWTLBO algorithm, and TLBO and NIWTLBO algorithm are reported in Table 4. Here, "+" symbol indicates that t value is significant at 0.05 level of significance by two tailed tests, "·" symbol marks t value being not statistically significant, and "NA" means not applicable due to the results of one pair of algorithms having the same accuracy.

The comparative results of each benchmark function for PSO, ABC, DE, and TLBO are presented in Table 2 in the form of average solution and standard deviation obtained in 30 independent runs on each benchmark function. The significance of NIWTLBO comparing with PSO, ABC, DE, and TLBO is shown in Table 4. It is observed from Tables 2 and 4 that the performance of NIWTLBO outperforms PSO, ABC, DE, and TLBO for functions f_1–f_8, f_{18}, f_{19}, f_{21}, and f_{23}. Furthermore, TLBO performs better than PSO, ABC, and DE for functions f_1–f_8 and f_{18}. For functions f_{10}–f_{17}, the performance of NIWTLBO, PSO, ABC, DE, and TLBO is alike that almost all the algorithms can obtain the global optimum value except for ABC on Bohachevsky3. For Rosenbrock, the performance of different algorithms is similar to each other. For Griewank and Multimod, the performance of NIWTLBO, DE, and TLBO is alike and better than PSO and ABC. For Weierstrass, the performance of NIWTLBO and TLBO is alike and outperforms PSO, ABC, and DE.

It is observed from the results in Table 3 that the smaller the number of fitness evaluations the more quickly the algorithm obtains the global optimum value; that is, the convergence rate of the algorithm is faster. Obviously, the NIWTLBO algorithm requires less numbers of function evaluations than the basic TLBO algorithm and other algorithms mentioned to achieve the global optimum value for most of the benchmark functions. Hence, the convergence rate of the NIWTLBO algorithm is faster than other algorithms mentioned for most of the benchmark functions except Six-Hump Camel Back, Branin, and Goldstein-Price.

4.2. Experiment 2: NIWTLBO versus PSO-w, PSO-cf, CPSO-H, and CLPSO. In this section, the experiment is aimed at analysing the ability of the NIWTLBO algorithm to obtain the global optimum value comparing with other variant PSO algorithms such as PSO-w [29], PSO-cf [30], CPSO-H [31], and CLPSO [32]. In this experiment, 8 different unimodal and multimodal benchmark functions are tested using the NIWTLBO algorithm. The details of benchmark functions are shown in Table 1. In order to maintain the consistency in the comparison, NIWTLBO algorithm is performed with the same maximum function evaluations and dimensions. Each benchmark function is independently experimented 30 times for NIWTLBO. The comparative results are reported in Table 5 in the form of the average solution and standard deviation obtained in 30 independent runs on each benchmark function. In Table 5, the results of algorithms except NIWTLBO are taken from literatures [24, 27], where the algorithms run 30,000FEs with 10 population sizes for 10 dimensional functions.

It is observed from the results in Table 5 that the performance of NIWTLBO and TLBO algorithms is better than PSO-w, PSO-cf, CPSO-H, and CLPSO algorithms for Sphere, Ackley, and Griewank. The performance of NIWTLBO and CLPSO is alike for Rastrigin, Noncontinuous Rastrigin, and Weierstrass. For Rosenbrock and Schwefel 2.26, the NIWTLBO algorithm does not perform well comparing with other algorithms.

4.3. Experiment 3: NIWTLBO versus CABC, GABC, RABC, and IABC. In this section, the experiment is conducted to identify the performance of the NIWTLBO algorithm to achieve the global optimum value versus CABC [33], GABC [34], RABC [8], and IABC [35] on 7 benchmark functions shown in Table 1. The comparative results are reported in Table 6. To maintain the consistency in the comparison, the parameters of the algorithms are similar to the literature [8], where the population size is set as 20 and dimension is set as 30. The results of CABC, GABC, RABC, and IABC are taken from the literature [23] directly. The results of NIWTLBO and TLBO, in the form of average solution and standard deviation, are obtained in 30 independent runs on each benchmark function. In this experiment, TLBO and NIWTLBO are tested with the same function evaluations listed in Table 6 to compare their performance with CABC, GABC, RABC, and IABC algorithms.

From Table 6, it is observed obviously that the performance of NIWTLBO and TLBO algorithms is better than CABC, GABC, and RABC for all benchmark functions. The performance of NIWTLBO algorithm is similar to IABC for

Table 2: Performance comparisons of PSO, ABC, DE, TLBO, and NIWTLBO in terms of fitness value. Population size: 40; D: 30 (except f_{10}~f_{17}: $2D$); max. eval.: 80,000FEs.

