AIMS Mathematics, 6(2): 1395–1442.
DOI: 10.3934/math.2021087
Received: 06 September 2020
Accepted: 21 October 2020
Published: 19 November 2020

http://www.aimspress.com/journal/Math

Research article

TLMPA: Teaching-learning-based Marine Predators algorithm

Keyu Zhong[1], Qifang Luo[1,2,3,*], Yongquan Zhou[1,2,3] and Ming Jiang[4]

[1] College of Artificial Intelligenc, Guangxi University for Nationalities, Nanning 530006, China
[2] School of Computer and Electronics and Information, Guangxi University, Nanning 530004, China
[3] Guangxi Key Laboratories of Hybrid Computation and IC Design Analysis, Nanning 530006, China
[4] Guangxi Institute of Digital Technology, Nanning 530000, China

* **Correspondence:** Email: l.qf@163.com.

Abstract: Marine Predators algorithm (MPA) is a newly proposed nature-inspired metaheuristic algorithm. The main inspiration of this algorithm is based on the extensive foraging strategies of marine organisms, namely Lévy movement and Brownian movement, both of which are based on random strategies. In this paper, we combine the marine predator algorithm with Teaching-learning-based optimization algorithm, and propose a hybrid algorithm called Teaching-learning-based Marine Predator algorithm (TLMPA). Teaching-learning-based optimization (TLBO) algorithm consists of two phases: the teacher phase and the learner phase. Combining these two phases with the original MPA enables the predators to obtain prey information for foraging by learning from teachers and interactive learning, thus greatly increasing the encounter rate between predators and prey. In addition, effective mutation and crossover strategies were added to increase the diversity of predators and effectively avoid premature convergence. For performance evaluation TLMPA algorithm, it has been applied to IEEE CEC-2017 benchmark functions and four engineering design problems. The experimental results show that among the proposed TLMPA algorithm has the best comprehensive performance and has more outstanding performance than other the state-of-the-art metaheuristic algorithms in terms of the performance measures.

Keywords: Marine Predators algorithm; Teaching-learning-based optimization; mutation and crossover; hybrid metaheuristic algorithm
Mathematics Subject Classification: 68W50, 68T20, 90C59

1. Introduction

In the past few decades, natural heuristic algorithms have shown superiority compared with traditional optimization methods when solving complex nonlinear real-world problems. Natural

heuristic algorithms solve a wide range of problems by simulating various natural phenomena. Some of the most common and widely used natural heuristic algorithms are as follows: the genetic algorithm (GA) [1] simulates the evolutionary process of biological population genetics, mutation and natural selection. Particle swarm optimization (PSO) [2] simulates the predation behavior of birds in nature. Differential evolution algorithm (DE) [3, 4] is derived from GA, but the former has special crossover and selection methods. DE is a very powerful evolution algorithm, which has been widely used in chemical engineering, electric power, mechanical design, control engineering, robotics, and artificial neural networks, signal processing, data mining, biology, operations research, scheduling problems and other fields. Artificial bee colony algorithm (ABC) [5] simulates the honey-collecting behavior of bees. Simulated Annealing Algorithm (SA) [6] simulates the high temperature annealing liquid crystallization process of metal materials. In recent years, some new and effective algorithms have been proposed. For example, Marine Predator algorithm (MPA) [7] simulates the predatory behavior of marine predators. Grey wolf optimizer (GWO) [8] simulates the grey wolf population hierarchy and predation behavior in nature. Whale optimization algorithm (WOA) [9] simulates the predation behavior of humpback whale groups in nature. Salp swarm algorithm (SSA) [10] simulates the clustering and foraging behavior of salp swarm. The Teaching-learning-based optimization (TLBO) [11] simulates the process of teacher's work affecting learners.

In the natural heuristic algorithm, according to the principle of "no free lunch" [12], an algorithm A shows a very ideal performance on a specific problem set, but there is always another set problem, and algorithm A performs poorly on this problem set. Therefore, this theorem allows researchers to constantly propose new algorithms or improve on existing algorithms. As for the improvement of existing algorithms, the combination of two or more algorithms is a good strategy to realize the algorithm blending by combining the excellent characteristics of various algorithms. In [13], a novel hybrid whale optimization enhanced with Lévy flight and differential evolution algorithm was presented to solve job shop scheduling problems. Tansui et al. [14] proposed a hybrid hydrozoa and sea turtle foraging algorithms for solving continuous optimization problems. An effective hybrid between gravitational search algorithm and genetic algorithm for constrained optimization problems was introduced in [15]. Le et al. [16] proposed a novel hybrid electromagnetism-like algorithm with firefly algorithm to solve discrete structural optimization. It is worth noting that, compared with the original algorithm; the hybrid algorithm mentioned above has been proved to have higher efficiency. In view of the effectiveness of the hybrid algorithm, this paper mixes the marine predator algorithm with a Teaching-learning-based optimization algorithm to combine better exploration gains and maintain a balanced efficiency between exploration and exploitation.

MPA is a new algorithm proposed by Faramarzi et al., which is inspired by the predation behavior of marine predators, such as sharks, monitor lizards, sunfish, equine fishes and swordfish, etc. [17]. Marine Predators algorithm mainly consists of three phases. In phase 1, the prey moves faster than the predator, the predator adopts Brownian movement as its predation strategy. Phase 2 is under the unit speed ratio or when the predator and prey move at almost the same speed, the strategy adopted by the predator is to carry out Lévy movement and Brownian movement simultaneously. Half of the predators of the population carry out Lévy movement and the other half carry out Brownian movement. In phase 3, the predator moves faster than the prey, the strategy adopted by the predator is Lévy movement.

The MPA mainly imitates the Lévy movement and Brownian movement adopted by marine

organisms during predation [18], both of which are based on random strategies, leading to certain blindness of predators during predation. In this paper, the TLBO algorithm is combined with the marine predator algorithm. Predators can utilize the knowledge gained from teacher and students' self-study to hunt, which greatly improves the encounter rate between predators and prey and makes it easier for predators to hunt prey.

Since the MPA was proposed, there have been some preliminary studies on the MPA. For example, AI-Qaness et al. [19] used the Marine Predators algorithm to predict confirmed COVID-19 cases in Italy, the US, Iran and South Korea. Dalia et al. [20] applied the marine predator algorithm to improve the performance of photovoltaic system. Abdel-Basset et al. [21] proposed a hybrid detection model of COVID-19 using an improved MPA and a sort-based diversity reduction strategy. Mohamed et al. [22] used the improved MPA for image segmentation with multi-level threshold. Although the MPA has many advantages, such as fewer parameters, simple setting, easy to implement and accurate calculation, it also has some disadvantages, such as local optimization and premature convergence. For the MPA, the hybrid of two or more algorithms is still a little work. In this paper, TLBO algorithm is hybridized with the MPA, and the mutation and crossover strategy of differential evolution is added, which greatly improves the performance of the basic MPA.

Teaching-learning-based optimization algorithm simulates the process of teacher's teaching influencing students' learning. It was proposed by Rao et al. in 2011. The TLBO process is divided into two parts: the first part is the teacher phase, in which students learn from the teacher; and the second part is the learner phase, which is the interactive learning between learners. Due to the effectiveness and interest of the TLBO algorithm, there are many related researches on its improvement and application. For the improvement of the algorithm, Rao and Patel proposed modified TLBO for solving for the multi-objective optimization of heat exchangers [23]. Rao and Patel proposed improved TLBO for solving unconstrained optimization problems [24]. Yildiz proposed a hybrid TLBO with Taguchi's method for optimization problems of manufacturing area [25]. In [26], a hybrid TLBO with differential evolution algorithm was presented to solve numerical and engineering optimization problems. Uzlu et al. [27] proposed a hybrid ANN with TLBO to estimate energy consumption in Turkey. For the application of TLBO algorithm, Toğan V has utilized TLBO to optimize the design problem of planar steel frames [28]. In [29], Teaching-learning-based optimization algorithm was applied for the process parameter optimization of selected modern machining processes. The application of Teaching-learning-based optimization algorithm for optimal coordination of DOCR relays in a looped power system was introduced in [30]. In [31], the teaching-learning based optimization technique was used to solve the optimal power flow problem. Since the TLBO algorithm exhibits strong search performance in various applications, this paper considers combining TLBO with MPA, Enhance the search performance of MPA through the effective global search of TLBO.

Based on the characteristics of the MPA and TLBO algorithms, we propose a Teaching-learning-based Marine Predator algorithm (TLMPA), which enables marine predators to learn more intelligently in the process of predation, so as to obtain more location information of their prey. The encounter rate between predators and prey has been greatly increased. The core idea of this hybrid technology is to make full use of the good global search capability of TLBO and the fast convergence capability of MPA. In addition, the cross mutation strategy of the differential evolution algorithm is added to increase the diversity of the population, make up for the possible loss of

diversity in TLBO, effectively avoid the phenomenon of premature convergence, and achieve a better balance between algorithm exploration and development.

Major contributions of this paper can be summarized as follows:

1. A hybrid meta-heuristic algorithm using Teaching-learning-based optimization and Marine Predators is proposed.

2. TLMPA combines the global optimization capability of TLBO and the fast convergence capability of MPA. In addition, the cross mutation strategy of differential evolution is introduced, which effectively avoids the occurrence of premature convergence and better balances the exploration and exploitation capabilities.

3. In order to verify the effectiveness of TLMPA, 29 benchmark functions of IEEE CEC-2017 are used in this paper to evaluate the search accuracy and statistical performance of the algorithm.

4. The TLMPA algorithm proposed in this paper has been used to solve four constrained engineering optimization problems, such as: (a) welded beam design problem; (b) multi-plate disc clutch brake design problem; (c) pressure vessel design problem; (d) tension/compression spring design problem.

The rest of the paper is organized as follows: In Section 2, the overview of MPA and TLBO is presented. Section 3 discusses the proposed TLMPA algorithm in detail. Simulation experiments and result analysis are carried out in Section 4 and Section 5. In addition, Section 6 explains the limitations of this research and future research. Finally, Section 7 is the conclusion of the paper.

2. Marine Predators and Teaching-learning-based optimization algorithm

2.1. Marine Predators algorithm (MPA)

The MPA is a new natural heuristic algorithm. The algorithm is inspired by the characteristics of predators and prey in nature. In marine life, many species, including sharks, monitor lizards, sunfish, equine fishes and swordfish; exhibit Lévy's behavior in their search for prey [32]. Faramarzi et al. proposed the Marine Predators algorithm (MPA). The initial solution of MPA is uniformly distributed in the search space, just like the initialization steps of most meta-heuristic algorithms. The initialization formula is as follows:

$$X_0 = X_{\min} + rand\,(X_{\max} - X_{\min}). \tag{1}$$

where X_{min} and X_{max} are the lower and upper limits of variables, and rand is a random vector between 0 and 1.

