Article

A Cyclical Non-Linear Inertia-Weighted Teaching–Learning-Based Optimization Algorithm

Zongsheng Wu ¹ and Ru Xue ^{2,*}¹ School of Computer Science, Xianyang Normal University, Xianyang 712000, China; wuzs2005@163.com² School of Information Engineering, Xizang Minzu University, Xianyang 712000, China

* Correspondence: rxue@xzmu.edu.cn; Tel.: +86-139-9105-5209

Received: 6 March 2019; Accepted: 23 April 2019; Published: 3 May 2019

Abstract: After the teaching–learning-based optimization (TLBO) algorithm was proposed, many improved algorithms have been presented in recent years, which simulate the teaching–learning phenomenon of a classroom to effectively solve global optimization problems. In this paper, a cyclical non-linear inertia-weighted teaching–learning-based optimization (CNIWTLBO) algorithm is presented. This algorithm introduces a cyclical non-linear inertia weighted factor into the basic TLBO to control the memory rate of learners, and uses a non-linear mutation factor to control the learner’s mutation randomly during the learning process. In order to prove the significant performance of the proposed algorithm, it is tested on some classical benchmark functions and the comparison results are provided against the basic TLBO, some variants of TLBO and some other well-known optimization algorithms. The experimental results show that the proposed algorithm has better global search ability and higher search accuracy than the basic TLBO, some variants of TLBO and some other algorithms as well, and can escape from the local minimum easily, while keeping a faster convergence rate.

Keywords: optimization; teaching–learning-based optimization (TLBO); global optimization; swarm intelligent optimization

1. Introduction

As is well known, the research and application of swarm intelligence optimization mostly focus on nature-inspired algorithms. In the past decades, many classical nature-inspired optimization algorithms based on population have been proposed. These algorithms have been proven effective in solving global optimization problems and specific types of engineering optimization problems, such as GA (genetic algorithm) [1,2], ACO (ant colony optimization) [3,4], PSO (particle swarm optimization) [5,6], ABC (artificial bee colony) [7–9] and DE (differential evolution) [10–12], etc. However, any algorithm has its own merits and demerits in solving diverse problems. In order to solve the demerits, such as easily being trapped into a local optimum and slow convergence, numerous improved algorithms for various swarm intelligence algorithms are presented, including the variants of various algorithms or their hybrid algorithms. In general, the quality of an optimization algorithm depends on three basic factors, i.e., the ability obtaining the true global optima value, the fast convergence speed and a minimum of control parameters. Therefore, our ultimate aim in the practical optimization applications is that the optimization algorithms should have higher calculation accuracy, faster convergence speed and a minimum of control parameters for ease of use in program.

The teaching–learning-based optimization (TLBO) algorithm is a swarm intelligence algorithm, which simulates the phenomenon of teaching and learning in a class. A few years ago, the TLBO algorithm was first proposed by Rao et al. [13,14]. The TLBO is a parameter-less algorithm [15] requiring only the common control parameters, such as population size and numbers of generation, and does not need any other specific control parameters. Therefore, there is no burden of tuning

control parameters in the TLBO algorithm. So, the TLBO algorithm is simpler, more effective, and its computational cost is relatively less. Because the TLBO algorithm has the ability to achieve better results at faster convergence speed than other algorithms mentioned above, it has been successfully applied in many diverse optimization fields [16–20]. Of course, TLBO also has some disadvantages too. Although TLBO has high search accuracy and fast convergence speed, but it has poor exploiting ability and is easy to fall into the local optimum and appeared premature convergence in multimodal function. Therefore, many improved algorithms for TLBO have been presented so far. To improve the performance of TLBO, some variants of the TLBO have been presented. Rao et al. proposed an ETLBO (elitist TLBO) algorithm [15] for solving complex constrained optimization problems, and applied a modified TLBO algorithm [17] to the multi-objective optimization problem. Aiming at neural network training in portable AI (artificial intelligence) devices, Yang et al. [21] proposed the CTLBO (compact teaching–learning-based optimization) algorithm to solve global continuous problems, which can reduce the memory requirement while maintaining the high performance. Wang et al. [22] presented an improved TLBO algorithm. In order to balance the diversity and convergence, an efficient subpopulation was employed in the teacher phase and a ranking differential vector was used in the learner phase. To improve the convergence solution, the neighbor learning and differential mutation is introduced into the basic TLBO by Kumar Shukla et al. [23]. Combining TLBO and ABC, Chen et al. [24] propose a new hybrid teaching–learning-based artificial bee colony (TLABC) to solve the parameters estimation problems of solar photovoltaic. Furthermore, some other improved TLBO algorithms [25–28] have been presented for solving the global function optimization problem. We have even proposed a non-linear inertia weighted teaching–learning-based optimization algorithm (NIWTLBO) [29], which has a fast convergence rate and high accuracy. However, its exploiting ability is weaker.

To enhance the exploiting ability and avoid the premature phenomenon of NIWTLBO, we propose a new improved TLBO variant which is a cyclical non-linear inertia weighted teaching–learning-based optimization algorithm (called as CNIWTLBO). This algorithm uses a cyclical non-linear inertia weight factor to replace the old one to control the memory rate of learners, and employs a non-linear mutation factor to control the learner’s mutation randomly in teacher and learner phase. Experiments are validated on 21 well-known benchmark problems. Simulation results of CNIWTLBO are compared with the original TLBO and other variants of improved TLBO. As a result, the CNIWTLBO has improved the exploitation ability. Furthermore, it has not only faster convergence speed than the basic TLBO, even providing higher search accuracy for most of these benchmark problems.

The rest of this paper is organized as follows. The basic TLBO algorithm is briefly introduced in Section 2. In Section 3, the proposed CNIWTLBO algorithm will be described in detail. Section 4 provides the simulation results and discussions demonstrating the performance of CNIWTLBO in comparison with other optimization algorithms. Finally, the conclusion and future work are summarized in Section 5.

2. Teaching–Learning-Based Optimization

The basic TLBO working is divided into “Teacher Phase” and “Learner Phase”. Learning from the teacher to make the student’s knowledge level closer to the teacher’s is termed as “Teacher Phase”, and learning through interaction with learners to increase their knowledge is called as “Learner Phase”. In TLBO, population is described as a group of learners. Each learner is considered as an individual of the evolutionary algorithm. The subjects offered to the learners are comparable to the different design variables which are the input parameters of the objective function in the optimization problem. The fitness value of the objective function is the learner’s total result. The learner having the best solution of the optimization problem is considered as the teacher in the entire population.

The class $\{X_1, X_2, \dots, X_{NP}\}$ is composed of one teacher and some learners, where $X_i = (X_{i,1}, \dots, X_{i,j}, \dots, X_{i,D})$ ($i = 1, 2, \dots, NP$) denotes the i -th learner, NP is the number of learners (i.e., population size), D represents the number of major subjects in the class (i.e., dimensions of design

variables). $X_{i,j}$ represents the result of the i -th learner on the j -th major subject. The class X (i.e., population) is randomly initialized by a search space bounded by $NP \times D$ matrix whose values between the lower bound and upper bound of design variable.