Number	Function	$f_{\min}$		PSO	ABC	DE	TLBO	NIWTLBO
f_1	Sphere	0	Mean	$8.99E-12$	$9.91E-16$	$7.15E-27$	$1.85E-286$	0
			Std.	$5.92E-12$	$5.36E-16$	$1.06E-26$	0	0
f_2	SumSquares	0	Mean	$1.11E-10$	$7.81E-16$	$9.06E-26$	$1.57E-286$	0
			Std.	$2.49E-10$	$1.32E-16$	$3.07E-26$	0	0
f_3	Tablet	0	Mean	$3.68E-08$	$9.54E-16$	$2.40E-26$	$7.66E-285$	0
			Std.	$1.64E-08$	$1.78E-16$	$1.87E-26$	0	0
f_4	Quartic	0	Mean	$5.84E-02$	$1.52E-01$	$4.03E-01$	$2.07E-02$	$2.03E-02$
			Std.	$3.83E-02$	$4.18E-02$	$1.29E-01$	$5.26E-02$	$3.52E-02$
f_5	Schwefel 1.2	0	Mean	$2.47E+05$	$8.82E+03$	$2.21E+04$	$1.52E-84$	0
			Std.	$1.48E+05$	$1.28E+03$	$5.21E+03$	$2.97E-84$	0
f_6	Schwefel 2.22	0	Mean	$5.16E-03$	$2.01E-14$	$4.31E-16$	$1.79E-143$	$4.45E-323$
			Std.	$6.94E-03$	$1.08E-14$	$1.04E-16$	$1.21E-143$	0
f_7	Schwefel 2.21	0	Mean	$1.21E+00$	$5.49E+01$	$1.21E-02$	$8.31E-120$	$2.40E-315$
			Std.	$6.02E-01$	$1.38E+01$	$2.81E-03$	$4.05E-120$	0
f_8	Zakharov	0	Mean	$1.62E+02$	$2.59E+02$	$5.84E+01$	$5.95E-51$	$1.06E-319$
			Std.	$6.33E+01$	$2.84E+01$	$7.01E+00$	$5.22E-51$	0
f_9	Rosenbrock	0	Mean	$3.01E+01$	$1.04E+01$	$2.43E+01$	$1.29E+01$	$1.83E+01$
			Std.	$2.57E+01$	$2.57E+00$	$4.61E+00$	$5.28E+00$	$6.91E+00$
f_{10}	Schaffer	−1	Mean	−1	−1	−1	−1	−1
			Std.	0	0	0	0	0
f_{11}	Dropwave	−1	Mean	−1	−1	−1	−1	−1
			Std.	0	0	0	0	0
f_{12}	Bohachevsky1	0	Mean	0	0	0	0	0
			Std.	0	0	0	0	0
f_{13}	Bohachevsky2	0	Mean	0	0	0	0	0
			Std.	0	0	0	0	0
f_{14}	Bohachevsky3	0	Mean	0	$8.46E-16$	0	0	0
			Std.	0	$2.95E-16$	0	0	0
f_{15}	Six-Hump Camel Back	−1.03163	Mean	−1.03163	−1.03163	−1.03163	−1.03163	−1.03163
			Std.	0	0	0	0	0
f_{16}	Branin	0.398	Mean	0.3979	0.3979	0.3979	0.3979	0.3979
			Std.	0	0	0	0	0
f_{17}	Goldstein-Price	3	Mean	3	3	3	3	3
			Std.	$8.11E-15$	$4.32E-15$	$1.36E-15$	$6.78E-16$	$6.56E-16$
f_{18}	Ackley	0	Mean	$1.18E+00$	$2.82E-13$	$2.49E-14$	$4.44E-15$	$8.66E-16$
			Std.	$3.85E-01$	$3.06E-14$	$6.07E-15$	0	0
f_{19}	Rastrigin	0	Mean	$1.08E+02$	$1.29E-13$	$9.33E+01$	$6.93E+00$	0
			Std.	$2.80E+01$	$2.57E-13$	$9.43E+00$	$5.92E+00$	0
f_{20}	Griewank	0	Mean	$6.77E-03$	$7.10E-03$	0	0	0
			Std.	$9.29E-03$	$9.56E-03$	0	0	0
f_{21}	Schwefel 2.26	−837.9658	Mean	−8789.43	−12561.79	−11312.51	−9178.59	−8324.302
			Std.	$4.63E+02$	$1.96E+02$	$1.58E+03$	$7.97E+02$	$1.71E+02$
f_{22}	Multimod	0	Mean	$8.69E-67$	$8.52E-19$	$4.66E-311$	0	0
			Std.	$1.74E-66$	$8.34E-19$	0	0	0
f_{23}	Noncontinuous Rastrigin	0	Mean	$1.83E+02$	$1.99E-14$	$6.94E+01$	$1.55E+01$	0
			Std.	$3.15E+01$	$1.83E-14$	$9.13E+00$	$2.65E+00$	0
f_{24}	Weierstrass	0	Mean	$6.27E+01$	$1.12E-02$	$1.38E+01$	0	0
			Std.	$2.03E+01$	$7.73E-03$	$6.07E-01$	0	0

Table 3: Convergence comparisons in terms of number of fitness evaluations. Population size: 40; D: 30 (except f_{10}~f_{17}: $2D$); max. eval.: 80,000FEs.