While the predator is looking for food, the prey is also looking for food. Therefore, two matrices need to be defined. The fittest solution is nominated as a top predator to construct a matrix which is called *Elite*, an array of which monitors the search and search for prey based on the location information of the prey. The second matrix is the *Prey* matrix, which has the same dimension as Elite, and the predator updates its position based on this matrix. The definition of *Elite* and *Prey* is as follows:

$$Elite = \begin{bmatrix} X_{11}^{I} & X_{12}^{I} & \ldots & X_{1d}^{I} \\ X_{21}^{I} & X_{22}^{1} & \ldots & X_{2d}^{I} \\ \ldots & \ldots & \ldots & \ldots \\ X_{n1}^{I} & X_{n2}^{I} & \ldots & X_{nd}^{I} \end{bmatrix} \tag{2}$$

$$Prey = \begin{bmatrix} X_{11} & X_{12} & ... & X_{1d} \\ X_{21} & X_{22} & ... & X_{2d} \\ ... & ... & ... & ... \\ X_{n1} & X_{n2} & ... & X_{nd} \end{bmatrix} \tag{3}$$

where X^I represents the optimal predator vector, n is the number of search agents and d is the number of dimensions. Both predators and prey are considered search agents. And $X_{i,j}$ is the *jth* dimension of the *ith Prey*. The details of each step of the marine predator algorithm are as follows:

(1) ***Phase 1***: High-velocity ratio or when the prey moves faster than the predator. This stage takes place in the first 1/3 of the total number of iterations of the algorithm, during which the exploration of the algorithm is performed. In this stage, the prey moves faster than the predator, so the predator adopts a waiting strategy, monitoring the movement of the prey from the spot. The prey position update formula is as follows:

$$S_i = R_B \otimes (Elite_i - R_B \otimes \Pr ey_i), i = 1, ..., n \tag{4}$$

$$Prey_i = Prey_i + P.R \otimes S_i \tag{5}$$

where $R \in [0, 1]$ is a random vector extracted from a uniform distribution, $P = 0.5$ is a constant number, R_B is a vector containing random numbers based on normal distribution representing the Brownian movement, $\otimes$ is the multiplication of elements. S_i represents the step size of ith prey's next move. The multiplication of R_B by *Prey* simulates the Brownian movement of prey.

(2) ***Phase 2***: In unit velocity ratio or when both predator and prey are moving at the same pace. This phase occurs in the intermediate phase of the algorithm, when exploration begins to transform to exploitation, and both exploration and exploitation are carried out. In this phase, the prey and the predator move at almost the same speed, with the predator following the Brownian movement and the prey following the Lévy movement. In this phase, both exploration and exploitation matters. Consequently, half of the population is designated for exploration and the other half for exploitations. In this phase, prey is responsible for exploitation and predator for exploration.

$$\begin{aligned} &S_i = R_L \otimes (Elite_i - R_L \otimes Prey_i), i = 1, ..., n/2 \\ &Prey_i = Prey_i + P.R \otimes S_i \end{aligned} \tag{6}$$

where R_L refers to Levi's movement and is a random number vector, and the multiplication of R_L by *Prey* simulates the Lévy movement of prey. Eq 6 is responsible for the first half of the exploration. Eq 7 is responsible for the exploitation of the second half.

$$\begin{aligned} &S_i = R_B \otimes (R_B \otimes Elite_i - Prey_i), i = n/2, ..., n \\ &Prey_i = Elite_i + P.CF \otimes S_i \end{aligned} \tag{7}$$

while $CF = \left(1 - \frac{Iter}{Max\_Iter}\right)^{\left(2\frac{Iter}{Max\_Iter}\right)}$ is considered as an adaptive parameter to control the moving step size of predators. The multiplication of R_B by *Elite* simulates the Brownian movement of predators.

(3) ***Phase 3***: In low-velocity ratio or when predator is moving faster than prey. This phase occurs in the last third of the total number of iterations of the algorithm and belongs to the final phase of the algorithm. In this phase, the prey moves more slowly than the predator, and the predator's strategy is lévy movement. The predator location update formula is as follows:

$$\begin{gathered} S_i = R_L \otimes \left(R_L \otimes Elite_i - Prey_i\right), i = 1, ..., n \\ Prey_i = Elite_i + P.CF \otimes S_i \end{gathered} \tag{8}$$

where the multiplication of R_L and $Elite$ simulates the Lévy movement of predators. S_i represents the step size of ith predator's next move.

(4) ***Eddy formation and FADs' effect***

Environmental issues have a great influence on the predation of marine predators. For example, the formation of eddy currents and the action of fish gathering devices ($FADs$) [33] will change the predation behavior of predators. These effects can be expressed mathematically as:

$$Prey_i = \begin{cases} Prey_i + CF\left[X_{\min} + R \otimes \left(X_{\max} - X_{\min}\right)\right] \otimes U & if \ \ r \leq FADs \\ Prey_i + \left[FADs\left(1 - r\right) + r\right]\left(Prey_{r1} - Prey_{r2}\right) & if \ \ r > FADs \end{cases} \tag{9}$$

where $FADs = 0.2$, U is the binary vector with arrays including zero and one. If the random solution is less than 0.2, it is converted to 0, if the random solution is greater than 0.2, it is converted to 1, $r \in [0, 1]$ represents a random number, X_{min} and X_{max} are the vectors containing the lower and upper bounds of the dimensions, r_1 and r_2 are random indices of the prey matrix. When $r \geq FADs$, predators will take a longer jump in different dimensions probably to find an environment with another prey distribution. And when $r > FADs$, the predator will randomly move within the current predator space.

Algorithm 1: pseudo of MPA

1. Initialize search agents (Prey) populations
2. ***while*** termination criteria are not met
3. Calculate the fitness, construct the Elite matrix and accomplish memory saving
4. ***if*** $Iter < Max\_Iter/3$
5. Update prey based on Eq 4 and Eq 5
6. ***else if*** $Max\_Iter/3 < Iter < 2 * Max\_Iter/3$
7. For the first half of the populations($i = 1, ..., n/2$)
8. Update prey based on Eq 6
9. For the other half of the populations
10. Update prey based on Eq 7
11. ***else if*** $Iter > 2 * Max\_Iter/3$
12. Update prey based on Eq 8
13. ***end if***
14. Accomplish memory saving and Elite update
15. Applying FADs effect and update based on Eq 9
16. ***end while***

(5) ***Marine memory***

Marine Predators have good memories to remind them of successful foraging places [34–36]. By saving the optimal solution of the previous iteration, the current solution is compared with the historical optimal solution, and if the current solution is more appropriate, the current solution is replaced. The pseudo-code for the MPA is shown in Algorithm 1.

2.2. Teaching-learning-based optimization algorithm

Teaching and learning is an important process of individual growth. Empirical knowledge can be acquired by learning from teachers or by learning through interaction between learners. Rao et al. [11] proposed the Teaching-learning-based optimization algorithm based on this. This algorithm consists of two important parts, namely teacher phase and learner phase, and two basic learning methods. In the teacher phase, learners learn from the teacher. In the learning phase, learners interact with each other. In TLBO, learners' learning results are similar to fitness values, and teachers are considered to be the best solution achieved so far. The algorithm is carried out by two basic operations in the teacher phase and the learner phase. The algorithm steps are described in detail below.

1) ***Teacher phase***

This phase is the initial phase of the algorithm, in which students improve their knowledge with the help of the teacher, who is the most knowledgeable person in the class, and always motivates the learners to acquire knowledge in this phase. Teachers try to use their anger to improve the subject average of learners. M_i is assumed to be the subject average score of learners, and T_i is the teacher. Teachers try to approach M_i towards their ability level, and the average score of learner changes with the efforts of teachers, resulting in a new mean M_{new}. Update the solution based on the difference between the existing mean and the new mean. The mean difference is calculated as follows:

$$\text{Diff\_Mean}_\text{i} = r_i \left(M_{new} - T_F M_i\right) \tag{10}$$

where TF is the teaching factor that changes the mean value and the value can be 1 or 2, is a random number between 0 and 1, M_i is the mean value of the learner, M_{new} represents the best learner. To update the solution according to the obtained mean difference, the update formula is as follows:

$$X_{new,i} = X_{old,i} + Diff\_Mean_i \tag{11}$$

where $X_{new,i}$ is updated by $X_{old,i}$, if $X_{new,i}$ has a better fitness value, $X_{old,i}$ will be replaced with $X_{new,i}$.

2) ***Learner phase***

In this phase, learners interact randomly with other learners through their interactions, such as group discussions, demonstrations, formal communication, etc. If another learner has more knowledge than he or she, then the learner will learn new knowledge. The expression updated by the learner is as follows:

$$X_{new,i} = \begin{cases} X_{old,i} + r_i\left(X_i - X_j\right) & if \;\; f\left(X_i\right) < f\left(X_j\right) \\ X_{old,i} + r_i\left(X_j - X_i\right) & if \;\; f\left(X_i\right) > f\left(X_j\right) \end{cases} \tag{12}$$

Accept that if the $X_{new,i}$ result is better than the existing solution, i and j are the indexes of two different learners, r_i is a random value among (0, 1). The pseudo-code of TLBO algorithm is shown in algorithm 2.

Algorithm 2: pseudo of TLBO
1. Initialize search agents (students) populations
2. Calculate the mean of each design variables
3. Identify the best solution(teacher)
4. ***while*** $Iter < Max\_Iter$
5. ***for*** $i = 1 : n$
6. Update the solution based on Eq 10 and Eq 11
7. Select the one with better fitness between the existing solution and the new solution
8. ***end for***
9. ***for*** $i = 1 : n$
10. Update the solution based on Eq 12
11. Select the one with better fitness between the existing solution and the new solution
12. ***end for***
13. ***end while***

3. The proposed Teaching-learning-based Marine Predators algorithm

Marine creatures often adopt random foraging strategies when foraging. They have a good memory to save their successful foraging position, which increases the chance of their next foraging success, but this is only slightly increased the chance of success. In most cases, they tend to adopt a random foraging strategy. The Teaching-learning-based Marine Predators algorithm proposed in this paper combines the TLBO algorithm with the MPA, and adds the mutation and crossover strategies of differential evolution, so that the marine predators can acquire more knowledge about the prey through continuous learning, greatly increasing the encounter rate of marine predators and prey, greatly enhancing the success rate of predation. Fusion of mutation and crossover strategies can avoid the phenomenon of premature convergence of the algorithm.

In phase 1 of MPA, the marine predator follows Brownian movement. At this time, the predation movement is irregular. In this phase, TLMPA combining the teaching phase of TLBO algorithm, the predator obtains prey information by learning from the top predator. Because the top predator is always more knowledgeable than the average predator, knowing where there are more prey. The top predators always try to increase their average predation ability by moving the general predators towards their own ability level. The mathematical model of phase 1 is as follows:

$$\begin{aligned} &\text{While} \quad \text{Iter} < \frac{1}{3}\text{Max\_Iter} \\ &\text{Diff\_Mean}_{\text{i}} = R_B\left(M_{new} - T_F M_i\right) \\ &\text{Prey}_i = \text{Prey}_i + \text{Diff\_Mean}_{\text{i}} \end{aligned} \tag{13}$$

where M_i is assumed to be the subject average score of learners (predators), and T_i is the teacher (top predator). T_F is a teaching factor of 1 or 2. M_{new} denotes the best learner. R_B is a vector containing random numbers based on normal distribution representing the Brownian movement.