2.1. Teacher Phase

In the teacher phase, the teacher provides knowledge to the learners to increase the mean result of the class. The learner with the best fitness in the current generation is considered as the teacher $X_{teacher}$, and the mean result of the learners on a particular subject j ($j = 1, 2, \dots, D$) is represented as $M_j = \frac{1}{NP} \left(\sum_{i=1}^{NP} X_{i,j} \right)$. The mean result of a class may increase from a low level to the teacher's level. Due to the individual differences and the forgetfulness of memory, it is impossible for the learners to gain all the knowledge of the teacher to reach the teacher's level. The solution of each learner is updated by Equation (2):

$$X_{i,j}^{new} = X_{i,j}^{old} + r(X_{teacher,j} - T_F M_j) \quad (1)$$

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

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

Since the optimization problem is a minimization problem, so the optimization goal is to find the minimum of objective function f . If the new value gives a better function value in each iteration, then the old value $X_{i,j}^{old}$ is updated with the new value $X_{i,j}^{new}$. The updated formula is given as:

$$\begin{aligned} &\text{if } f(X_i^{new}) < f(X_i^{old}) \\ &\quad X_{i,j}^{old} = X_{i,j}^{new} \\ &\text{end if} \end{aligned} \quad (3)$$

where $f(X_i^{new})$ and $f(X_i^{old})$ represent the new and old total result of the i -th student, respectively. All the new values accepted at the end of the teacher phase become the input to the learner phase.

2.2. Learner Phase

During this phase, the learners increase their knowledge through mutual interaction among themselves. A learner interacts randomly with other learners to achieve new knowledge, so that he can achieve the goal of increasing their knowledge level. In this learner phase, let the i -th learner is X_i and randomly chosen another learner is X_q , $i \neq q$. The updated formula for the new result of the i -th learner is given as:

$$X_{i,j}^{new} = \begin{cases} X_{i,j}^{old} + r(X_{i,j} - X_{q,j}) & \text{if } f(X_i) < f(X_q), \\ X_{i,j}^{old} + r(X_{q,j} - X_{i,j}) & \text{otherwise} \end{cases} \quad (4)$$

where r is a random number between 0 and 1, $f(X_i)$ and $f(X_q)$ are the best solution of the learners X_i and X_q , respectively. If the objective function has the better fitness value with new value, then accept the new value. Similarly, the learner will be updated by using Equation (3).

3. Cyclical Non-Linear Inertia-Weighted Teaching–Learning-Based Optimization (NIWTLBO) Algorithm

3.1. Algorithm Description

In the basic TLBO algorithm, the teacher tries to shift the mean of the learners towards himself or herself by teaching in the teacher phase and the learners improve their knowledge by interaction

among themselves in the learner phase. The learners improve their level by accumulating knowledge in the learning process, i.e., they learn new knowledge based on their existing knowledge. The teacher tends to hope that his students should achieve the knowledge equal to him as soon as possible. But it is impossible because of the forgetting characteristics of the student.

In the NIWTLBO, a phenomenon was described that a student usually forgets a part of existing knowledge due to the physiological phenomena of the brain [29]. Moreover, the learning curve and the forgetting curve presented by Ebbinghaus were introduced. As we know, new knowledge needs to be learned many times before it can be firmly remembered. Over time, we will forget a part of what we have learned. Therefore, we need to review the old knowledge again and again to keep our level of knowledge periodically. In order to simulating this learning process, a cyclical memory weight factor is added to the existing knowledge of the student. This weight factor is non-linear inertia, which control the memory rate of learners cyclically. So we introduce the cyclical non-linear inertia weight factor w_c into Equations (1) and (4) based on the basic TLBO, which scale the existing knowledge of the learner for calculating the new value. Compared with the TLBO algorithm, the previous knowledge accumulation of learners is determined by the weight factor w_c and which is used to calculate new values.

Let T to be the number of iteration of the algorithm in one learning cycle. So the cyclical non-linear inertia weight factor is defined as Equation (5).

$$w_c = 1 - \exp\left(\frac{-(\text{mod}(\text{iter}, T))^2}{2 \times (T/8)^2}\right) \times (1 - w_{cmin}), T \leq \text{MAXITER} \quad (5)$$

where iter is the current iteration number of the algorithm, and MAXITER is the maximum number of allowable iterations which is an integral multiple of the T . The w_{cmin} is the minimum value of cyclical non-linear inertia weight factor w_c and its value should be between 0.5 and 1. The value w_{cmin} should not be too small, otherwise the individuals are worse due to remembering too little existing knowledge in one iteration. If w_{cmin} is too small, it is difficult for the algorithm to converge to the true global optimal solution. In our experiment, the value is 0.6. The w_c is called as cyclical memory rate, and its curve is shown as Figure 1. The cyclical non-linear inertia weight factor w_c is applied to the new equations shown as Equations (7) and (8). During a learning cycle in this improved TLBO, individuals attempt to search diverse areas of the search space at an early stage. In the latter stage, the individuals move in a small range to adjust the trial solution slightly so as to be able to explore relatively small local space. Then repeat the learning cycle over and over again.

Figure 1. The cyclical memory rate curve.

In order to get a new set of better learners (i.e., individuals), the difference between the existing mean result and the corresponding result of the teacher is added to the existing learners in the teacher phase. Similarly, in the learner phase, the difference between the existing result of a learner and

the corresponding result of another learner selected randomly is added to the existing learner. As Equations (1) and (4) show, the value added to the existing learner is formed from the difference of result and the random number r . So the difference value is largely determined by the random number r in the teacher and learner phases. In our proposed method, the random number r in the basic TLBO is modified as follows:

$$r' = \frac{1 + \text{rand}(0, 1)}{2} \quad (6)$$

where $\text{rand}(0, 1)$ is a uniformly distributed random number within the range $[0, 1]$. Equation (6) generates a random number in the range $[0.5, 1]$ which is similar to the method proposed by Satapathy [27]. The r' was called as dynamic inertia weight proposed by Eberhart [30]. Thus, the mean value of the original random number r is increased from 0.5 to 0.75, so the probability of stochastic variations is increased and the difference value added to the existing learners is enlarged. Meanwhile, the w_c increases from small to large in one learning cycle. Under the combined effects of the w_c and r' , the proposed algorithm will not generate premature convergence. Instead, it can improve population diversity, avoid prematurity in the search process and increase the ability of the basic TLBO to escape from local optima. In this way, the algorithm performance is enhanced. On the surface of some multimodal functions, the original random number r may result in some populations clustering near a local optimal point. With the new dynamic inertia weight r' , the population has more chances to escape from the local optima and continuously move towards the global optimum point until reaching the true global optimum.

Now, the cyclical non-linear inertia weight factor and the dynamic inertia weight factor are applied to the basic TLBO algorithm. In the teacher phase, the new set of improved learners can be expressed by the equation as follows:

$$X_{i,j}^{\text{new}} = w_c X_{i,j}^{\text{old}} + r' (X_{\text{teacher},j} - T_F M_j) \quad (7)$$

And in the learner phase, the new set of improved learners can be expressed by the equation as follows:

$$X_{i,j}^{\text{new}} = \begin{cases} w_c X_{i,j}^{\text{old}} + r' (X_{i,j} - X_{q,j}) & \text{if } f(X_i) < f(X_q), \\ w_c X_{i,j}^{\text{old}} + r' (X_{q,j} - X_{i,j}) & \text{otherwise} \end{cases} \quad (8)$$

where w_c is given by Equation (5), and r' is given by Equation (6).