Number	Function		PSO	ABC	DE	TLBO	NIWTLBO
f_1	Sphere	Mean	80,000	80,000	80,000	80,000	29,514
		Std.	0	0	0	0	$1.02E + 02$
f_2	SumSquares	Mean	80,000	80,000	80,000	80,000	29,628
		Std.	0	0	0	0	$1.23E + 02$
f_3	Tablet	Mean	80,000	80,000	80,000	80,000	29,562
		Std.	0	0	0	0	$1.52E + 02$
f_4	Quartic	Mean	80,000	80,000	80,000	80,000	80,000
		Std.	0	0	0	0	0
f_5	Schwefel 1.2	Mean	80,000	80,000	80,000	80,000	39,416
		Std.	0	0	0	0	$1.09E + 02$
f_6	Schwefel 2.22	Mean	80,000	80,000	80,000	80,000	80,000
		Std.	0	0	0	0	0
f_7	Schwefel 2.21	Mean	80,000	80,000	80,000	80,000	80,000
		Std.	0	0	0	0	0
f_8	Zakharov	Mean	80,000	80,000	80,000	80,000	80,000
		Std.	0	0	0	0	0
f_9	Rosenbrock	Mean	80,000	80,000	80,000	80,000	80,000
		Std.	0	0	0	0	0
f_{10}	Schaffer	Mean	12,432	43,636	8,686	9,688	3,029
		Std.	$3.38E + 02$	$3.03E + 02$	$2.06E + 02$	$2.29E + 02$	$3.03E + 02$
f_{11}	Dropwave	Mean	11,394	13,824	5,490	3,021	812
		Std.	$3.26E + 01$	$1.09E + 02$	$1.53E + 02$	$1.22E + 02$	$3.32E + 01$
f_{12}	Bohachevsky1	Mean	9,532	3,263	3,992	2,266	842
		Std.	$2.21E + 02$	$7.52E + 01$	$8.74E + 01$	$3.23E + 01$	$2.01E + 01$
f_{13}	Bohachevsky2	Mean	9,578	4,717	4,245	2,568	952
		Std.	$1.33E + 02$	$9.27E + 01$	$1.17E + 02$	$2.05E + 01$	$2.56E + 01$
f_{14}	Bohachevsky3	Mean	9,792	80,000	5,376	2,875	965
		Std.	$2.52E + 02$	0	$1.26E + 02$	$1.03E + 02$	$3.12E + 01$
f_{15}	Six-Hump Camel Back	Mean	1,997	1,372	1,781	712	2,560
		Std.	$1.38E + 02$	$1.17E + 02$	$1.36E + 02$	$5.93E + 01$	$9.07E + 01$
f_{16}	Branin	Mean	1,851	1,813	1,891	1,086	2,172
		Std.	$1.17E + 02$	$1.23E + 02$	$1.04E + 02$	$1.06E + 02$	$1.23E + 02$
f_{17}	Goldstein-Price	Mean	2,018	1,857	1,765	1,228	2,865
		Std.	$1.25E + 02$	$1.48E + 02$	$2.08E + 02$	$6.85E + 01$	$1.42E + 02$
f_{18}	Ackley	Mean	80,000	80,000	80,000	80,000	80,000
		Std.	0	0	0	0	0
f_{19}	Rastrigin	Mean	80,000	80,000	80,000	80,000	1,436
		Std.	0	0	0	0	$3.02E + 01$
f_{20}	Griewank	Mean	80,000	80,000	53,032	12,064	1,284
		Std.	0	0	$6.16E + 02$	$9.37E + 01$	$2.54E + 01$
f_{21}	Schwefel 2.26	Mean	80,000	80,000	80,000	80,000	80,000
		Std.	0	0	0	0	0
f_{22}	Multimod	Mean	80,000	80,000	80,000	28,304	1,427
		Std.	0	0	0	$1.05E + 02$	$5.16E + 01$
f_{23}	Noncontinuous Rastrigin	Mean	80,000	80,000	80,000	80,000	1,324
		Std.	0	0	0	0	$1.22E + 02$
f_{24}	Weierstrass	Mean	80,000	80,000	80,000	12,712	2,044
		Std.	0	0	0	$1.19E + 02$	$1.21E + 02$

Table 4: t value, significant at 0.05 level of significance by two tailed tests using Table 2. The significance of NIWTLBO compares with PSO, ABC, DE, and TLBO.

Number	Function	PSO	ABC	DE	TLBO	Number	Function	PSO	ABC	DE	TLBO
f_1	Sphere	+	+	+	+	f_{13}	Bohachevsky2	NA	NA	NA	NA
f_2	SumSquares	+	+	+	+	f_{14}	Bohachevsky3	NA	+	NA	NA
f_3	Tablet	+	+	+	+	f_{15}	Six-Hump Camel Back	NA	NA	NA	NA
f_4	Quartic	+	+	+	·	f_{16}	Branin	NA	NA	NA	NA
f_5	Schwefel 1.2	+	+	+	+	f_{17}	Goldstein-Price	NA	NA	NA	NA
f_6	Schwefel 2.22	+	+	+	+	f_{18}	Ackley	+	+	+	+
f_7	Schwefel 2.21	+	+	+	+	f_{19}	Rastrigin	+	+	+	+
f_8	Zakharov	+	+	+	+	f_{20}	Griewank	+	+	NA	NA
f_9	Rosenbrock	·	·	·	·	f_{21}	Schwefel 2.26	+	+	+	+
f_{10}	Schaffer	NA	NA	NA	NA	f_{22}	Multimod	+	+	NA	NA
f_{11}	Dropwave	NA	NA	NA	NA	f_{23}	Noncontinuous Rastrigin	+	+	+	+
f_{12}	Bohachevsky1	NA	NA	NA	NA	f_{24}	Weierstrass	+	+	+	NA

"+" indicates that t value is significant, "·" indicates that t value is not statistically significant, and "NA" stands for not applicable.