In phase 2 of MPA, as mentioned earlier, half of the population follows Brownian movement. The other half of the population follows the Lévy movement. In his phase, TLMPA is combined with the teaching and learning phase of TLBO algorithm. The corresponding process of the proposed algorithm is that half of the predators continue to learn from the top predators, and the other half of the predators learn independently (after the first phase of learning, there is a certain knowledge base). This phase completes the transition from learning to the top predators to autonomous learning. The mathematical model of phase 2 is as follows:

$$\begin{aligned}
&\text{While} \quad \frac{1}{3}\text{Max_Iter} < \text{Iter} < \frac{2}{3}Max_Iter \\
&\text{For the first half of population,} \quad \text{i} = 1,..., \text{n}/2 \\
&\text{Prey}_\text{i} = \begin{cases} \text{Elite}_\text{i} + R_L\left(X_i - X_j\right) & ,if \;\; f(X_i) < f(X_j) \\ \text{Elite}_\text{i} + R_L\left(X_j - X_i\right) & ,otherwise \end{cases} \\
&\text{For the second half of population,} \quad \text{i} = \text{n}/2,..., \text{n} \\
&\text{Diff_Mean}_\text{i} = R_B\left(M_{new} - T_F M_i\right) \\
&\text{Prey}_i = \text{Prey}_i + \text{Diff_Mean}_\text{i}
\end{aligned} \tag{14}$$

where r_i is a random value among (0,1), i and j are the indexes of two different learners. $Prey_i$ will be accepted if it has a better fitness value. R_L is a vector containing random numbers based on Lévy distribution representing the Lévy movement.

In phase 3 of MPA, the marine predator follows the Lévy movement. In this phase, the TLMPA combines the learner phase of the TLBO algorithm. After the previous phase, the predator completes the conversion from learning to the top predators to autonomous learning. In phase 3, the predator will perform completely autonomous learning. The mathematical model of phase 3 is as follows:

$$\begin{aligned}
&\text{While} \quad \text{Iter} > \frac{2}{3}\text{Max_Iter} \\
&\text{Prey}_\text{i} = \begin{cases} \text{E}lite_\text{i} + R_L\left(X_i - X_j\right) & ,if \;\; f(X_i) < f(X_j) \\ \text{E}lite_\text{i} + R_L\left(X_j - X_i\right) & ,otherwise \end{cases}
\end{aligned} \tag{15}$$

where $i \neq j$, i and j are the indexes of two different learners. $Prey_i$ will be accepted if it has a better fitness value.

This paper also combines the mutation and crossover strategies of differential evolution algorithm, mainly to solve the loss of population diversity in the teacher phase or in the search process. When the population converges to the local optimum of the objective function, when the initial algorithm progresses slowly or does not proceed at all, or when the population loses diversity, premature convergence occurs. Mutation and crossover are effective strategies to increase population diversity and prevent premature convergence. The formula of mutation strategy is as follows:

$$V_i = X_{r1} + F\left(X_{r2} - X_{r3}\right) \tag{16}$$

where r_1, r_2 and r_3 are integers different from i randomly selected from 1 to n, i is the current individual, which represents the number of individuals. The parameter $F \in [0, 2]$ is the shrinkage factor, which

is used to control the amplification of the differential vector $X_{r2} - X_{r3}$. After the mutation phase is completed, the crossover operation is performed on each predator and its corresponding mutant to generate a test individual $U_{i,j} = (U_{i,1}, U_{i,2}, ..., U_{i,D})$. The crossover formula is:

$$U_{i,j} = \begin{cases} V_{i,j} & ,if\ \ r_j \leq CR\ \ or\ \ j = j_{rand} \\ X_{i,j} & ,otherwise \end{cases} \tag{17}$$

where $j = 1, 2...D$, $r_j \in (0, 1)$ are uniformly distributed random numbers generated for each j. $CR \in (0, 1)$ is the crossover probability parameter, $j_{rand} \in (1, 2...D)$ is a randomly selected dimension indicator. Figure 1 describes the flow chart of the proposed algorithm, and Algorithm 3 gives the pseudo-code for TLMPA.

Algorithm 3: pseudo of TLMPA

1. Initialize search agents (Prey) populations
2. Define parameters such as P, T_F, CF, $FADs$, F,CR
3. ***while*** $Iter < Max\_Iter$
4. Calculate the fitness, construct the Elite matrix and accomplish memory saving
5. Calculate the mean of each design variables
6. Select the best solution as top predator
7. ***if*** $Iter < Max\_Iter/3$
8. ***for*** $i = 1 : n$
9. Update prey based on Eq 13
10. ***end for***
11. ***else if*** $Max\_iter/3 < Iter < 2 * Max\_Iter/3$
12. ***for*** $i = 1 : n$
13. Update prey based on Eq 14
14. ***end for***
15. ***else if*** $Iter > 2 * Max\_Iter/3$
16. ***for*** $i = 1 : n$
17. Update prey based on Eq 15
18. ***end for***
19. ***end if***
20. Execute the process of mutation and crossover based on Eq 16 and Eq 17
21. Evaluate the fitness of prey and corresponding trial vector
22. Update the current best solution
23. Accomplish memory saving and Elite update
24. Applying FADs effect and update based on Eq 9
25. ***end while***

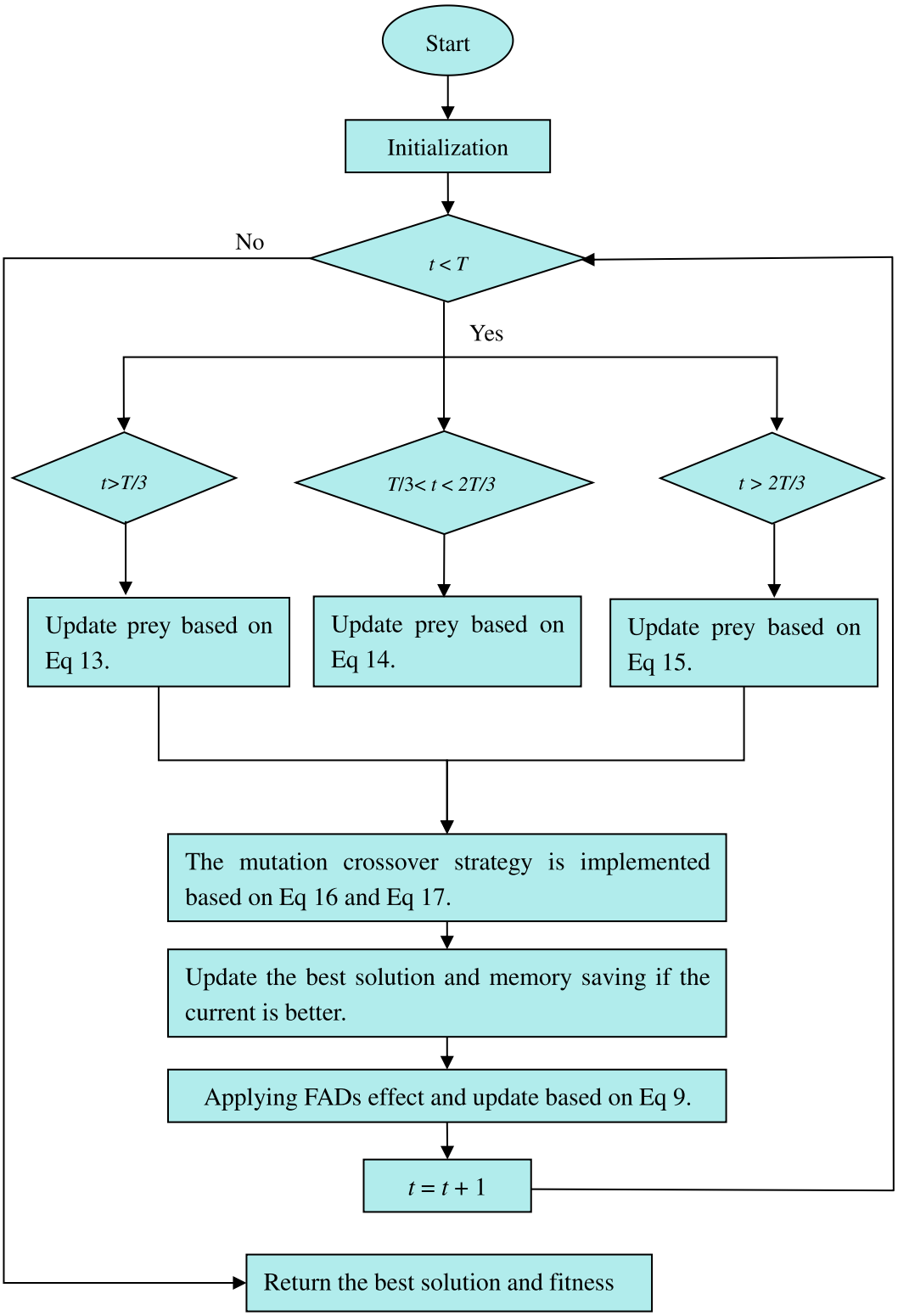

Figure 1. Flowchart of the proposed algorithm.

Complexity analysis

In order to better understand the algorithm process proposed in this article, in this section we analyze the time complexity and space complexity of the proposed algorithm. The analysis of time complexity is as follows:

In steps 1 and 2, initializing the population and setting the parameters requires $O(n * d)$, where n is the population number and d is the dimension of the problem. In Steps 4–6, it takes $O(n)$; steps 7–10, steps 11–14 and steps 15–19 correspond to the three stages of the algorithm, and they need $O(n * d)$; In steps 20, it costs $O(n * d)$; in steps 21–24, it costs $O(n)$; in addition, the outermost loop requires $O(t)$, where t is the number of iterations. Finally, the time complexity of TLMPA is $O(t * n * d)$.

Next, the space complexity of the proposed algorithm is discussed. Because the size of *Elite* matrix and *prey* matrix required in the algorithm are both $n * d$, and this is also the maximum space required by the algorithm. Finally, the space complexity of TLMPA is $O(n * d)$.

4. Experimental results and analysis

In order to verify the effectiveness of the proposed algorithm, this paper uses the IEEE CEC-2017 benchmark test set for testing. The CEC2017 contains 29 benchmark functions for evaluating optimization problems. These functions can be divided into four categories: unimodal function, multimodal function, mixed function and combined function, as shown in Table 1. The unimodal function has only one global optimal point, which is used to evaluate the exploitation ability of the meta-heuristic algorithm. The simple multimodal function has multiple local optima, which is used to evaluate the balance between exploitation and exploration and avoid falling into local optima. In the hybrid function, the variable is randomly divided into several subcomponents, and then different basic functions are used for different subcomponents. The composition function better integrates the properties of the sub-functions and maintains the continuity of the global/local optimal solution.

4.1. Compared with the well-known natural heuristic algorithm

The algorithm was implemented in MATLAB R2017b, and experiments were conducted on a PC with 2.4 GHz, Inter(R) Core(TM) i5 CPU, 64-bit system type, and windows7 operating system. In order to improve reliability and produce statistically significant results, in this verification test, the population size of all algorithms is set to 20. It should be noted that each method is run for 30 times, the maximum number of function calculations is 50000. Each function was run 30 times, and the average and standard deviation of the proposed algorithm and other algorithms were recorded. The initial parameter setting of each algorithm is shown in Table 2.

Table 1. IEEE CEC-2017 benchmark problems.