In order to keep the diversity of the algorithm and enhance the global searching ability at the beginning of each iteration cycle, two individuals with the worst solution will be randomly mutated into new individuals. The probability of mutation is expressed by Equation (9). If $P_c > \text{Rand}(0, 1)$ in each iteration, the worst two individuals will be randomly mutated. The mutation process is very simple, that is, the design variables of the two individuals will be initialized randomly in the search space.

$$P_c = 0.5 \exp\left(\frac{-\text{iter}^2}{2 \times (\text{MAXITER}/8)^2}\right) \quad (9)$$

In this way, it will expand the diversity of populations and restrain the prematurity of the algorithm.

3.2. Behavior Parameter Analysis

In the CNIWTLBO algorithm, there are two parameters which are w_{cmin} and T . The w_{cmin} is the minimum value of cyclical non-linear inertia weight factor w_c . Its value is in the range $[0.5, 1]$. So, the w_c is between 0.5 and 1. As is proved in the experiment, the algorithm performance is better when the w_{cmin} value is 0.6 (i.e., the w_c will increase from 0.6 to 1 in one learning cycle). If w_{cmin} is selected at a very small value, the initial value of w_c will become small in one learning cycle which results in the individuals remembering too little existing knowledge in the beginning phase. In this case, it is difficult for the algorithm to converge to the true global optimal solution. Due to the mean value of the r' is increased from 0.5 to 0.75, if w_{cmin} is selected a large value, it will generate premature convergence.

The T is the number of iteration of the algorithm in one learning cycle. For the complex function which is difficult to convergence, it needs 4 learning cycles at least. There is no strict limit to the simple function. The T value depends on the function complexity. The higher the function complexity, the greater the T value. The maximum number of allowable iterations must be an integral multiple of the T . A larger number of experiments proved that the T should be set around 200 which can get better results.

3.3. Framework of CNIWTLBO

The framework of the CNIWTLBO algorithm is described as follows (Algorithm 1):

Algorithm 1 The Framework of CNIWTLBO

- Step 1: Initialize the parameters of the algorithm. Set the number of population (NP , i.e., the number of students), the dimension of decision variables (D , i.e., the number of subjects), the generation number $iter = 1$, and the maximum number of iterations (Maxiter, i.e., the maximum generation number).
 - Step 2: Initialize the population. Generate a random population with NP solutions within the range of specified values, $P = \{X_1, X_2, \dots, X_{NP}\}$. Calculate the population fitness value of the objective function $f(x)$.
 - Step 3: Calculated the cyclical non-linear inertia weight factor w_c and the dynamic inertia weight r' according to the Equations (5) and (6), respectively.
 - Step 4: Choose the individual with the best fitness in the population as the teacher $X_{teacher}$, calculate the average result of each subject M_j .
 - Step 5: Execute the teacher phase. Calculate the new marks of the learners using Equation (7); Evaluate all learners by calculating the fitness value of the objective function and update the old values of the individuals using Equation (3).
 - Step 6: Execute the learner phase. Calculate the new values of the students using Equation (8); Re-calculating the fitness value of the objective function and update the old values of the individuals according to Equation (3) in the same way.
 - Step 7: Execute mutation strategy. Calculate the probability of variation P_c using Equation (9). If $P_c > rand(0, 1)$, then the two individuals with the worst solution will be randomly mutated into new individuals.
 - Step 8: Algorithm Termination: If the terminating condition is satisfied, i.e., $Iter > Maxiter$, then stop the algorithm procedure and output the best solution. Otherwise, go to Step 3.
-

4. Benchmark Tests

In this section, CNIWTLBO is tested on some benchmark functions to evaluate its performance by comparing with the basic TLBO and NIWTLBO, as well as with other optimization algorithms mentioned in literature. All algorithms are coded using the Matlab programming language and run in Matlab 2017a environment on a laptop having Intel core i7 2.60 GHz processor and 8 GB RAM.

In the experiments, 21 well-known benchmark functions with unimodality/multimodality characteristics are adopted. The details of these functions are shown in Table 1. The “C” denotes the characteristic of function; “D” is the dimensions of function; “Range” is the boundary of the variables in each function; And “MinFunVal” is the theoretical global minimum solution.

Table 1. The benchmark functions adopted in the paper.

No.	Function	Formulation	C	D	Range	MinFunVal
<i>f1</i>	Sphere	$f(x) = \sum_{i=1}^D x_i^2$	U	30	[-100, 100]	0
<i>f2</i>	SumSquares	$f(x) = \sum_{i=1}^D ix_i^2$	U	30	[-100, 100]	0
<i>f3</i>	Tablet	$f(x) = 10^6 x_1^2 + \sum_{i=1}^D x_i^2$	U	30	[-100, 100]	0
<i>f4</i>	Schwefel 1.2	$f(x) = \sum_{i=1}^D (\sum_{j=1}^i x_j)$	U	30	[-100, 100]	0
<i>f5</i>	Schwefel 2.22	$f(x) = \sum_{i=1}^D x_i + \prod_{i=1}^D x_i $	U	30	[-10, 10]	0
<i>f6</i>	Schwefel 2.21	$f(x) = \max_{1 \leq i \leq D} \{ x_i \}$	U	30	[-100, 100]	0
<i>f7</i>	Zakharov	$f(x) = \sum_{i=1}^D x_i^2 + (\sum_{i=1}^D 0.5ix_i)^2 + (\sum_{i=1}^D 0.5ix_i)^4$	U	30	[-5, 10]	0
<i>f8</i>	Rosenbrock	$f(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$	U	30	[-4, 4]	0
<i>f9</i>	Schaffer	$f(x) = \frac{\sin^2(\sqrt{x_1^2 + x_2^2}) - 0.5}{(1 + 0.001(x_1^2 + x_2^2))^2} - 0.5$	M	2	[-10, 10]	-1
<i>f10</i>	Dropwave	$f(x) = -\frac{1 + \cos(12\sqrt{x_1^2 + x_2^2})}{0.5(x_1^2 + x_2^2) + 2}$	M	2	[-2, 2]	-1
<i>f11</i>	Bohachevsky1	$f(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) - 0.4 \cos(4\pi x_2) + 0.7$	M	2	[-100, 100]	0
<i>f12</i>	Bohachevsky2	$f(x) = x_1^2 + 2x_2^2 - 0.3 \cos(3\pi x_1) + \cos(4\pi x_2) + 0.3$	M	2	[-100, 100]	0
<i>f13</i>	Six Hump Camel Back	$f(x) = 4x_1^2 - 2.1x_1^4 - \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$ $f(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1 - 3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)]$	M	2	[-5, 5]	-1.03163
<i>f14</i>	Goldstein-Price	$f(x) = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \sum_{i=1}^D x_i^2} \right) - \exp \left(\frac{1}{D} \sum_{i=1}^D \cos(2\pi x_i) \right) + 20 + e$	M	30	[-32, 32]	0
<i>f15</i>	Ackley	$f(x) = -\sum_{i=1}^D (x_i \sin(\sqrt{ x_i }))$	M	30	[-500, 500]	-837.9658
<i>f16</i>	Schwefel 2.26	$f(x) = \sum_{i=1}^D x_i \prod_{i=1}^D x_i $	M	30	[-10, 10]	0
<i>f17</i>	Multimod	$f(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$	M	30	[-5.12, 5.12]	0
<i>f18</i>	Rastrigin	$f(x) = \frac{1}{4000} \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(\frac{x_i}{\sqrt{i}}) + 1$	M	30	[-600, 600]	0
<i>f19</i>	Griewank	$f(x) = \sum_{i=1}^D (y_i^2 - 10 \cos(2\pi y_i) + 10),$ where $y_i = \begin{cases} x_i & x_i < 0.5 \\ 0.5 \text{round}(2x_i) & x_i \geq 0.5 \end{cases}$	M	30	[-5.12, 5.12]	0
<i>f20</i>	NCRastrigin	$f(x) = \sum_{i=1}^D (\sum_{k=0}^{km} [a^k \cos(2\pi b^k(x_i + 0.5))]) - D \sum_{k=0}^{km} [a^k \cos(2\pi b^k 0.5)]$ where $a = 0.5, b = 3, km = 20$	M	30	[-0.5, 0.5]	0
<i>f21</i>	Weierstrass					