Table 5: Comparative results of TLBO and NIWTLBO with other PSO algorithms. Population size: 10; D: 10; max. eval.: 30,000FEs; source: results of algorithms except NIWTLBO are taken from [24, 27].

Number	Function		PSO-w	PSO-cf	CPSO-H	CLPSO	TLBO	NIWTLBO
f_1	Sphere	Mean	$7.96E-51^{\dagger}$	$9.84E-105^{\dagger}$	$4.98E-45^{\dagger}$	$5.15E-29^{\dagger}$	0	0
		Std.	$3.56E-50$	$4.21E-104$	$1.00E-44$	$2.16E-28$	0	0
f_9	Rosenbrock	Mean	$3.08E+00^{\dagger}$	$6.98E-01^{\ddagger}$	$1.53E+00^{\ddagger}$	$2.46E+00^{\dagger}$	$1.72E+00^{\ddagger}$	$1.69E+00$
		Std.	$7.69E-01$	$1.46E+00$	$1.70E+00$	$1.70E+00$	$6.62E-01$	$7.18E-01$
f_{18}	Ackley	Mean	$1.58E-14^{\dagger}$	$9.18E-01^{\dagger}$	$1.49E-14^{\dagger}$	$4.32E-10^{\dagger}$	$3.55E-15^{\dagger}$	$8.58E-16$
		Std.	$1.60E-14$	$1.01E+00$	$6.97E-15$	$2.55E-14$	$8.32E-31$	$6.37E-32$
f_{19}	Rastrigin	Mean	$5.82E+00^{\dagger}$	$1.25E+01^{\dagger}$	$2.12E+00^{\dagger}$	0	$6.77E-08^{\dagger}$	0
		Std.	$2.96E+00$	$5.17E+00$	$1.33E+00$	0	$3.68E-07$	0
f_{20}	Griewank	Mean	$9.69E-02^{\dagger}$	$1.19E-01^{\dagger}$	$4.07E-02^{\dagger}$	$4.56E-03^{\dagger}$	0	0
		Std.	$5.01E-02$	$7.11E-02$	$2.80E-02$	$4.81E-03^{\dagger}$	0	0
f_{21}	Schwefel 2.26	Mean	$3.20E+02^{\dagger}$	$9.87E+02^{\dagger}$	$2.13E+02^{\ddagger}$	$0^{\ddagger}$	$2.94E+02^{\dagger}$	$2.67E+02$
		Std.	$1.85E+02$	$2.76E+02$	$1.41E+02$	0	$2.68E+02$	$1.92E+02$
f_{23}	Noncontinuous Rastrigin	Mean	$4.05E+00^{\dagger}$	$1.20E+01^{\dagger}$	$2.00E-01^{\dagger}$	0	$2.65E-08^{\dagger}$	0
		Std.	$2.58E+00$	$4.99E+00$	$4.10E-01$	0	$1.23E-07$	0
f_{24}	Weierstrass	Mean	$2.28E-03^{\dagger}$	$6.69E-01^{\dagger}$	$1.07E-15^{\dagger}$	0	$2.42E-05^{\dagger}$	0
		Std.	$7.04E-03$	$7.17E-01$	$1.67E-15$	0	$1.38E-20$	0

"†" mark indicates that NIWTLBO is statistically better than the corresponding algorithm.
"‡" mark indicates that NIWTLBO is statistically worse than the corresponding algorithm.

Rastrigin and Griewank and outperforms the IABC for the rest of benchmark functions in Table 6.

4.4. Experiment 4: NIWTLBO versus SaDE, jDE, and JADE. In this section, the experiment is carried out for comparing the performance of the NIWTLBO algorithm with SaDE, jDE, and JADE algorithms on 7 benchmark functions which are described in Table 1. The results of SaDE, jDE, and JADE are taken from the literature [36] directly. The results of NIWTLBO and TLBO, in the form of average solution and standard deviation, are obtained in 30 independent runs on each benchmark function. To be fair, the parameters of the algorithms are the same to the literature [36], where the population size is 20 and the dimension is 30. The comparative results are recorded in Table 7. In this experiment, TLBO and NIWTLBO are implemented with the same function evaluations listed in Table 7 to compare their performance with SaDE, jDE, and JADE algorithms.

It can be seen that NIWTLBO performs much better than these variants of DE on all the benchmark functions in Table 7. Therefore, it is shown that the NIWTLBO algorithm has a good performance.

4.5. Experiment 5: NIWTLBO versus TLBO with Different Dimensions. In this section, we analyse the convergence of NIWTLBO and TLBO algorithms with different dimensions. Two unimodal functions and two multimodal functions have been tested with dimensions 2, 10, 50, and 100. In this work,

Table 6: Comparative results of TLBO and NIWTLBO with other variants of ABC algorithms. Population size: 20; D: 30; source: results of algorithms except TLBO and NIWTLBO are taken from [23].