ID	Name of the function	Class	Range	Optimum
F1	Shifted and rotated bent cigar function	Unimodal	[-100,100]	100
F3	Shifted and rotated Zakharov function	Unimodal	[-100,100]	300
F4	Shifted and rotated Rosenbrock's function	Multimodal	[-100,100]	400
F5	Shifted and rotated Rastrigin's function	Multimodal	[-100,100]	500
F6	Shifted and rotated expanded Scaffer's F6 function	Multimodal	[-100,100]	600
F7	Shifted and rotated Lunacek bi-Rastrigin function	Multimodal	[-100,100]	700
F8	Shifted and rotated non-continuous Rastrigin's function	Multimodal	[-100,100]	800
F9	Shifted and rotated Lévy function	Multimodal	[-100,100]	900
F10	Shifted and rotated Schwefel's function	Multimodal	[-100,100]	1000
F11	Hybrid function 1 (N = 3)	Hybrid	[-100,100]	1100
F12	Hybrid function 2 (N = 3)	Hybrid	[-100,100]	1200
F13	Hybrid function 3 (N = 3)	Hybrid	[-100,100]	1300
F14	Hybrid function 4 (N = 4)	Hybrid	[-100,100]	1400
F15	Hybrid function 5 (N = 4)	Hybrid	[-100,100]	1500
F16	Hybrid function 6 (N = 4)	Hybrid	[-100,100]	1600
F17	Hybrid function 6 (N = 5)	Hybrid	[-100,100]	1700
F18	Hybrid function 6 (N = 5)	Hybrid	[-100,100]	1800
F19	Hybrid function 6 (N = 5)	Hybrid	[-100,100]	1900
F20	Hybrid function 6 (N = 6)	Hybrid	[-100,100]	2000
F21	Composition function 1 (N = 3)	Composition	[-100,100]	2100
F22	Composition function 2 (N = 3)	Composition	[-100,100]	2200
F23	Composition function 3 (N = 4)	Composition	[-100,100]	2300
F24	Composition function 4 (N = 4)	Composition	[-100,100]	2400
F25	Composition function 5 (N = 5)	Composition	[-100,100]	2500
F26	Composition function 6 (N = 5)	Composition	[-100,100]	2600
F27	Composition function 7 (N = 6)	Composition	[-100,100]	2700
F28	Composition function 8 (N = 6)	Composition	[-100,100]	2800
F29	Composition function 9 (N = 3)	Composition	[-100,100]	2900
F30	Composition function 10 (N = 3)	Composition	[-100,100]	3000

Table 2. Initial parameters for the meta-heuristic algorithms.

Algorithm	Parameter	Value
PSO	Topology	Fully connected
	Cognitive and social constant (C1, C2)	2, 2
	Inertia weight	linear reduction from 0.9 to 0.1
	Velocity limit	10% of the dimension range
SSA	Leader position update probability	0.5
LSHADE-cnEpSin	H, NP_{min}, P_{best} rate, Arc rate, p_s, p_c	5, 4, 0.11, 1.4, 0.5, 0.4
GWO	a	Linear reduction from 2 to 0
DE	$beta_{min}$, $beta_{max}$, pCR	0.2, 0.8, 0.2
TLBO	Teaching factor T_F , SearchAgent_no	1 or 2, 20
MPA	$FADs$, P, SearchAgent_no	0.2, 0.5, 20
TLMPA	$FADs$, P, $beta_{min}$, $beta_{max}$, pCR	0.2, 0.5, 0.2, 0.8, 0.2
	Teaching factor T_F , SearchAgent_no	1 or 2, 20

Compare TLMPA with seven natural heuristic algorithms, including marine predator algorithm (MPA), particle swarm algorithm (PSO), differential evolution algorithm (DE), salp swarm algorithm (SSA), LSHADE-cnEpSin, grey wolf optimizer (GWO), Teaching-learning-based optimization algorithm (TLBO). The results show that the proposed TLMPA has better search efficiency in

searching for the optimal solution of the problem. For the rest of this section, this paper will show the experimental results according to the four different types of functions of CEC2017.

4.1.1. Test on 10 dimensions

This section introduces the results of testing on the 10-dimensional problem of the CEC2017 benchmark function. Tables 3–6 respectively show the test results of the unimodal function, multimodal function, hybrid function and composition function. Figures 2 and 3 show the convergence curves of unimodal function tests; Figures 4–7 show the convergence curves of multimodal function tests; Figures 8–13 and Figures 14–19 show the convergence curves of hybrid functions and composition functions, respectively.

(1) *Test results using unimodal functions (F1, F3)*

The experimental results in Table 3 show that the proposed algorithm performs well on the unimodal test function and can find the optimal solution on both functions. It should be noted that the numbers in bold indicate the relative best values of the compared algorithms. The unimodal function mainly evaluates the exploitation ability of the algorithm. It can be seen from the data in the table that the proposed algorithm has strong exploitation capabilities. As one of the winners of the CEC2017 competition, the LSHADE-cnEpSin algorithm has excellent performance on unimodal functions, and it can also find the optimal solution. In these two functions, the proposed algorithm is comparable to LSHADE-cnEpSin. Among other algorithms, GWO and SSA have the worst performance, and the experimental results are far from the optimal value. In F1, DE has reached the optimal value in 30 experiments, but the overall result is still much worse than the optimal value; in the F3 function, DE performs slightly better, closer to the optimal value of 300. The worst value of MPA in the F1 function is far greater than the optimal value; in the F3 function, the performance is slightly more stable, and the result is closer to the optimal value. In general, from the experimental results in the unimodal function, the search performance of TLMPA is excellent and it has strong exploitation capabilities.

Table 3. Results of unimodal benchmark functions (D=10).

Functions	Algorithms	Best	Mean	Worst	Std.
F1	TLMPA	**100.00**	**100.00**	**100.00**	**0.00**
	MPA	**100.00**	181.08	886.43	206.44
	GWO	3.10E+07	1.76E+08	6.55E+08	1.64E+08
	DE	**100.00**	1112.12	4663.78	1266.62
	SSA	4.00E+07	8.80E+07	1.53E+08	3.35E+07
	TLBO	103.30	2383.37	7692.03	2483.15
	PSO	181.85	1.74E+04	1.80E+05	3.47E+04
	LSHADE-cnEpSin	**100.00**	**100.00**	**100.00**	**0.00**
F3	TLMPA	**300.00**	**300.00**	**300.00**	**0.00**
	MPA	**300.00**	300.04	301.06	0.19
	GWO	403.71	2819.93	8565.43	2616.18
	DE	300.35	339.70	480.99	55.66
	SSA	412.29	629.16	1015.42	166.55
	TLBO	**300.00**	**300.00**	**300.00**	**0.00**
	PSO	**300.00**	**300.00**	**300.00**	**0.00**
	LSHADE-cnEpSin	**300.00**	**300.00**	**300.00**	**0.00**

Figures 2 and 3 are the convergence graphs of TLMPA, MPA, GWO, DE, SSA, TLBO, PSO, and LSHADE-cnEpSin on the unimodal function. It can be seen from these figures that the convergence speed of TLMPA is faster than other algorithms. Among other algorithms, the search performance of DE is better than that of several other peer algorithms, but compared with TLMPA, TLMPA performs significantly better. The performance of LSHADE-cnEpSin in Figures 2 and 3 is quite stable, with high convergence ability. The convergence speed and accuracy of TLMPA on the unimodal function are equivalent to it. Through the analysis of the convergence curves shown in Figures 2 and 3, it is further proved that TLMPA can effectively find the optimal solution of the unimodal test function, which fully reflects its superior search performance.

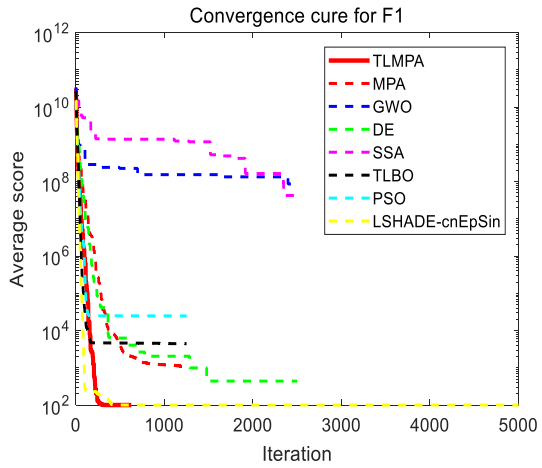

Figure 2. D=10, convergence cure for F1.

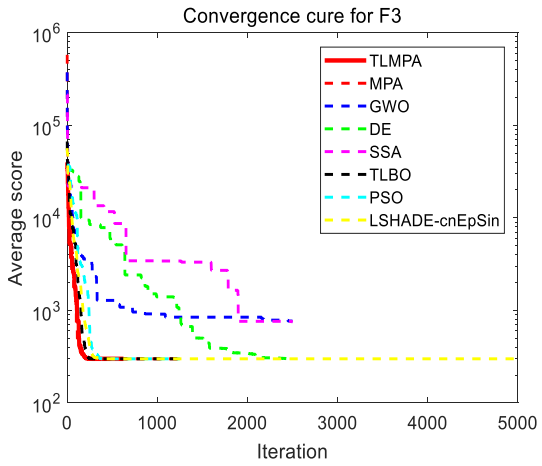

Figure 3. D=10, convergence cure for F3.

(2) *Test results using multimodal functions (F4–F10)*

Table 4 shows the experimental results of TLMPA and other comparison algorithms on multimodal functions. From the experimental results in the table, TLMPA does not perform the best on F4, F5, F7, F8 and F10, but it ranks in the top three in most functions. In F4, the performance of TLMPA is second only to the LSHADE-cnEpSin algorithm. It can find the theoretical optimal value, and its standard deviation is close to 0, and the mean and worst values are also very close to the global optimal value. In F5 and F7, none of the algorithms found the optimal value. Among them, the closest to the optimal value is the LSHADE-cnEpSin algorithm, followed by DE, and third is TLMPA. Although TLMPA does not perform the best, its result value is already very close to the optimal value, and its standard deviation is relatively small among all algorithms, which is better than the results of other peer algorithms, which also shows the performance of the TLMPA algorithm is stable. In F6 and F9, TLMPA performed very well, tied for first place with DE and LSHADE-cnEpSin, and they were able to find the global optimal value. On the whole, TLMPA is more effective in optimizing multimodal function problems, has a higher global search capability, and the balance between exploitation and exploration capabilities is better than most of the comparison algorithms.

Figures 4–7 are the convergence curves obtained by testing on the multimodal function. It can be seen from Figure 4 and Figure 7 that the convergence curves of TLMPA and other algorithms are similar, because all algorithms in F4 can find solutions that are closer to the optimal value. In Figure 7,

in addition to GWO and SSA, other algorithms can converge to a position close to the optimal solution. It can be seen from Figure 5 that TLMPA is second only to LSHADE-cnEpSin and DE by a small gap. Figure 6 shows that TLMPA has a faster convergence speed than other algorithms. In general, TLMPA shows higher convergence ability than most of the compared algorithms, reflecting its better search performance.