U: Unimodal, M: Multimodal.

4.1. CNIWTLBO vs. Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Differential Evolution (DE) and Teaching–Learning-Based Optimization (TLBO)

In order to identify the ability of the proposed algorithm to achieve the global optimum value, 20 different benchmark functions in Table 1 are tested using the PSO, ABC, DE, TLBO and CNIWTLBO algorithms. To maintain the consistency in the comparison, all algorithms are experimented with using the same maximum function evaluations (FES) and the same values of common control parameters such as population size. In general, compared with the other algorithms, the algorithm requiring fewer number of function evaluations to get the same best solution can be considered as better. It should be noted that the FES is significantly different between TLBO variants and other meta-heuristic algorithms, which has been discussed in [28]. So, 2 FES are counted in each iteration for original TLBO,

CNIWTLBO and most of TLBO variants. In this experiment, the maximum fitness function evaluation is 80,000 for all benchmark functions, and the other specific parameters of algorithms are given in Table 2.

Table 2. Parameter settings for particle swarm optimization (PSO), artificial bee colony (ABC), differential evolution (DE), teaching–learning-based optimization (TLBO) and cyclical non-linear inertia-weighted teaching–learning-based optimization (CNIWTLBO) algorithms.

Algorithm	Parameter Settings
PSO	Population size NP = 40, Cognitive attraction $C_1 = 2$, Social attraction $C_2 = 2$, Inertia weight $w = 0.9$
ABC	NP = 40
DE	NP = 40, mutation rate F = 0.5, crossover rate R = 0.4
TLBO	NP = 40
CNIWTLBO	NP = 40, $w_{min} = 0.6$, T = 250

Each benchmark function in Table 1 is independently experimented 30 times with PSO, ABC, DE, TLBO and CNIWTLBO algorithms and comparative results. Each algorithm is terminated after running for 80,000 FEs or when it reached the global minimum value in advance. The results, in the form of mean value and average standard deviation (SD) of objective function obtained after 30 independent runs, are shown in Table 3.

Moreover, the number of function fitness evaluations produced by the experiments when the algorithms reach the true global optimum solution, in the form of mean value and average standard deviation, are reported in Table 4.

It is observed from the comparative results in Table 3 that the performance of CNIWTLBO outperforms PSO, ABC, DE and TLBO for functions $f1$ – $f7$, $f15$, $f16$, $f18$, $f20$. For functions $f9$ – $f14$, the performance of CNIWTLBO, PSO, ABC, DE and TLBO is very similar that all the algorithms can obtain the global optimal value. Moreover, the TLBO is better than PSO, ABC, DE for functions $f1$ – $f7$ and $f15$. The performance of CNIWTLBO, DE and TLBO is similar and better than PSO and ABC for $f17$ (Multimod) and $f19$ (Griewank). For $f8$ (Rosenbrock), the ABC is better than others. For $f21$ (Weierstrass), the performance of NIWTLBO and TLBO is alike and outperform PSO, ABC and DE.

From Table 4 it can be seen that the number of fitness evaluations is smaller, it is indicated that the algorithm obtains the true global optimum value more quickly. In other words, the smaller the number of fitness evaluations, the faster the convergence rate of the algorithm. It is obvious that the CNIWTLBO algorithm requires less number of function evaluations than the basic TLBO, PSO, ABC, DE to achieve the true global optimum value for most of the benchmark functions. Therefore, the convergence rate of the CNIWTLBO algorithm is faster than TLBO, PSO, ABC, DE for most of the benchmark functions in Table 2 except $f13$ (Six Hump Camel Back) and $f14$ (Goldstein-Price).

Table 3. Performance comparisons of PSO, ABC, DE, TLBO, CNIWTLBO in term of fitness value.