Number	Function		CABC	GABC	RABC	IABC	TLBO	NIWTLBO
f_1	Sphere	Mean	$2.3E-40^{\dagger}$	$3.6E-63^{\dagger}$	$9.1E-61^{\dagger}$	$5.34E-178^{\dagger}$	0	0
	FEs: 1.5×10^5	Std.	$1.7E-40$	$5.7E-63$	$2.1E-60$	0	0	0
f_5	Schwefel 1.2	Mean	$8.4E+02^{\dagger}$	$4.3E+02^{\dagger}$	$2.9E-24^{\dagger}$	$1.78E-65^{\dagger}$	0	0
	FEs: 5.0×10^5	Std.	$9.1E+02$	$8.0E+02$	$1.5E-23$	$2.21E-65$	0	0
f_6	Schwefel 2.22	Mean	$3.5E-30^{\dagger}$	$4.8E-45^{\dagger}$	$3.2E-74^{\dagger}$	$8.82E-127^{\dagger}$	0	0
	FEs: 2.0×10^5	Std.	$4.8E-30$	$1.4E-45$	$2.0E-73$	$3.49E-126$	0	0
f_7	Schwefel 2.21	Mean	$6.1E-03^{\dagger}$	$3.6E-06^{\dagger}$	$2.8E-02^{\dagger}$	$4.98E-38^{\dagger}$	0	0
	FEs: 5.0×10^5	Std.	$5.7E-03$	$7.6E-07$	$1.7E-02$	$8.59E-38$	0	0
f_{18}	Ackley	Mean	$1.0E-05^{\dagger}$	$1.8E-09^{\dagger}$	$9.6E-07^{\dagger}$	$3.87E-14^{\dagger}$	$4.48E-15^{\dagger}$	$8.65E-16$
	FEs: 5.0×10^4	Std.	$2.4E-06$	$7.7E-10$	$8.3E-07$	$8.52E-15$	$2.16E-30$	$2.38E-31$
f_{19}	Rastrigin	Mean	$1.3E-00^{\dagger}$	$1.5E-10^{\dagger}$	$2.3E-02^{\dagger}$	0	$6.36E+00^{\dagger}$	0
	FEs: 1.0×10^5	Std.	$2.7E-00$	$2.7E-10$	$5.1E-01$	0	$4.78E+00$	0
f_{20}	Griewank	Mean	$1.2E-04^{\dagger}$	$6.0E-13^{\dagger}$	$8.7E-08^{\dagger}$	0	0	0
	FEs: 5.0×10^5	Std.	$4.6E-04$	$7.7E-13$	$2.1E-08$	0	0	0

"†" mark indicates that NIWTLBO is statistically better than the corresponding algorithm.

Table 7: Comparative results of TLBO and NIWTLBO with other variants of DE algorithms. Population size: 20; D: 30; source: results of algorithms except TLBO and NIWTLBO are taken from [23].

Number	Function		SaDE	jDE	JADE	TLBO	NIWTLBO
f_1	Sphere	Mean	$4.5E-20^{\dagger}$	$2.5E-28^{\dagger}$	$1.8E-60^{\dagger}$	0	0
	FEs: 1.5×10^5	Std.	$1.9E-14$	$3.5E-28$	$8.4E-60$	0	0
f_5	Schwefel 1.2	Mean	$9.0E-37^{\dagger}$	$5.2E-14^{\dagger}$	$5.7E-61^{\dagger}$	0	0
	FEs: 5.0×10^5	Std.	$5.4E-36$	$1.1E-13$	$2.7E-60$	0	0
f_6	Schwefel 2.22	Mean	$1.9E-14^{\dagger}$	$1.5E-23^{\dagger}$	$1.8E-25^{\dagger}$	0	0
	FEs: 2.0×10^5	Std.	$1.1E-14$	$1.0E-23$	$8.8E-25$	0	0
f_7	Schwefel 2.21	Mean	$7.4E-11^{\dagger}$	$1.4E-15^{\dagger}$	$8.2E-24^{\dagger}$	0	0
	FEs: 5.0×10^5	Std.	$1.82E-10$	$1.0E-15$	$4.0E-23$	0	0
f_{18}	Ackley	Mean	$2.7E-03^{\dagger}$	$3.5E-04^{\dagger}$	$8.2E-10^{\dagger}$	$4.48E-15^{\dagger}$	$8.65E-16$
	FEs: 5.0×10^4	Std.	$5.1E-04$	$1.0E-04$	$6.9E-10$	$2.16E-30$	$2.38E-31$
f_{19}	Rastrigin	Mean	$1.2E-03^{\dagger}$	$1.5E-04^{\dagger}$	$1.0E-04^{\dagger}$	$6.36E+00^{\dagger}$	0
	FEs: 1.0×10^5	Std.	$6.5E-04$	$2.0E-04$	$6.0E-05$	$4.78E+00$	0
f_{20}	Griewank	Mean	$7.8E-04^{\dagger}$	$1.9E-05^{\dagger}$	$9.9E-08^{\dagger}$	0	0
	FEs: 5.0×10^5	Std.	$1.2E-03$	$5.8E-05$	$6.0E-07$	0	0