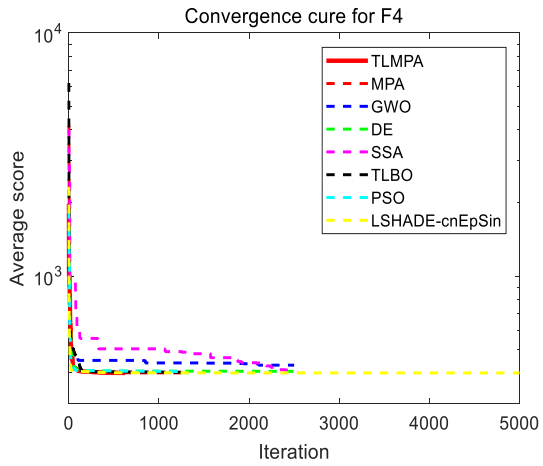

Figure 4. D=10, convergence cure for F4.

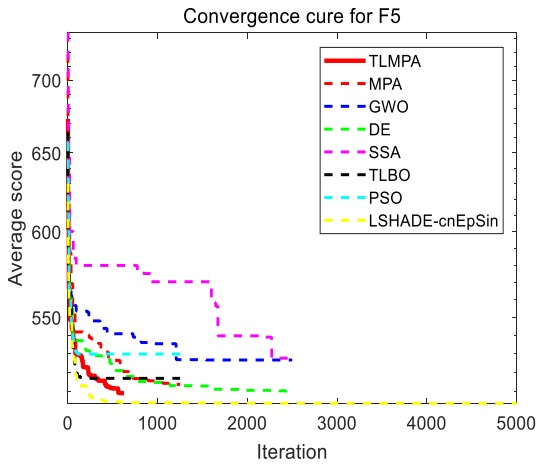

Figure 5. D=10, convergence cure for F5.

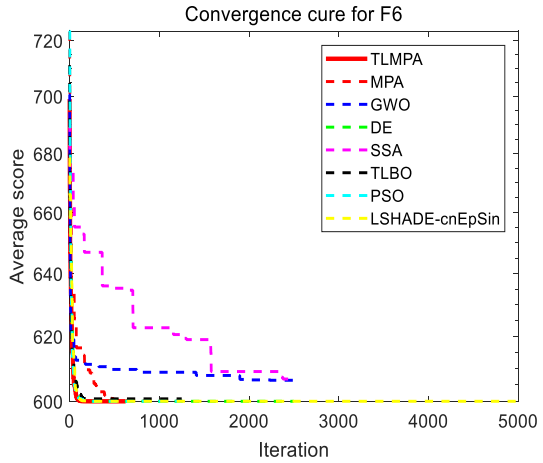

Figure 6. D=10, convergence cure for F6.

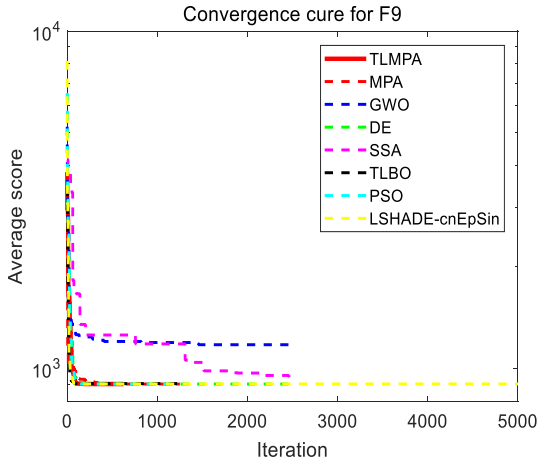

Figure 7. D=10, convergence cure for F9.

Table 4. Results of multimodal benchmark functions (D=10).

Functions	Algorithms	Best	Mean	Worst	Std.
F4	TLMPA	**400.00**	400.03	400.29	0.06
	MPA	**400.00**	400.73	404.80	1.33
	GWO	409.23	422.67	468.96	17.84
	DE	401.73	405.59	407.12	1.08
	SSA	407.09	414.79	471.19	12.06
	TLBO	400.93	403.15	406.47	1.27
	PSO	400.02	401.96	406.83	1.91
	LSHADE-cnEpSin	400.00	**400.00**	**400.00**	**0.00**
F5	TLMPA	506.40	510.58	515.83	2.58
	MPA	502.17	512.82	520.84	4.14
	GWO	517.35	529.62	541.60	5.75
	DE	503.05	506.69	**511.63**	**1.82**
	SSA	526.68	538.27	553.03	6.85
	TLBO	504.97	516.41	538.80	8.16
	PSO	503.98	515.55	533.83	7.78
	LSHADE-cnEpSin	**501.99**	**505.74**	513.93	2.65
F6	TLMPA	**600.00**	**600.00**	**600.00**	**0.00**
	MPA	600.05	600.29	601.00	0.27
	GWO	605.03	607.83	611.10	1.72
	DE	**600.00**	**600.00**	**600.00**	**0.00**
	SSA	604.70	610.36	624.39	4.83
	TLBO	**600.00**	601.16	608.16	1.69
	PSO	**600.00**	600.56	602.61	0.88
	LSHADE-cnEpSin	**600.00**	**600.00**	**600.00**	**0.00**
F7	TLMPA	716.60	723.63	730.66	3.08
	MPA	719.25	731.79	744.21	6.33
	GWO	733.90	747.24	767.57	8.37
	DE	714.27	717.85	**722.69**	**1.98**
	SSA	738.77	761.54	781.48	10.86
	TLBO	712.25	725.02	745.14	7.81
	PSO	707.47	724.59	740.95	7.42
	LSHADE-cnEpSin	**703.65**	**715.73**	722.96	3.56
F8	TLMPA	803.94	811.56	820.75	3.72
	MPA	804.02	811.50	817.80	3.14
	GWO	814.99	822.16	836.37	5.35
	DE	802.84	807.04	**810.95**	**2.05**
	SSA	823.92	840.11	851.18	7.07
	TLBO	803.98	811.76	825.39	5.21
	PSO	805.97	817.64	840.79	8.59
	LSHADE-cnEpSin	**801.00**	**806.16**	819.90	3.66
F9	TLMPA	**900.00**	**900.00**	**900.00**	**0.00**
	MPA	**900.00**	900.24	901.92	0.40
	GWO	903.58	930.09	1176.72	50.32
	DE	**900.00**	**900.00**	**900.00**	**0.00**
	SSA	913.76	945.59	1168.28	46.63
	TLBO	**900.00**	907.35	932.36	8.76
	PSO	**900.00**	902.93	918.41	4.46
	LSHADE-cnEpSin	**900.00**	**900.00**	**900.00**	**0.00**
F10	TLMPA	1394.27	1613.14	1784.97	113.19
	MPA	1238.73	1605.54	1886.28	157.58
	GWO	1229.25	1805.08	2380.15	281.31
	DE	1126.12	1295.65	**1566.73**	**104.27**
	SSA	1817.99	2229.12	2768.07	205.49
	TLBO	1003.54	1553.23	2253.07	297.66
	PSO	1242.01	1613.81	2004.14	203.03
	LSHADE-cnEpSin	**1007.02**	**1239.08**	1601.76	133.49

(3) *Test results using hybrid functions (F11–F20)*

Table 5 shows the test results of TLMPA and other algorithms in the hybrid functions. As can be seen from the data in the table, TLMPA did not get the best results in F11, F13, F16, F17 and F18, but it was always among the top three algorithms, and the results of TLMPA in these functions were very close to the theoretical optimal value. In F14, F15, F19 and F20, the performance of TLMPA is the best among all algorithms, and the average value is smaller than LSHADE-cnEpSin. In summary, the performance of TLMPA in the hybrid function is slightly inferior to the LSHADE-cnEpSin algorithm, but compared to other algorithms, TLMPA still has a strong search performance.

Figures 8–13 illustrate the convergence of the optimization process using the hybrid functions. In Figure 9, Figure 10 and Figure 13, TLMPA has the best convergence speed and accuracy. In Figure 8, the convergence of TLMPA is similar to that of MPA and LSAHDE-cnEpSin, and both have good search performance. Although the final result of TLMPA in Figure 11 is not the best, it is also very close to the optimal value. In general, TLMPA performs well in mixed functions, and compared with most of the comparison algorithms, it can effectively search for the optimal solution.

From the F21, F22, F26 and F28 in the table, it can be seen that TLMPA can find the optimal value in these functions, although some of its mean and minimum values are not optimal or the best among all comparison algorithms. However, in F22, F27, F28, and F30, the mean value of TLMPA is the closest to the optimal value among all algorithms. From this point of view, its search performance is better than other algorithms. Among F21, F24, F25 and F26, TLMPA is not the best performing algorithm, but the best value it finds is the smallest among all algorithms. In F23 and F29, TLMPA did not get very good results, but in terms of results, it is not much different from those algorithms that have better results.

(4) *Test results using composition functions (F21–F30)*

Table 6 shows the experimental result data obtained by testing in the composition functions. On the whole, the performance of TLMPA in the composition functions is one of the best in the comparison algorithm, and it can effectively search for the optimal solution or the approximate optimal solution. Figures 14–19 are the convergence curves obtained by testing in the composition functions. Figure 14, Figure 16, Figure 17, and Figure 19 show that the convergence speed and accuracy of TLMPA are better than most comparison algorithms. The convergence of each algorithm in Figure 15 is very similar, which is mainly because the result values found by each algorithm are very close. In Figure 18, the performance of TLMPA is not very good, but it is not so bad, and it is still competitive compared with the other algorithms. But in terms of overall performance, TLMPA is an excellent algorithm, compared with other algorithms, it has high search performance.

Table 5. Results of hybrid benchmark functions (D=10).