No.	PSO (Mean $\pm$ SD)	ABC (Mean $\pm$ SD)	DE (Mean $\pm$ SD)	TLBO (Mean $\pm$ SD)	CNIWTLBO (Mean $\pm$ SD)
<i>f1</i>	8.68E-12 $\pm$ 6.15E-12	9.53E-16 $\pm$ 5.13E-16	7.28E-27 $\pm$ 2.35E-26	3.42E-287 $\pm$ 0.00	0.00E+00 $\pm$ 0.00E+00
<i>f2</i>	2.14E-10 $\pm$ 1.26E-10	7.38E-16 $\pm$ 1.32E-16	8.52E-26 $\pm$ 2.66E-26	8.74E-286 $\pm$ 0.00	0.00E+00 $\pm$ 0.00E+00
<i>f3</i>	3.26E-08 $\pm$ 1.52E-08	8.62E-16 $\pm$ 1.29E-16	3.02E-26 $\pm$ 1.38E-26	6.28E-285 $\pm$ 0.00	0.00E+00 $\pm$ 0.00E+00
<i>f4</i>	2.25E+05 $\pm$ 1.16E+05	7.52E+03 $\pm$ 1.46E+03	2.32E+04 $\pm$ 2.34E+03	2.48E-84 $\pm$ 1.29E-84	0.00E+00 $\pm$ 0.00E+00
<i>f5</i>	5.02E-03 $\pm$ 3.26E-03	2.36E-14 $\pm$ 1.22E-14	4.57E-16 $\pm$ 1.16E-16	1.65E-143 $\pm$ 1.32E-143	4.62e-323 $\pm$ 0.00E+00
<i>f6</i>	1.23E+00 $\pm$ 5.24E-01	3.67E+01 $\pm$ 1.05 E+01	1.19E-02 $\pm$ 2.15E-03	7.68E-120 $\pm$ 3.82E-120	2.64E-315 $\pm$ 0.00E+00
<i>f7</i>	1.58E+02 $\pm$ 4.72E+01	2.45E+02 $\pm$ 2.16E+01	5.63E+01 $\pm$ 6.74E+00	6.04E-51 $\pm$ 4.62E-51	1.82E-319 $\pm$ 0.00E+00
<i>f8</i>	3.05E+01 $\pm$ 2.26E+01	1.12E+01 $\pm$ 2.43E+00	2.51E+01 $\pm$ 3.47E+00	1.32E+01 $\pm$ 4.36E+00	1.78E+01 $\pm$ 5.13E+00
<i>f9</i>	−1.00E+00 $\pm$ 0.00E+00	−1.00E+00 $\pm$ 0.00E+00	−1.00E+00 $\pm$ 0.00E+00	−1.00E+00 $\pm$ 0.00E+00	−1.00E+00 $\pm$ 0.00E+00
<i>f10</i>	−1.00E+00 $\pm$ 0.00E+00	−1.00E+00 $\pm$ 0.00E+00	−1.00E+00 $\pm$ 0.00E+00	−1.00E+00 $\pm$ 0.00E+00	−1.00E+00 $\pm$ 0.00E+00
<i>f11</i>	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00
<i>f12</i>	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00
<i>f13</i>	−1.03163 $\pm$ 0.00	−1.03163 $\pm$ 0.00	−1.03163 $\pm$ 0.00	−1.03163 $\pm$ 0.00	−1.03163 $\pm$ 0.00
<i>f14</i>	3.00 $\pm$ 8.23E-15	3.00 $\pm$ 5.02E-15	3.00 $\pm$ 1.53E-15	3.00 $\pm$ 6.18E-16	3.00 $\pm$ 5.82E-16
<i>f15</i>	1.48E+00 $\pm$ 3.85E-01	3.14E-13 $\pm$ 3.06E-14	2.56E-14 $\pm$ 6.07E-15	4.45E-15 $\pm$ 2.85E-16	8.88E-16 $\pm$ 0.00
<i>f16</i>	−8.79E+03 $\pm$ 4.27E+02	−1.24E+04 $\pm$ 1.81E+02	−1.15E+04 $\pm$ 1.58E+03	−9.18E+03 $\pm$ 7.65E+02	−7.33E+03 $\pm$ 1.66E+02
<i>f17</i>	7.55E-67 $\pm$ 1.63E-66	6.83E-19 $\pm$ 1.62E-19	5.03-311 $\pm$ 0.00	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00
<i>f18</i>	1.12E+02 $\pm$ 2.24E+01	1.32E-13 $\pm$ 2.48E-13	9.02E+01 $\pm$ 8.57E+00	7.21E+00 $\pm$ 5.78E+00	0.00E+00 $\pm$ 0.00E+00
<i>f19</i>	6.69E-03 $\pm$ 5.46E-04	7.15E-03 $\pm$ 6.86E-04	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00
<i>f20</i>	1.78E+02 $\pm$ 3.22E+01	2.01E-14 $\pm$ 1.67E-14	6.82E+01 $\pm$ 8.75E+00	1.48E+01 $\pm$ 2.76E+00	0.00E+00 $\pm$ 0.00E+00
<i>f21</i>	6.13E+01 $\pm$ 2.12E+01	1.36E-02 $\pm$ 8.06E-03	1.38E+01 $\pm$ 5.84E-01	0.00E+00 $\pm$ 0.00E+00	0.00E+00 $\pm$ 0.00E+00

Population Size: **40**, D: **30**(except *f9~f14* 2D), Max. Eval.: **80,000** FEs.

Table 4. Convergence comparisons in term of fitness evaluations.

No.	PSO (Mean $\pm$ SD)	ABC (Mean $\pm$ SD)	DE (Mean $\pm$ SD)	TLBO (Mean $\pm$ SD)	CNIWTLBO (Mean $\pm$ SD)
<i>f1</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	60,720 $\pm$ 2.15E+02
<i>f2</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	58,920 $\pm$ 2.08E+02
<i>f3</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	61,160 $\pm$ 2.26E+02
<i>f4</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	72,280 $\pm$ 1.38E+02
<i>f5</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00
<i>f6</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00
<i>f7</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00
<i>f8</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00
<i>f9</i>	12,315 $\pm$ 3.26E+02	43,502 $\pm$ 3.12E+02	8594 $\pm$ 2.13E+02	9596 $\pm$ 2.04E+02	2840 $\pm$ 2.86E+02
<i>f10</i>	11,406 $\pm$ 3.18E+01	13,749 $\pm$ 1.13E+02	5412 $\pm$ 1.46E+02	3002 $\pm$ 2.32E+02	1215 $\pm$ 3.18E+01
<i>f11</i>	9356 $\pm$ 2.18E+02	3178 $\pm$ 7.85E+01	3916 $\pm$ 8.39E+01	2258 $\pm$ 3.64E+01	1232 $\pm$ 2.13E+01
<i>f12</i>	9503 $\pm$ 1.25E+02	4685 $\pm$ 9.14E+01	4221 $\pm$ 1.14E+02	2556 $\pm$ 2.23E+01	1284 $\pm$ 2.38E+01
<i>f13</i>	1915 $\pm$ 1.29E+02	1357 $\pm$ 1.26E+02	1735 $\pm$ 1.19E+02	718 $\pm$ 6.02E+01	1130 $\pm$ 8.16E+01
<i>f14</i>	2102 $\pm$ 1.32E+02	1838 $\pm$ 1.51E+02	1702 $\pm$ 2.13E+02	1232 $\pm$ 6.67E+01	2810 $\pm$ 2.05E+02
<i>f15</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00
<i>f16</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00
<i>f17</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	28,215 $\pm$ 1.12E+02	8040 $\pm$ 8.44E+01
<i>f18</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	8120 $\pm$ 5.12E+01
<i>f19</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	52,326 $\pm$ 6.21E+02	12,003 $\pm$ 8.73E+02	8160 $\pm$ 3.86E+01
<i>f20</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	8080 $\pm$ 1.22E+02
<i>f21</i>	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	80,000 $\pm$ 0.00	12,625 $\pm$ 1.13E+02	9040 $\pm$ 2.48E+02

Population Size: 40, D: 30(except *f9*~*f14*: 2D), Max. Eval.: 80,000 FEs.

4.2. CNIWTLBO vs. the Variants of PSO

In order to compare the ability of the CNIWTLBO algorithm with other variant PSO algorithms such as PSO-w [31], PSO-cf [32], CPSO-H [33] and CLPSO [34] to obtain the global optimal value, 8 different unimodal and multimodal benchmark functions in Table 1 are tested in this experiment. To maintain the consistency, the CNIWTLBO algorithm and the variants of PSO are performed with the same maximum function evaluations (30,000 FEs) and dimensions (10D). In the same way, CNIWTLBO algorithm is independently run 30 times for each benchmark function. The comparative results obtained after 30 independent runs of the algorithms on each benchmark function, in the form of the mean value and average standard deviation, are shown in Table 5. In this experiment, the results of the algorithms except CNIWTLBO are taken from literature [28,35], and the population size of each algorithm is 10.

Table 5. Comparative results of CNIWTLBO and other variants of PSO.