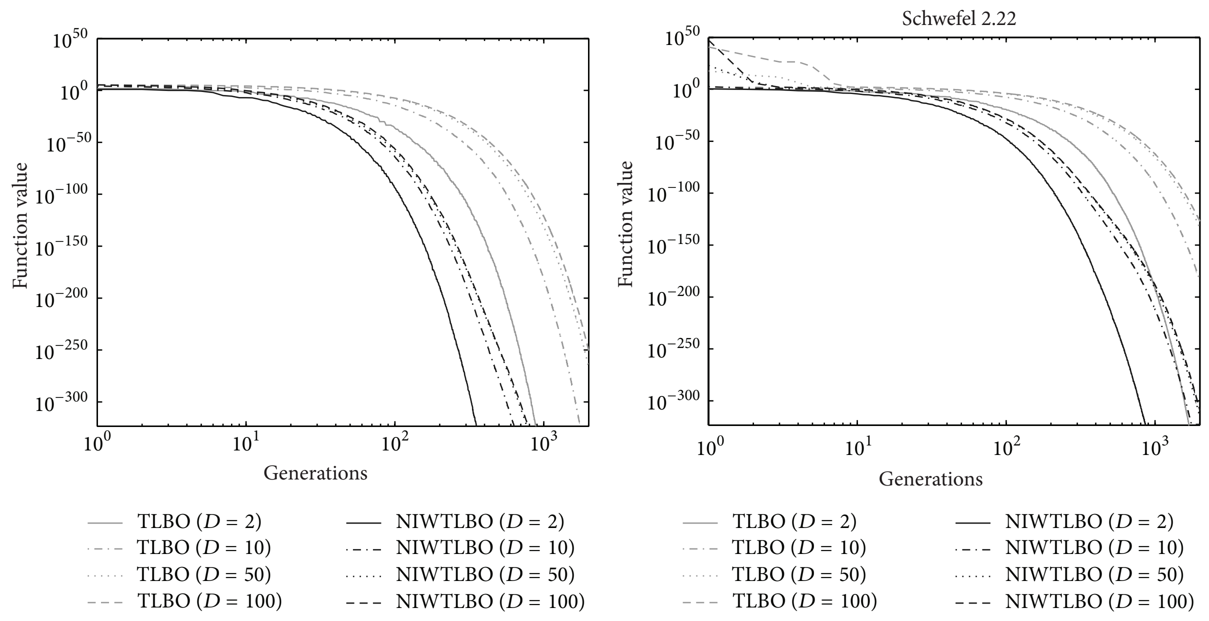
"†" mark indicates that NIWTLBO is statistically better than the corresponding algorithm.

evolutionary generation is employed to evaluate the performance of NIWTLBO and TLBO algorithms. The population size is set as 40 and the number of evolutionary generations is set as 2000. The experiment results of NIWTLBO and TLBO algorithms for 2, 10, 50, and 100 dimensional functions over 30 independent runs are listed in Table 8, which is in form of the mean solution. The graphs are plotted between the function value and evolutionary generations on logarithmic scale.

Figures 2 and 3 show the convergence graphs of the unimodal and multimodal functions for different dimensions, respectively. It is observed from the graphs that the convergence rate of the NIWTLBO algorithm is faster than the basic TLBO algorithm for both these unimodal and multimodal functions for all dimensions. Furthermore, it is

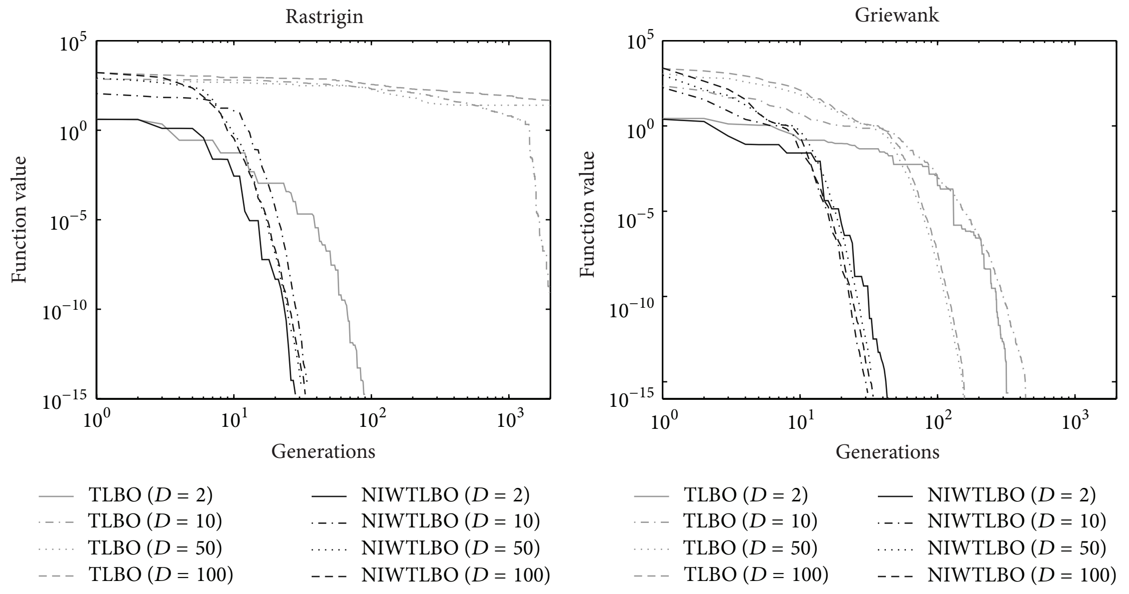
Table 8: Comparative results of TLBO and NIWTLBO with different dimensions. Population size: 40; generations: 2000.