Functions	Algorithms	Best	Mean	Worst	Std.
F11	TLMPA	1100.58	1102.51	1104.56	1.08
	MPA	1100.51	1103.31	1106.86	1.47
	GWO	1118.42	1143.13	1191.94	15.53
	DE	1100.13	1102.04	1104.05	1.08
	SSA	1132.40	1168.07	1250.69	28.49
	TLBO	1102.99	1125.18	1192.53	20.87
	PSO	1101.01	1112.36	1143.79	8.82
	LSHADE-cnEpSin	**1100.00**	**1100.76**	**1102.98**	**0.97**
F12	TLMPA	1206.20	1310.64	1589.51	95.32
	MPA	**1200.33**	**1301.26**	**1476.00**	**71.26**
	GWO	2.39E+05	2.22E+06	8.77E+06	1.96E+06
	DE	1.56E+04	4.30E+04	1.04E+05	2.33E+04
	SSA	1.19E+06	7.08E+06	1.44E+07	4.11E+06
	TLBO	3068.00	1.32E+04	5.45E+04	1.26E+04
	PSO	1919.17	3.29E+04	1.14E+05	3.35E+04
	LSHADE-cnEpSin	1318.65	1332.72	1492.57	77.40
F13	TLMPA	1306.49	1309.72	1315.79	**2.36**
	MPA	1302.78	1308.89	**1314.36**	2.54
	GWO	3278.40	1.46E+04	4.18E+04	8437.26
	DE	1307.14	2845.93	1.11E+04	2551.30
	SSA	3331.90	2.63E+04	1.01E+05	2.09E+04
	TLBO	1712.00	5375.69	1.48E+04	3688.28
	PSO	1312.88	1.87E+04	6.82E+04	1.90E+04
	LSHADE-cnEpSin	**1301.97**	**1303.91**	1315.57	3.12
F14	TLMPA	**1400.19**	**1402.71**	**1408.59**	**1.88**
	MPA	1401.19	1404.44	1413.91	3.30
	GWO	1457.84	2309.90	5178.96	1419.00
	DE	1400.14	1535.10	2188.66	232.73
	SSA	1461.45	1525.63	1674.09	43.44
	TLBO	1428.11	1446.34	1516.29	18.97
	PSO	1430.59	2318.41	15986.26	2710.74
	LSHADE-cnEpSin	1409.95	1421.96	1425.98	6.95
F15	TLMPA	1500.27	**1500.76**	**1501.58**	**0.36**
	MPA	**1500.18**	1501.07	1503.73	0.70
	GWO	1569.15	3852.43	8049.74	1960.74
	DE	1500.19	2174.62	14418.27	2355.72
	SSA	1664.41	2017.68	3158.44	346.14
	TLBO	1525.19	1587.36	1938.48	75.97
	PSO	1545.76	8079.85	50681.18	11463.48
	LSHADE-cnEpSin	1503.02	1513.38	1517.93	5.09
F16	TLMPA	1600.77	1604.45	1623.75	4.61
	MPA	1601.16	1603.13	1607.06	1.32
	GWO	1625.10	1764.13	2152.77	133.20
	DE	1600.23	1607.66	1719.05	22.54
	SSA	1612.25	1688.94	1797.46	62.46
	TLBO	1601.29	1672.82	1949.35	85.75
	PSO	1600.67	1691.89	1855.82	84.86
	LSHADE-cnEpSin	**1600.00**	**1601.31**	**1605.58**	**1.23**
F17	TLMPA	1700.49	1706.29	1720.46	4.82
	MPA	1712.14	1731.88	1742.95	8.21
	GWO	1740.23	1763.80	1876.61	25.53
	DE	**1700.00**	1703.31	1728.68	7.01
	SSA	1739.25	1775.41	1854.68	23.72
	TLBO	1711.90	1747.67	1766.82	12.93
	PSO	1706.96	1769.63	1941.29	57.57
	LSHADE-cnEpSin	**1700.00**	**1702.95**	**1707.80**	**1.82**
F18	TLMPA	1800.49	1801.47	1809.13	1.65
	MPA	**1800.12**	**1801.25**	**1803.55**	**0.92**
	GWO	1.41E+04	4.43E+04	1.14E+05	1.77E+04
	DE	1800.62	2616.51	9075.50	1574.95
	SSA	6199.50	2.91E+04	6.28E+04	1.75E+04
	TLBO	2367.16	6848.11	2.30E+04	4908.72
	PSO	2334.33	2.61E+04	1.91E+05	3.53E+04
	LSHADE-cnEpSin	1810.03	1818.03	1837.01	7.20
F19	TLMPA	1900.03	**1900.45**	**1901.85**	**0.47**
	MPA	1900.29	1901.24	1902.58	0.68
	GWO	1953.81	4994.66	1.43E+04	4769.57
	DE	1900.05	2004.07	2534.12	176.12
	SSA	1943.04	2290.12	4196.77	480.71
	TLBO	1908.68	1955.57	2079.30	43.66
	PSO	1902.38	7460.92	3.11E+04	8299.78
	LSHADE-cnEpSin	**1900.00**	1905.21	1906.15	1.03
F20	TLMPA	**2000.00**	**2000.72**	2019.79	**3.61**
	MPA	2002.48	2022.22	2038.72	9.22
	GWO	2042.35	2107.12	2223.19	53.86
	DE	**2000.00**	2004.56	2118.44	21.62
	SSA	2029.73	2083.89	2203.50	39.28
	TLBO	2000.31	2039.04	2196.08	37.08
	PSO	2001.99	2041.84	2165.09	37.76
	LSHADE-cnEpSin	**2000.00**	2010.72	**2018.75**	5.97

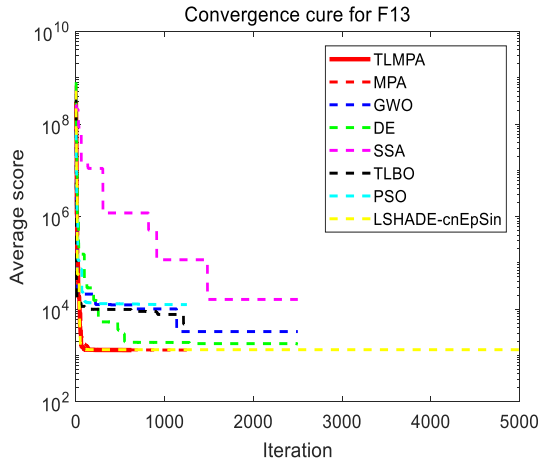

Figure 8. D=10,convergence cure for F13.

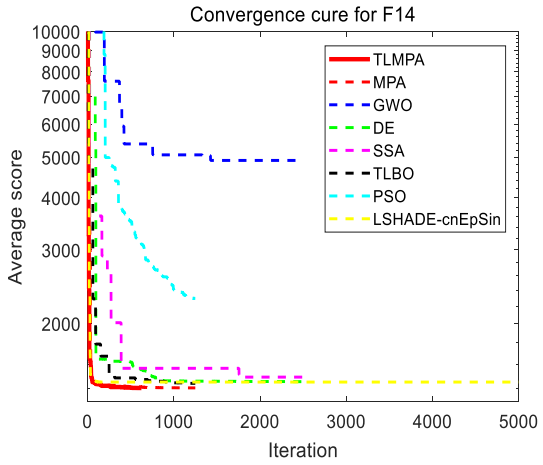

Figure 9. D=10,convergence cure for F14.

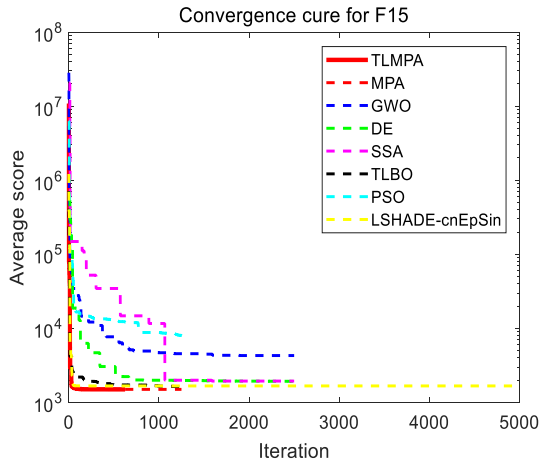

Figure 10. D=10, convergence cure for F15.

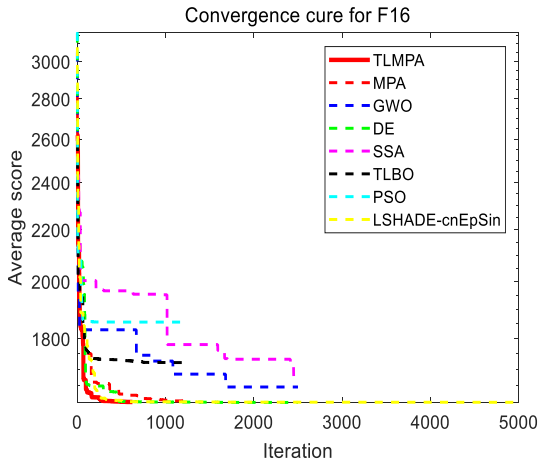

Figure 11. D=10, convergence cure for F16.

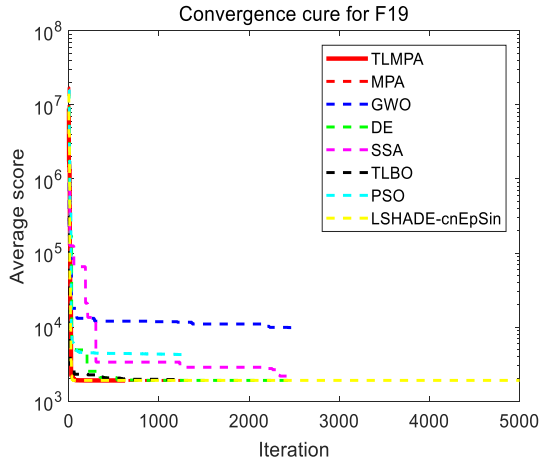

Figure 12. D=10, convergence cure for F19.

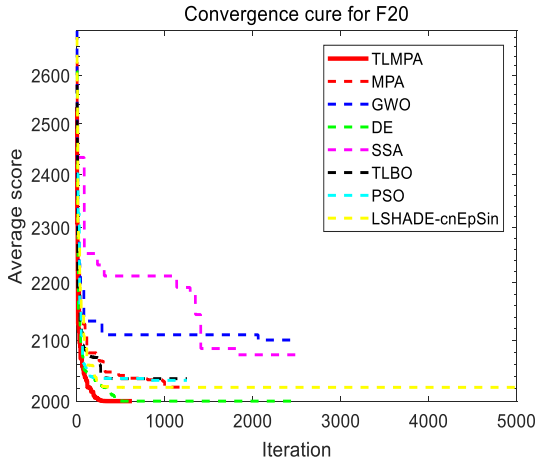

Figure 13. D=10, convergence cure for F20.

Table 6. Results of composition benchmark functions (D=10).