No.	Function		PSO-w	PSO-cf	CPSO-H	CLPSO	TLBO	CNIWTLBO
<i>f1</i>	Sphere	Mean	7.96E-51	9.84E-105	4.98E-45	5.15E-29	0.00E+00	0.00E+00
		SD	3.56E-50	4.21E-104	1.00E-44	2.16E-28	0.00E+00	0.00E+00
<i>f8</i>	Rosenbrock	Mean	3.08E+00	6.98E-01	1.53E+00	2.46E+00	4.21E+00	6.68E+00
		SD	7.69E-01	1.46E+00	1.70E+00	1.70E+00	6.83E-01	6.32E-01
<i>f15</i>	Ackley	Mean	1.58E-14	9.18E-01	1.49E-14	4.32E-10	3.55E-15	8.44E-16
		SD	1.60E-14	1.01E+00	6.97E-15	2.55E-14	8.32E-31	5.24E-17
<i>f16</i>	Schwefel2.26	Mean	3.20E+02	9.87E+02	2.13E+02	0.00E+00	−4.01E+03	−4.07E+03
		SD	1.85E+02	2.76E+02	1.41E+02	0.00E+00	1.85E+02	1.14E+02
<i>f18</i>	Rastrigin	Mean	5.82E+00	1.25E+01	2.12E+00	0.00E+00	6.77E-08	0.00E+00
		SD	2.96E+00	5.17E+00	1.33E+00	0.00E+00	3.68E-07	0.00E+00
<i>f19</i>	Griewank	Mean	9.69E-02	1.19E-01	4.07E-02	4.56E-03	0.00E+00	0.00E+00
		SD	5.01E-02	7.11E-02	2.80E-02	4.81E-03	0.00E+00	0.00E+00
<i>f20</i>	NCRastrigin	Mean	4.05E+00	1.20E+01	2.00E-01	0.00E+00	2.65E-08	0.00E+00
		SD	2.58E+00	4.99E+00	4.10E-01	0.00E+00	1.23E-07	0.00E+00
<i>f21</i>	Weierstrass	Mean	2.28E-03	6.69E-01	1.07E-15	0.00E+00	2.42E-05	0.00E+00
		SD	7.04E-03	7.17E-01	1.67E-15	0.00E+00	1.38E-20	0.00E+00

Population Size: 10, D: 10, Max. Eval.: 30,000 FEs. Source: Results of algorithms except CNIWTLBO are taken from References [28,35].

From the results in Table 5, it is observed obviously that the performance of CNIWTLBO and TLBO algorithms outperform PSO-w, PSO-cf, CPSO-H and CLPSO algorithms for *f1*(Sphere), *f15*(Ackley) and *f19*(Griewank). The performance of CNIWTLBO and CLPSO is similar for Rastrigin, NCRastrigin and Weierstrass. For Rosenbrock and Schwefel2.26, the CNIWTLBO algorithm does not perform well compared with other algorithms.

4.3. CNIWTLBO vs. the Variants of ABC, DE

The experiment in this section is aimed at identifying the performance of the CNIWTLBO algorithm to achieve the global optimum value by comparing with the variants of ABC and DE on 7 benchmark functions shown in Table 1. The variants of ABC, such as the gbest-guided artificial bee colony GABC algorithm [36] and the improved artificial bee colony IABC algorithm [37], and the variants of DE, such as SaDE and JADE, are used in this experiment. To be fair in the comparison, the parameters of the algorithms are same as in the literature [27], where the population size is 20 and dimension is 30. Alike other algorithms, TLBO and CNIWTLBO are tested with the same function evaluations listed in Table 6. The comparative results, in the form of mean value and average standard deviation, are listed in Table 6. The results of GABC, IABC, SaDE and JADE are taken from the literature [27] directly. The results of TLBO and CNIWTLBO are obtained after 30 independent runs on each benchmark function in the same way.

Table 6. Comparative results of CNIWTLBO and the variants of ABC, DE.

No.	Function		GABC	IABC	SaDE	JADE	TLBO	CNIWTLBO
<i>f1</i>	Sphere FEs: 1.5×10^5	Mean	3.6E-63	5.34E-178	4.5E-20	1.8E-60	0.00E+00	0.00E+00
		SD	5.7E-63	0	1.9E-14	8.4E-60	0.00E+00	0.00E+00
<i>f4</i>	Schwefel 1.2 FEs: 5.0×10^5	Mean	4.3E+02	1.78E-65	9.0E-37	5.7E-61	0.00E+00	0.00E+00
		SD	8.0E+02	2.21E-65	5.4E-36	2.7E-60	0.00E+00	0.00E+00
<i>f5</i>	Schwefel 2.22 FEs: 2.0×10^5	Mean	4.8E-45	8.82E-127	1.9E-14	1.8E-25	0.00E+00	0.00E+00
		SD	1.4E-45	3.49E-126	1.1E-14	8.8E-25	0.00E+00	0.00E+00
<i>f6</i>	Schwefel 2.21 FEs: 5.0×10^5	Mean	3.6E-06	4.98E-38	7.4E-11	8.2E-24	0.00E+00	0.00E+00
		SD	7.6E-07	8.59E-38	1.82E-10	4.0E-23	0.00E+00	0.00E+00
<i>f15</i>	Ackley FEs: 5.0×10^4	Mean	1.8E-09	3.87E-14	2.7E-03	8.2E-10	4.44E-15	8.46E-16
		SD	7.7E-10	8.52E-15	5.1E-04	6.9E-10	2.58E-30	2.15E-31
<i>f18</i>	Rastrigin FEs: 1.0×10^5	Mean	1.5E-10	0.00E+00	1.2E-03	1.0E-04	1.88E+01	0.00E+00
		SD	2.7E-10	0.00E+00	6.5E-04	6.0E-05	4.65E+00	0.00E+00
<i>f19</i>	Griewank FEs: 5.0×10^5	Mean	6.0E-13	0.00E+00	7.8E-04	9.9E-08	0.00E+00	0.00E+00
		SD	7.7E-13	0.00E+00	1.2E-03	6.0E-07	0.00E+00	0.00E+00

Population Size: 20, D: 30, Source: Results of algorithms except TLBO and CNIWTLBO are taken from Reference [27].

It can be observed from the results that the performance of CNIWTLBO performs much better than GABC, SaDE and JADE on all the benchmark functions in Table 6, and outperforms the IABC algorithm for *f1*(Sphere), *f4*(Schwefel 1.2), *f5*(Schwefel 2.22), *f6*(Schwefel 2.21) and *f15*(Ackley). Furthermore, CNIWTLBO is better than TLBO for *f15*(Ackley) and *f18*(Rastrigin). So, it is indicated that the CNIWTLBO algorithm has a good performance.

4.4. CNIWTLBO vs. the Variants of TLBO in Different Dimensions

In order to identify the performance of the CNIWTLBO, the experiments are carried out to compare the CNIWTLBO algorithm with some other variants of TLBO in different dimensions. The variants of the TLBO algorithm including WTLBO [25], ITLBO [26], I-TLBO [28], and NIWTLBO [29] are adopted. In the experiments, 9 unimodal and multimodal benchmark functions listed in Table 1 are used to evaluate the performance of the algorithms. In order to make fair comparisons, the CNIWTLBO and all adopted TLBO variants use the same parameters. In this work, evolutionary generation is used to evaluate the performance of CNIWTLBO and the TLBO variants. The population size is set to 30 and the number of evolutionary generations is set to 2000. In I-TLBO algorithm, the number of teachers is 4. The learning cycle T in CNIWTLBO is 500. The CNIWTLBO and the TLBO variants are tested on 9 benchmark functions with 20, 50 and 100 dimensions, respectively. To eliminate the randomness impact on the results, 30 independent runs for the algorithms are conducted. The experimental results are reported in Table 7, which is in form of the mean solution.