Function	D	Unimodal		Multimodal	
		Sphere	Schwefel 2.22	Rastrigin	Griewank
TLBO	2	0	0	0	0
	10	0	$1.05E-184$	$5.78E-08$	0
	50	$2.09E-267$	$4.64E-134$	$2.48E+01$	0
	100	$4.13E-251$	$8.91E-128$	$4.71E+01$	0
NIWTLBO	2	0	0	0	0
	10	0	$2.50E-323$	0	0
	50	0	$4.43E-317$	0	0
	100	0	$4.09E-310$	0	0



(a) Sphere (b) Schwefel 2.22

Figure 2: Convergence of TLBO and NIWTLBO algorithms for unimodal function.



(a) Rastrigin (b) Griewank

Figure 3: Convergence of TLBO and NIWTLBO algorithms for multimodal function.

observed from Table 8 and the figures that the performance of NIWTLBO algorithm is almost not affected by the dimension. But the performance of TLBO algorithm will be reduced slightly with the dimension increasing.

4.6. Experiment 6: NIWTLBO versus Other Variants of TLBO. In order to show the advantages and disadvantages of the NIWTLBO, we make experiments to compare the performance of the NIWTLBO algorithm with some other variants of TLBO in this section. The variants of TLBO include WTLBO [21], ITLBO22 [22], ITLBO23 [23], and ITLBO [24]. Some benchmark functions described in Table 1 are tested for experiments. In the experiments, the population size is 20 and dimension is 2. The number of teachers is 4 in ITLBO. To maintain the consistency, the execution of the NIWTLBO and other variants of TLBO algorithms is stopped after running for 80,000FEs or when the difference between the fitness obtained by the algorithm and the global optimum value is less than 0.1% (e.g., if the optimum

Table 9: Comparative results of NIWTLBO and different variants of TLBO algorithms. Population size: 20; D: 2; max. eval.: 80,000FEs.

Number	Function		WTLBO	ITLBO22	ITLBO23	I-TLBO (NT = 4)	NIWTLBO
f_1	Sphere	MNFE	365	386	482	372	281
		Succ%	100	100	100	100	100
f_6	Schwefel 2.22	MNFE	442	428	563	416	324
		Succ%	100	100	100	100	100
f_9	Rosenbrock	MNFE	1643	704	726	684	1606
		Succ%	65	100	100	100	100
f_{14}	Bohachevsky3	MNFE	468	432	516	398	364
		Succ%	100	100	100	100	100
f_{16}	Branin	MNFE	41010	649	763	367	1922
		Succ%	28	100	100	100	100
f_{18}	Ackley	MNFE	564	508	682	491	443
		Succ%	100	100	100	100	100
f_{19}	Rastrigin	MNFE	4608	651	1406	632	481
		Succ%	100	100	100	100	100
f_{20}	Griewank	MNFE	18246	1208	2248	1024	965
		Succ%	85	100	81	100	100
f_{24}	Weierstrass	MNFE	19642	1243	2325	1186	1042
		Succ%	78	100	93	100	100

value is 0, the solution is accepted if it differs from the optimum value by less than 0.001). If the solution to the algorithm is not accepted after running for 80,000FEs, it is unsuccessful. Each benchmark function is tested 100 times with the NIWTLBO and other variants of TLBO algorithms and the comparative results in the form of mean function evaluations and success percentage are shown in Table 9. "MNFE" denotes the number of function evaluations when the solution is accepted. The number of function evaluations in the variants of TLBO is = (2 × population size × number of generations).

It is observed from Table 9 that, except for Rosenbrock and Branin, the NIWTLBO algorithm requires fewer number of function evaluations than other algorithms to reach the global optimum value, with a very high success rate of 100%. For Rosenbrock, Branin, Griewank, and Weierstrass, the WTLBO algorithm performs worse than other algorithms with low success rate, which is easily trapped in local optima. From this, it is shown that the NIWTLBO algorithm has a better performance than some other variants of TLBO.

5. Conclusion

In this paper, we propose the NIWTLBO algorithm which introduced a nonlinear inertia weighted factor into the basic TLBO to control the memory rate of learners and used a dynamic inertia weighted factor to replace the original random number in teacher phase and learner phase. The proposed algorithm is implemented on 24 benchmark functions having different characteristics to evaluate its performance which is compared with the basic TLBO and some other state-of-the-art optimization algorithms available in the literature. Furthermore, the comparisons between the NIWTLBO and other algorithms mentioned are also reported.

The experiment results have shown the satisfactory performance of the NIWTLBO algorithm for solving global optimization problems. The NIWTLBO algorithm not only enhances the local searching ability of TLBO but also improves the global performance. Moreover, the NIWTLBO algorithm can increase the convergence speed and enhance the ability of the TLBO to escape from local optima.

In future work, the NIWTLBO algorithm will be extended to handle more complex functions and solve constrained/multiobjective optimization problems. Furthermore, we will also open up a new way to improve the diversity of TLBO using a hybrid method, so as to utilize the advantages of other intelligent algorithms to further improve the global performance of TLBO.

Conflict of Interests

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

Acknowledgments

The present study was partially supported by the National Natural Science Foundation of China (10872160). The authors thank Rao R. V. for providing the source code of the basic TLBO algorithm and Dervis Karaboga for providing the source code of the ABC algorithm.

References

[1] C. Blum, "Ant colony optimization: introduction and recent trends," *Physics of Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[2] B. C. Mohan and R. Baskaran, "A survey: ant colony optimization based recent research and implementation on several

engineering domain," *Expert Systems with Applications*, vol. 39, no. 4, pp. 4618–4627, 2012.