Functions	Algorithms	Best	Mean	Worst	Std.
F21	TLMPA	**2200.00**	2214.99	2316.96	38.45
	MPA	**2200.00**	**2207.40**	**2310.85**	27.52
	GWO	2202.36	2306.09	2340.28	46.46
	DE	2229.61	2292.04	2316.01	30.22
	SSA	2202.31	2261.89	2350.04	67.36
	TLBO	**2200.00**	2270.03	2337.51	54.52
	PSO	2303.01	2321.40	2337.41	**7.90**
	LSHADE-cnEpSin	**2200.00**	2210.06	2317.69	44.38
F22	TLMPA	**2200.00**	**2288.34**	**2301.26**	29.04
	MPA	2212.85	2295.43	2302.78	22.41
	GWO	2313.66	2333.72	2445.20	30.84
	DE	2232.42	2297.15	2301.70	13.40
	SSA	2314.51	2324.08	2336.42	**5.59**
	TLBO	2242.06	2300.48	2310.99	13.71
	PSO	2300.86	2402.10	3326.92	272.01
	LSHADE-cnEpSin	2211.56	2298.03	2303.44	16.34
F23	TLMPA	2604.60	2611.55	2620.87	4.04
	MPA	2603.80	2610.30	2620.94	3.93
	GWO	2607.53	2635.34	2656.25	6.73
	DE	2609.03	2610.02	2613.79	**1.79**
	SSA	2608.92	2636.93	2648.84	6.77
	TLBO	2608.93	2619.02	2633.83	7.52
	PSO	2607.45	2621.13	2638.47	8.42
	LSHADE-cnEpSin	**2603.74**	**2607.79**	**2613.14**	3.18
F24	TLMPA	**2439.29**	2587.59	2747.14	112.70
	MPA	2500.00	**2521.44**	**2730.90**	53.13
	GWO	2514.84	2749.66	2788.14	62.38
	DE	2575.54	2737.81	2751.38	30.85
	SSA	2527.98	2759.13	2790.39	44.64
	TLBO	2500.00	2731.45	2760.12	58.00
	PSO	2738.87	2757.52	2792.37	**12.56**
	LSHADE-cnEpSin	2500.00	2529.59	2747.53	43.67
F25	TLMPA	**2897.74**	2905.56	**2943.82**	17.29
	MPA	2897.74	**2901.18**	2943.94	**11.58**
	GWO	2904.29	2942.61	3037.82	23.36
	DE	2897.95	2916.84	2949.87	21.15
	SSA	2904.68	2943.96	3029.06	23.85
	TLBO	2897.78	2923.14	2952.27	24.56
	PSO	2897.78	2932.68	2981.20	25.34
	LSHADE-cnEpSin	2897.87	2922.22	2949.26	23.36
F26	TLMPA	**2600.00**	2879.99	**2900.00**	76.11
	MPA	**2600.00**	**2820.21**	2900.40	117.37
	GWO	2919.27	3191.92	4105.35	409.94
	DE	2800.18	2923.62	2988.25	40.87
	SSA	2757.54	2999.14	3985.36	268.84
	TLBO	2800.00	2976.87	3209.81	106.93
	PSO	2800.00	3044.33	3676.63	163.54
	LSHADE-cnEpSin	2900.00	2900.00	**2900.00**	**0.00**
F27	TLMPA	3088.47	**3089.36**	**3091.56**	**0.58**
	MPA	3089.01	3089.53	3092.86	0.75
	GWO	3094.42	3114.02	3207.80	31.95
	DE	3088.98	3090.72	3093.44	1.22
	SSA	3094.02	3098.30	3102.59	2.11
	TLBO	3089.64	3100.06	3171.39	14.24
	PSO	**3071.57**	3153.85	3200.00	61.70
	LSHADE-cnEpSin	3086.72	3090.31	3093.73	0.88
F28	TLMPA	**2800.00**	**3108.92**	3383.73	92.59
	MPA	3100.00	3109.49	3383.73	51.80
	GWO	3175.85	3408.12	3592.96	91.97
	DE	3165.14	3339.90	3411.89	91.99
	SSA	3130.08	3244.71	3732.09	155.95
	TLBO	3100.00	**3300.51**	3731.81	155.23
	PSO	3100.00	3242.22	3300.00	**51.60**
	LSHADE-cnEpSin	3100.00	3259.12	3311.81	80.94
F29	TLMPA	3150.76	3165.07	3199.24	11.13
	MPA	**3135.05**	3157.50	3179.58	9.78
	GWO	3164.00	3213.82	3335.00	38.70
	DE	3145.08	3160.64	3233.19	16.66
	SSA	3157.92	3201.97	3308.21	31.39
	TLBO	3141.74	3190.93	3259.24	29.49
	PSO	3143.04	3231.20	3434.08	65.05
	LSHADE-cnEpSin	3135.59	**3142.18**	**3167.48**	**8.46**
F30	TLMPA	3410.56	**4133.44**	2.09E+04	3178.85
	MPA	3395.34	3.11E+04	8.21E+05	1.49E+05
	GWO	1.22E+04	7.46E+05	2.97E+06	9.35E+05
	DE	5686.84	3.44E+04	4.67E+05	8.37E+04
	SSA	1.81E+04	2.21E+05	1.44E+06	3.48E+05
	TLBO	4250.34	2.21E+05	1.25E+06	4.42E+05
	PSO	**3221.32**	4140.47	**1.05E+04**	**1754.09**
	LSHADE-cnEpSin	3419.21	4600.24	3.18E+04	5135.38

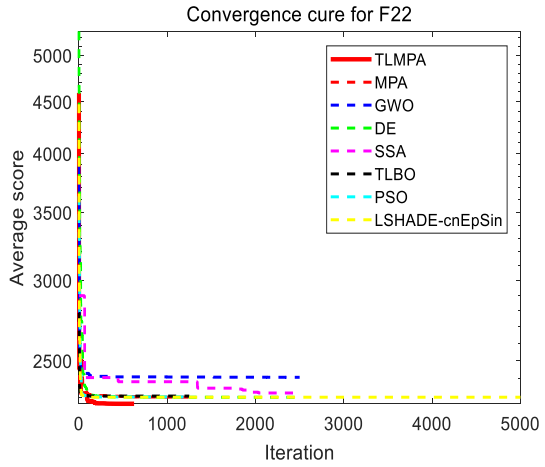

Figure 14. D=10, convergence cure for F22.

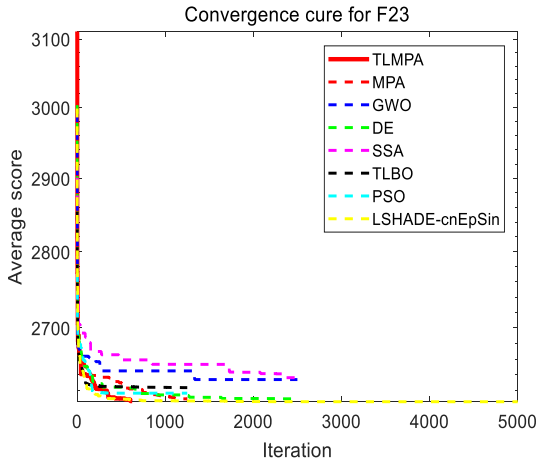

Figure 15. D=10, convergence cure for F23.

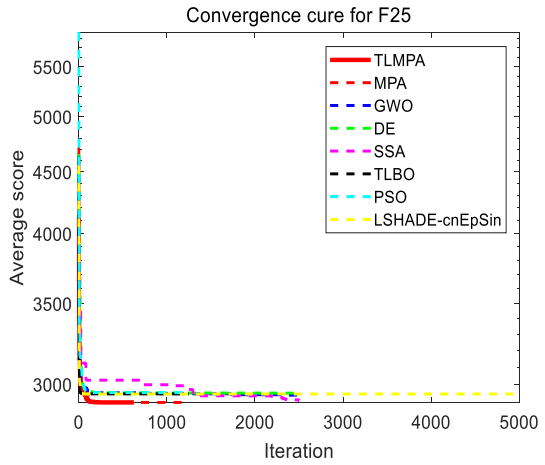

Figure 16. D=10, convergence cure for F25.

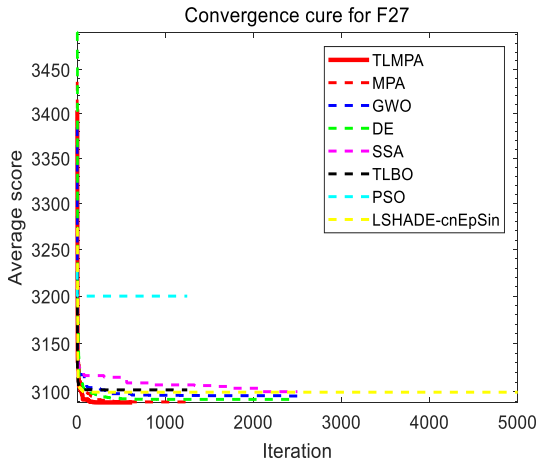

Figure 17. D=10, convergence cure for F27.

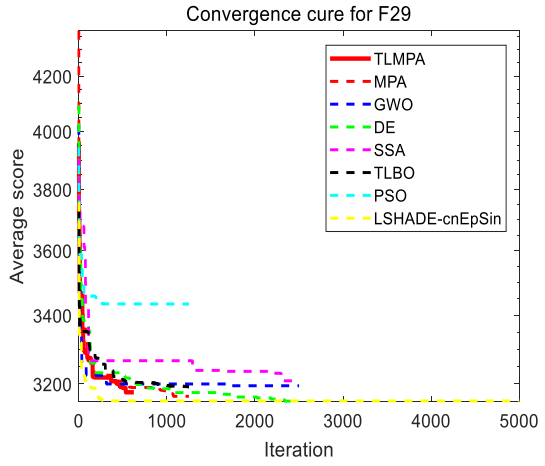

Figure 18. D=10, convergence cure for F29.

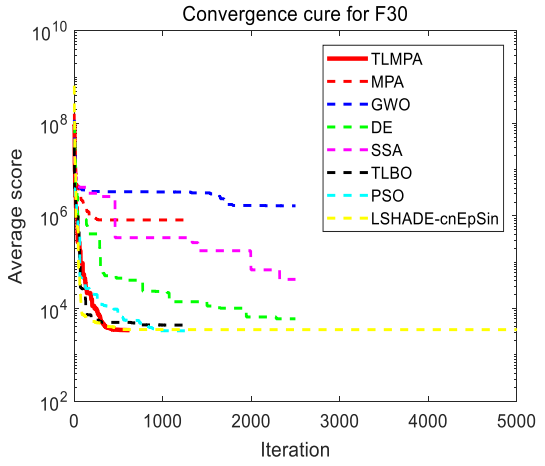

Figure 19. D=10, convergence cure for F30.

4.1.2. Test on 30 dimension

This section introduces the results of testing on 30-dimensional problems. Table 7 shows the test results using unimodal function, multimodal function, hybrid function and composition function. Figure 20 is the convergence curve obtained by the unimodal function test in 30 dimensions; Figures 21–23 are the convergence curves of the multimodal function test; Figures 24–26 and Figures 27–29 respectively show the convergence curve of hybrid function and composition function test.

(1) *Test on unimodal functions (D = 30)*

It can be clearly seen from Table 7 that both TLMPA and LSHADE-cnEpSin have found the optimal solution for the unimodal function test problem, and the performance is far more outstanding than other algorithms. It can be proved that the proposed algorithm still has a powerful search performance in 30 dimensions. From the comparison of the unimodal function test results in the two dimensions of 10-dimensional and 30-dimensional, TLMPA can achieve good results in both dimensions, which just shows that TLMPA has good development capabilities.

Table 7. Results of unimodal benchmark functions (D=30).

Functions	Algorithms	Best	Mean	Worst	Std.
F1	TLMPA	**100.00**	**100.00**	**100.00**	**0.00**
	MPA	2.64E+08	9.26E+08	1.87E+09	4.63E+08
	GWO	2.19E+09	4.83E+09	8.20E+09	1.49E+09
	DE	108.43	4234.45	17246.55	5121.82
	SSA	9.63E+08	1.71E+09	2.67E+09	4.12E+08
	TLBO	154.51	4531.55	1.92E+04	5679.88
	PSO	108.94	1.85E+04	2.07E+05	4.52E+04
	LSHADE-cnEpSin	**100.00**	**100.00**	**100.00**	**0.00**
F3	TLMPA	300.01	300.56	306.92	1.52
	MPA	7901.63	1.83E+04	3.26E+04	6445.35
	GWO	1.81E+04	3.91E+04	6.40E+04	1.00E+04
	DE	2.05E+04	3.06E+04	5.14E+04	7529.50
	SSA	7563.69	1.11E+04	1.85E+04	2803.64
	TLBO	300.01	316.90	575.48	61.19
	PSO	963.62	30167.90	1.05E+05	3.04E+04
	LSHADE-cnEpSin	**300.00**	**300.00**	**300.00**	**0.00**

(2) *Test on multimodal functions (D = 30)*

The test results of TLMPA on the multimodal function are shown in the test results from F4 to F10 in Table 8. In the multimodal function test, the performance of the proposed algorithm is second only to the LSHADE-cnEpSin algorithm. Among them, in F6 and F9, TLMPA can find the theoretical optimal value; in F4, F5, F7 and F10, the test result of TLMPA is very close to the result value of the first place; in F8, the proposed algorithm is the best among all algorithms. From the test results of multimodal functions in 10 and 30 dimensions, the performance of TLMPA in 30 dimensions is better than that in 10 dimensions. In 30 dimensions, most function test results can rank second. It can also be seen that TLMPA performs better in high-dimensional multimodal function tests than low-dimensional tests. In summary, TLMPA is better than most comparison algorithms in terms of multimodal function, which also shows that TLMPA is much better than other algorithms in terms of balancing exploration and mining capabilities.