From the Table 7, it could be observed that the TLBO and its different variants perform identically for functions *f1*, *f5*, *f8*, *f15*, *f16* and *f18–f21*. The performance of the CNIWTLBO is better than rest of the algorithms for *f5* (Schwefel2.22), *f15*(Ackley), *f16*(Schwefel2.26) function in 20, 50 and 100 dimensions. Moreover, it outperforms other algorithms for *f20*(NC Rastrigin) in 100 dimensions and *f21*(Weierstrass) in 50 and 100 dimensions. For the *f1*(Sphere), *f18*(Rastrigin) and *f19*(Griewank), the performance of CNIWTLBO and NIWTLBO is similar. But the CNIWTLBO performs worse than other algorithms for the *f8*(Rosenbrock) function.

Table 7. Comparative results of CNIWTLBO and different variants of TLBO algorithms.

No.	Function	D	TLBO	WTLBO	ITLBO	I-TLBO (NT = 4)	NIWTLBO	CNIWTLBO
<i>f1</i>	Sphere	20	1.43E-315	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
		50	1.35E-274	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
		100	4.13E-251	2.85E-316	5.38E-312	0.00E+00	0.00E+00	0.00E+00
<i>f5</i>	Schwefel2.22	20	2.67E-158	3.72E-231	5.36E-182	7.48E-287	2.50E-322	2.47E-323
		50	9.28E-138	5.16E-208	2.87E-168	3.62E-281	4.43E-318	1.04E-322
		100	2.82E-130	2.58E-189	1.94E-152	4.26E-268	7.46E-310	3.41E-316
<i>f8</i>	Rosenbrock	20	1.62E+01	1.83E+01	2.14E+01	1.11E+01	1.08E+01	1.12E+01
		50	4.75E+01	5.02E+01	5.13E+01	4.39E+01	4.52E+01	4.43E+01
		100	9.26E+01	9.78E+01	1.02E+02	9.52E+01	9.63E+01	9.41E+01
<i>f15</i>	Ackley	20	4.44E-15	3.18E-15	3.55E-15	9.12E-16	8.88E-16	8.88E-16
		50	4.44E-15	6.42E-14	5.28E-14	2.45E-15	4.44E-15	8.88E-16
		100	7.99E-15	6.35E-15	6.87E-15	2.44E-15	4.44E-15	8.88E-16
<i>f16</i>	Schwefel2.26	20	−6.49E+03	−6.63E+03	−7.52E+03	−6.21E+03	−5.74E+03	−4.67E+03
		50	−2.05E+04	−1.86E+04	−2.23E+04	−2.06E+04	−1.32E+04	−1.21E+04
		100	−2.35E+04	−2.41E+04	−2.47E+04	−2.36E+04	−2.39E+04	−2.16 E+04
<i>f18</i>	Rastrigin	20	1.98E+00	2.26E-251	4.12E-203	0.00E+00	0.00E+00	0.00E+00
		50	2.38E+01	5.58E-235	1.84E-188	0.00E+00	0.00E+00	0.00E+00
		100	4.83E+01	1.86E-186	3.12E-142	4.58E-322	0.00E+00	0.00E+00
<i>f19</i>	Griewank	20	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
		50	0.00E+00	0.00E+00	4.64E-312	0.00E+00	0.00E+00	0.00E+00
		100	0.00E+00	5.42E-324	2.83E-288	1.87E-323	0.00E+00	0.00E+00
<i>f20</i>	NCRastrigin	20	1.30E+01	3.58E-250	2.67E-239	3.24E-318	0.00E+00	0.00E+00
		50	3.12E+01	1.86E-238	3.18E-225	2.55E-301	0.00E+00	0.00E+00
		100	5.03E+01	2.64E-216	1.83E-209	3.62E-287	1.35E-328	0.00E+00
<i>f21</i>	Weierstrass	20	1.02E+00	2.72E-308	3.65E-285	6.12E-312	0.00E+00	0.00E+00
		50	1.53E+01	2.37E-282	1.83E-256	4.63E-288	3.62E-323	0.00E+00
		100	5.02E+01	4.68E-198	2.66E-186	2.14E-216	2.73E-312	0.00E+00

Population Size: 30, Max. generations: 2000, T: 500.

5. Conclusions

A modified TLBO algorithm called CNIWTLBO has been proposed in this paper for solving global optimization problems. Two learning factors are introduced in the proposed algorithm. A cyclical non-linear inertia weight factor is introduced into the basic TLBO to control the memory rate of learners, and a non-linear mutation factor is introduced into the basic TLBO to control the learner's mutation randomly during the learning process. With the modifications on the basic TLBO, the CNIWTLBO algorithm has stronger exploration capacities. Furthermore, the proposed algorithm is effectively prevented from falling into a local minimum and ensures assessment accuracy. In the experiments, 21 classical benchmark functions have been tested to evaluate the performance of the CNIWTLBO, and the experimental results are compared with the other meta-heuristic algorithms and their variants available in the literatures. Moreover, the comparison results between the CNIWTLBO and the other variants of TLBO are reported in this paper. The experimental results show that the performance of the CNIWTLBO for solving global optimization problems is satisfactory.

In future work, the CNIWTLBO algorithm will be tested on some engineering benchmarks. To verify the efficiency, the proposed method will be applied to handle constrained engineering optimization problems. Furthermore, the hybrid method combining the proposed algorithm and the other classic intelligent algorithms will be researched to improve the performance of the TLBO algorithm.

Author Contributions: Methodology, Z.W.; software, Z.W.; validation, R.X.; data curation, Z.W.; writing—original draft preparation, Z.W.; writing—review and editing, R.X.

Funding: This research was funded by the Natural Science Foundation in Xizang Province of China(No. XZ2018ZR G-64), the Natural Science Foundation of in Shaanxi Province of China (No. 17JK0826), the Science Basic Research

Program in Xianyang Normal University of China (No. XSYK18012) and the Education Scientific Program of 13th Five-year Plan in Shaanxi Province of China (No. SGH18H373).

Conflicts of Interest: The authors declare no conflict of interest.