[3] B. Goswami and D. Mandal, "A genetic algorithm for the level control of nulls and side lobes in linear antenna arrays," *Journal of King Saud University—Computer and Information Sciences*, vol. 25, no. 2, pp. 117–126, 2013.

[4] M. Thakur, "A new genetic algorithm for global optimization of multimodal continuous functions," *Journal of Computational Science*, vol. 5, no. 2, pp. 298–311, 2014.

[5] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[6] Z. Beheshti and S. M. Shamsuddin, "Non-parametric particle swarm optimization for global optimization," *Applied Soft Computing Journal*, vol. 28, pp. 345–359, 2015.

[7] M. Tanweer, S. Suresh, and N. Sundararajan, "Self regulating particle swarm optimization algorithm," *Information Sciences*, vol. 294, pp. 182–202, 2015.

[8] F. Kang, J. Li, and Z. Ma, "Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions," *Information Sciences*, vol. 181, no. 16, pp. 3508–3531, 2011.

[9] B. Akay and D. Karaboga, "Artificial bee colony algorithm for large-scale problems and engineering design optimization," *Journal of Intelligent Manufacturing*, vol. 23, no. 4, pp. 1001–1014, 2012.

[10] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artificial Intelligence Review*, vol. 42, no. 1, pp. 21–57, 2014.

[11] 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.

[12] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[13] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems," *Computer-Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.

[14] R. V. Rao, V. J. Savsani, and D. P. Vakharia, "Teaching-learning-based optimization: an optimization method for continuous non-linear large scale problems," *Information Sciences*, vol. 183, no. 1, pp. 1–15, 2012.

[15] R. V. Rao and V. Patel, "An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 3, no. 4, pp. 535–560, 2012.

[16] R. Venkata Rao and V. D. Kalyankar, "Parameter optimization of modern machining processes using teaching—learning-based optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 1, pp. 524–531, 2013.

[17] R. V. Rao and V. Patel, "Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm," *Applied Mathematical Modelling*, vol. 37, no. 3, pp. 1147–1162, 2013.

[18] A. Shabanpour-Haghighi, A. R. Seifi, and T. Niknam, "A modified teaching-learning based optimization for multi-objective optimal power flow problem," *Energy Conversion and Management*, vol. 77, pp. 597–607, 2014.

[19] S. Sultana and P. K. Roy, "Multi-objective quasi-oppositional teaching learning based optimization for optimal location of distributed generator in radial distribution systems," *International Journal of Electrical Power & Energy Systems*, vol. 63, pp. 534–545, 2014.

[20] M. Ghasemi, S. Ghavidel, M. Gitizadeh, and E. Akbari, "An improved teaching-learning-based optimization algorithm using Lévy mutation strategy for non-smooth optimal power flow," *International Journal of Electrical Power & Energy Systems*, vol. 65, pp. 375–384, 2015.

[21] S. C. Satapathy, A. Naik, and K. Parvathi, "Weighted teaching-learning-based optimization for global function optimization," *Applied Mathematics*, vol. 4, no. 3, pp. 429–439, 2013.

[22] D. Chen, F. Zou, Z. Li, J. Wang, and S. Li, "An improved teaching–learning-based optimization algorithm for solving global optimization problem," *Information Sciences*, vol. 297, pp. 171–190, 2015.

[23] S. C. Satapathy and A. Naik, "Improved teaching learning based optimization for global function optimization," *Decision Science Letters*, vol. 2, no. 1, pp. 23–34, 2013.

[24] R. V. Rao and V. Patel, "An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems," *Scientia Iranica*, vol. 20, no. 3, pp. 710–720, 2013.

[25] F. Zou, L. Wang, X. Hei, D. Chen, and B. Wang, "Multi-objective optimization using teaching-learning-based optimization algorithm," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 4, pp. 1291–1300, 2013.

[26] R. V. Rao and V. Patel, "A multi-objective improved teaching-learning based optimization algorithm for unconstrained and constrained optimization problems," *International Journal of Industrial Engineering Computations*, vol. 5, no. 1, pp. 1–22, 2014.

[27] B. Akay and D. Karaboga, "A modified Artificial Bee Colony algorithm for real-parameter optimization," *Information Sciences*, vol. 192, pp. 120–142, 2012.

[28] R. C. Eberhart and Y. Shi, "Tracking and optimizing dynamic systems with particle swarms," in *Proceedings of the Congress on Evolutionary Computation*, vol. 1, pp. 94–100, May 2001.

[29] Y. Shi and R. C. Eberhart, "A modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence*, pp. 69–73, IEEE, Anchorage, Alaska, USA, May 1998.

[30] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.

[31] V. D. B. Frans and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, pp. 225–239, 2004.

[32] J. J. Liang, P. N. Suganthan, A. K. Qin, 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.

[33] B. Alatas, "Chaotic bee colony algorithms for global numerical optimization," *Expert Systems with Applications*, vol. 37, no. 8, pp. 5682–5687, 2010.

[34] G. P. Zhu and S. Kwong, "Gbest-guided artificial bee colony algorithm for numerical function," *Applied Soft Computing*, vol. 10, pp. 445–456, 2010.

[35] W. Gao and S. Liu, "Improved artificial bee colony algorithm for global optimization," *Information Processing Letters*, vol. 111, no. 17, pp. 871–882, 2011.

[36] 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.