References

- Thakur, M. A new genetic algorithm for global optimization of multimodal continuous functions. *J. Comput. Sci.* **2014**, *5*, 298–311. [[CrossRef](#)]
- Hussein, H.A.; Demiroglu, I.; Johnston, R.L. Application of a parallel genetic algorithm to the global optimization of medium-sized Au–Pd sub-nanometre clusters. *Eur. Phys. J. B* **2018**, *91*, 34. [[CrossRef](#)]
- Chandra Mohan, B.; Baskaran, R. A survey: Ant colony optimization based recent research and implementation on several engineering domain. *Expert Syst. Appl.* **2012**, *39*, 4618–4627. [[CrossRef](#)]
- Chen, Z.; Wang, R.L. Ant colony optimization with different crossover schemes for global optimization. *Clust. Comput.* **2017**, *20*, 1247–1257. [[CrossRef](#)]
- Ali, M.M.; Kaelo, P. Improved particle swarm algorithms for global optimization. *Appl. Math. Comput.* **2008**, *196*, 578–593. [[CrossRef](#)]
- Liu, P.; Jing, L. Multi-leader PSO (MLPSO): A new PSO variant for solving global optimization problems. *Appl. Soft Comput.* **2017**, *61*, 256–263. [[CrossRef](#)]
- Kang, F.; Li, J.; Ma, Z. Rosenbrock artificial bee colony algorithm for accurate global optimization of numerical functions. *Inf. Sci.* **2011**, *181*, 3508–3531. [[CrossRef](#)]
- Karaboga, D.; Gorkemli, B.; Ozturk, C.; Karaboga, N. A comprehensive survey: Artificial bee colony (ABC) algorithm and applications. *Artif. Intell. Rev.* **2014**, *42*, 21–57. [[CrossRef](#)]
- You, X.; Ma, Y.; Liu, Z.; Xie, M. An ABC Algorithm with Recombination. *Int. J. Comput. Commun. Control* **2018**, *13*, 590–601. [[CrossRef](#)]
- Mohamed, A.K.; Mohamed, A.W.; Elfeky, E.Z.; Saleh, M. Solving constrained non-linear integer and mixed-integer global optimization problems using enhanced directed differential evolution algorithm. In *Machine Learning Paradigms: Theory and Application*; Springer: Cham, Switzerland, 2019.
- Mohamed, A.W. Solving large-scale global optimization problems using enhanced adaptive differential evolution algorithm. *Complex Intell. Syst.* **2017**, *3*, 205–231. [[CrossRef](#)]
- Nouioua, M.; Li, Z. Using differential evolution strategies in chemical reaction optimization for global numerical optimization. *Appl. Intell.* **2017**, *47*, 935–961. [[CrossRef](#)]
- Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching–learning-based optimization: A novel method for constrained mechanical design optimization problems. *Comput.-Aided Des.* **2011**, *43*, 303–315. [[CrossRef](#)]
- Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching–learning-based optimization: An optimization method for continuous non-linear large scale problems. *Inf. Sci.* **2012**, *183*, 1–15. [[CrossRef](#)]
- Rao, R.V.; Patel, V. An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems. *Int. J. Ind. Eng. Comput.* **2012**, *3*, 535–560. [[CrossRef](#)]
- Venkata Rao, R.; Kalyankar, V.D. Parameter optimization of modern machining processes using teaching–learning-based optimization algorithm. *Eng. Appl. Artif. Intell.* **2013**, *26*, 524–531. [[CrossRef](#)]
- Rao, R.V.; Patel, V. Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm. *Appl. Math. Model.* **2013**, *37*, 1147–1162. [[CrossRef](#)]
- Shabanpour-Haghighi, A.; Seifi, A.R.; Niknam, T. A modified teaching–learning based optimization for multi-objective optimal power flow problem. *Energy Convers. Manag.* **2014**, *77*, 597–607. [[CrossRef](#)]
- Sultana, S.; Roy, P.K. Multi-objective quasi-oppositional teaching learning based optimization for optimal location of distributed generator in radial distribution systems. *Int. J. Electr. Power Energy Syst.* **2014**, *63*, 534–545. [[CrossRef](#)]
- Ghasemi, M.; Ghavidel, S.; Gitizadeh, M.; Akbari, E. An improved teaching–learning-based optimization algorithm using Lévy mutation strategy for non-smooth optimal power flow. *Int. J. Electr. Power Energy Syst.* **2015**, *65*, 375–384. [[CrossRef](#)]
- Yang, Z.; Li, K.; Guo, Y.; Ma, H.; Zheng, M. Compact real-valued teaching-learning based optimization with the applications to neural network training. *Knowl.-Based Syst.* **2018**, *159*, 51–62. [[CrossRef](#)]
- Wang, B.C.; Li, H.X.; Feng, Y. An Improved teaching-learning-based optimization for constrained evolutionary optimization. *Inf. Sci.* **2018**, *456*, 131–144. [[CrossRef](#)]

23. Kumar Shukla, A.; Singh, P.; Vardhan, M. Neighbour teaching learning based optimization for global optimization problems. *J. Intell. Fuzzy Syst.* **2018**, *34*, 1583–1594. [[CrossRef](#)]
24. Chen, X.; Xu, B.; Mei, C.; Ding, Y.; Li, K. Teaching-learning-based artificial bee colony for solar photovoltaic parameter estimation. *Appl. Energy* **2018**, *212*, 1578–1588. [[CrossRef](#)]
25. Satapathy, S.C. Weighted Teaching-learning-based optimization for global function optimization. *Appl. Math.* **2013**, *4*, 429–439. [[CrossRef](#)]
26. Chen, D.; Zou, F.; Li, Z.; Wang, J.; Li, S. An improved teaching-learning-based optimization algorithm for solving global optimization problem. *Inf. Sci.* **2015**, *297*, 171–190. [[CrossRef](#)]
27. Satapathy, S.C. Improved teaching learning based optimization for global function optimization. *Decis. Sci. Lett.* **2013**, *2*, 23–34. [[CrossRef](#)]
28. Rao, R.V.; Patel, V. An improved teaching-learning-based optimization algorithm for solving unconstrained optimization problems. *Sci. Iran.* **2013**, *20*, 710–720. [[CrossRef](#)]
29. Wu, Z.-S.; Fu, W.-P.; Xue, R. Nonlinear Inertia weighted teaching-learning-based optimization for solving global optimization problem. *Comput. Intell. Neurosci.* **2015**, *2015*, 292576. [[CrossRef](#)]
30. Eberhart, R.C.; Yuhui, S. Tracking and optimizing dynamic systems with particle swarms. In Proceedings of the 2001 Congress on Evolutionary Computation, Seoul, Korea, 27–30 May 2001; pp. 94–100.
31. Shi, Y.; Eberhart, R.C. A modified particle swarm optimizer. In Proceedings of the 1998 IEEE International Conference on Evolutionary Computation Proceedings, IEEE World Congress on Computational Intelligence, Anchorage, AK, USA, 4–9 May 1998; pp. 69–73.
32. Clerc, M.; Kennedy, J. The particle swarm—Explosion, stability, and convergence in a multidimensional complex space. *IEEE Trans. Evolut. Comput.* **2002**, *6*, 58–73. [[CrossRef](#)]
33. Frans, V.D.B.; Engelbrecht, A.P. A Cooperative approach to particle swarm optimization. *IEEE Trans. Evolut. Comput.* **2004**, *8*, 225–239.
34. Liang, J.J.; Suganthan, P.N.; Qin, A.K.; Baska, S. Comprehensive Learning particle swarm optimizer for global optimization of multimodal functions. *IEEE Trans. Evolut. Comput.* **2006**, *10*, 281–295. [[CrossRef](#)]
35. Akay, B.; Karaboga, D. A modified artificial bee colony algorithm for real-parameter optimization. *Inf. Sci.* **2012**, *192*, 120–142. [[CrossRef](#)]
36. Zhu, G.P.; Kwong, S. Gbest-guided artificial bee colony algorithm for numerical function. *Appl. Soft Comput.* **2010**, *10*, 445–456. [[CrossRef](#)]
37. Gao, W.; Liu, S. Improved artificial bee colony algorithm for global optimization. *Inf. Proc. Lett.* **2011**, *111*, 871–882. [[CrossRef](#)]

© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).