Table 8. Results of multimodal benchmark functions (D=30).

Functions	Algorithms	Best	Mean	Worst	Std.
F4	TLMPA	403.99	467.79	**512.42**	32.13
	MPA	536.00	588.64	641.68	31.22
	GWO	592.89	693.65	888.07	90.76
	DE	486.64	493.11	518.54	9.94
	SSA	595.21	662.23	801.47	61.96
	TLBO	**400.15**	476.77	526.80	41.46
	PSO	407.47	448.13	540.26	34.49
	LSHADE-cnEpSin	404.09	**411.04**	516.22	**3.38**
F5	TLMPA	**517.91**	550.13	609.14	18.77
	MPA	602.94	646.58	677.13	18.38
	GWO	668.38	702.73	752.49	20.64
	DE	600.29	613.89	633.11	**8.32**
	SSA	708.13	746.96	776.46	17.69
	TLBO	576.61	622.43	652.23	22.71
	PSO	568.65	626.91	676.11	37.59
	LSHADE-cnEpSin	520.80	**543.48**	**602.48**	25.67
F6	TLMPA	**600.00**	**600.00**	**600.00**	**0.00**
	MPA	604.53	608.42	614.50	2.16
	GWO	618.92	626.42	635.48	4.88
	DE	**600.00**	**600.00**	**600.00**	**0.00**
	SSA	617.08	635.04	653.51	9.87
	TLBO	612.78	622.29	634.94	6.42
	PSO	605.30	614.63	629.22	7.31
	LSHADE-cnEpSin	**600.00**	**600.00**	**600.00**	**0.00**
F7	TLMPA	770.46	807.70	874.47	26.60
	MPA	849.40	946.75	994.59	38.15
	GWO	934.99	982.12	1037.39	28.11
	DE	833.67	853.13	876.12	11.09
	SSA	991.73	1050.10	1163.71	46.67
	TLBO	879.59	931.31	1048.48	45.24
	PSO	806.95	889.56	1024.77	51.48
	LSHADE-cnEpSin	**768.38**	**801.37**	**812.54**	**9.19**
F8	TLMPA	**818.90**	**847.90**	**870.64**	14.48
	MPA	892.60	936.07	966.24	20.32
	GWO	952.52	982.20	1017.10	16.32
	DE	905.86	922.18	937.60	**9.11**
	SSA	974.46	1050.46	1115.75	28.74
	TLBO	861.69	905.26	939.29	22.91
	PSO	880.59	946.36	1030.83	39.15
	LSHADE-cnEpSin	839.80	861.28	892.53	13.05
F9	TLMPA	**900.00**	901.34	906.81	2.06
	MPA	1261.53	1612.25	2541.47	303.93
	GWO	1823.67	2847.26	4807.11	751.15
	DE	**900.00**	**900.00**	**900.00**	**0.00**
	SSA	2102.28	5326.49	12904.42	3195.42
	TLBO	1498.50	2494.39	3886.44	759.77
	PSO	1496.15	4057.33	9025.01	2231.98
	LSHADE-cnEpSin	**900.00**	**900.00**	**900.00**	**900.00**
F10	TLMPA	5134.61	6025.47	6952.04	443.24
	MPA	4332.75	5463.91	6458.99	552.62
	GWO	5880.99	6844.05	7971.80	499.43
	DE	5406.03	6045.46	6505.80	306.52
	SSA	7482.77	8205.98	8700.37	393.08
	TLBO	3685.79	7177.34	8436.34	1284.94
	PSO	3918.78	4836.21	7734.18	852.09
	LSHADE-cnEpSin	**2807.29**	**3267.93**	**3599.24**	**250.22**

(3) *Test on multimodal functions (D = 30)*

In Table 9, F11 to F20 are the test results of hybrid functions. The performance of TLMPA in hybrid function fully reflects its superior search performance. According to the test results of F13, F14, F15 and F19, TLMPA has achieved excellent results in the test of these functions, especially in all algorithms; among F11, F12, F16, F17, F18 and F20, TLMPA is not the first but always the second. It is worth mentioning that the best values obtained by TLMPA in F17, F18 and F20 are the smallest among all algorithms. Similarly, judging from the comparison of the results of 10-dimensional and 30-dimensional, the performance of TLMPA in 30-dimensional is better than 10-dimensional, because the number of second-ranked ones is higher. Finally, TLMPA shows its strong search ability in the hybrid function.

(4) *Test on composition functions (D = 30)*

The test results of the composition function correspond to the test results from F21 to F30 in the Table 10. From the result point of view, the test results of TLMPA in F21, F23, F24, F29 and F30 are the best among all the algorithms and the closest to the optimal value. Although in F22, F25, F26, F27 and F28, TLMPA performance is not the best algorithm, but it is also a sub-optimal algorithm, which also reflects that its search performance in complex functions is unmatched by other algorithms.The 30-dimensional composition function test result is also much better without the 10-dimensional situation. TLMPA fully reflects its powerful search performance in the case of high-dimensionality.

Figure 20 shows that the convergence speeds and convergence accuracy of TLMPA are better than most algorithms; Figures 21–23 show the convergence curves of the multimodal function test. Except for Figure 23, the convergence of TLMPA is second to that of the LSHADE-cnEpSin algorithm; Figures 24–25 show that the convergence of TLMPA is better than most algorithms, and can be ranked second. In Figure 26 to Figure 29, the convergence of TLMPA is the best among all algorithms, and both search speed and accuracy are better than other algorithms.

Table 9. Results of hybrid benchmark functions (D=30).

Functions	Algorithms	Best	Mean	Worst	Std.
F11	TLMPA	**1108.76**	1129.83	1180.87	22.24
	MPA	1179.64	1280.45	1339.68	39.75
	GWO	1464.39	2026.35	3574.74	786.37
	DE	1114.26	1160.71	1188.52	27.13
	SSA	1454.52	1602.71	1712.10	66.50
	TLBO	1174.93	1253.13	1332.92	46.03
	PSO	1126.04	1226.37	1317.43	47.37
	LSHADE-cnEpSin	1100.49	**1115.07**	**1167.50**	**16.70**
F12	TLMPA	**5896.59**	3.07E+04	1.01E+05	2.14E+04
	MPA	1.68E+06	6.70E+06	1.69E+07	3.95E+06
	GWO	8.29E+07	2.66E+08	5.53E+08	1.34E+08
	DE	3.46E+05	2.14E+06	7.38E+06	2.16E+06
	SSA	8.36E+07	2.11E+08	3.10E+08	7.11E+07
	TLBO	1.78E+04	8.22E+04	7.93E+05	1.74E+05
	PSO	3.48E+04	5.04E+05	3.05E+06	6.59E+05
	LSHADE-cnEpSin	8109.98	**1.86E+04**	**4.43E+04**	**9330.52**
F13	TLMPA	**1352.68**	**1504.95**	**2129.78**	**181.13**
	MPA	1996.48	9282.64	5.02E+04	1.16E+04
	GWO	3.30E+07	1.17E+08	3.84E+08	9.19E+07
	DE	1.58E+04	4.95E+04	1.54E+05	3.30E+04
	SSA	3.35E+07	5.85E+07	1.22E+08	2.27E+07
	TLBO	3789.75	1.85E+04	6.01E+04	1.52E+04
	PSO	1930.95	6.57E+04	2.34E+05	7.33E+04
	LSHADE-cnEpSin	2507.85	4384.25	6.34E+03	1204.77
F14	TLMPA	**1405.34**	**1428.44**	**1445.72**	**11.06**
	MPA	1444.35	1459.16	1486.11	11.20
	GWO	2.83E+04	8.38E+04	4.90E+05	1.02E+05
	DE	2.06E+04	1.08E+05	4.10E+05	9.30E+04
	SSA	1.46E+04	3.59E+04	8.76E+04	1.86E+04
	TLBO	2241.97	7425.54	1.70E+04	4700.62
	PSO	2653.42	4.64E+04	1.58E+05	4.22E+04
	LSHADE-cnEpSin	1458.51	1616.62	1777.20	84.13
F15	TLMPA	**1507.48**	**1528.57**	**1605.80**	**24.60**
	MPA	1613.72	1705.10	1929.66	85.17
	GWO	4.02E+05	1.48E+06	4.92E+06	1.23E+06
	DE	3474.42	1.31E+04	3.15E+04	8338.03
	SSA	1.95E+06	6.62E+06	1.27E+07	3.03E+06
	TLBO	1872.63	5302.28	1.13E+04	3238.49
	PSO	1850.12	2.62E+04	9.86E+04	2.58E+04
	LSHADE-cnEpSin	1649.09	1806.91	2008.03	103.08
F16	TLMPA	1803.01	2171.83	2425.07	205.44
	MPA	1898.36	2284.67	2639.25	189.55
	GWO	2115.97	2725.28	3233.92	305.06
	DE	1904.83	2142.38	2500.67	160.14
	SSA	2815.55	3211.74	3543.79	248.60
	TLBO	**1796.40**	2342.55	2663.98	249.58
	PSO	1872.43	2645.30	3538.39	372.68
	LSHADE-cnEpSin	1871.02	**2034.69**	**2377.25**	**91.79**
F17	TLMPA	**1742.30**	1846.41	2054.81	85.76
	MPA	1761.61	1831.86	**1995.17**	72.27
	GWO	1845.78	2106.08	2370.32	146.82
	DE	1791.89	1874.63	2042.24	67.73
	SSA	1929.51	2222.28	2518.75	170.94
	TLBO	1823.71	2084.07	2374.49	170.26
	PSO	2058.56	2337.84	2616.10	167.34
	LSHADE-cnEpSin	1754.84	**1818.86**	2002.95	**65.97**
F18	TLMPA	**1885.75**	4713.72	2.80E+04	5966.83
	MPA	2332.33	1.31E+04	4.29E+04	1.00E+04
	GWO	6.35E+05	1.77E+06	3.96E+06	9.65E+05
	DE	1.47E+05	4.38E+05	1.18E+06	2.61E+05
	SSA	2.45E+05	8.85E+05	1.92E+06	4.98E+05
	TLBO	4.61E+04	2.17E+05	9.34E+05	2.01E+05
	PSO	4.54E+04	3.89E+05	1.71E+06	4.42E+05
	LSHADE-cnEpSin	2126.33	**4348.39**	**1.78E+04**	**4168.74**
F19	TLMPA	**1908.00**	**1913.86**	**1919.50**	**3.19**
	MPA	1917.47	1937.40	1970.86	13.30
	GWO	1.19E+06	3.38E+06	8.90E+06	2.01E+06
	DE	2096.58	1.15E+04	3.48E+04	9541.34
	SSA	2.67E+06	1.84E+07	5.30E+07	1.26E+07
	TLBO	2227.19	9094.10	2.64E+04	6916.59
	PSO	2178.11	1.52E+04	5.70E+04	1.62E+04
	LSHADE-cnEpSin	2002.99	2200.74	2670.13	178.36
F20	TLMPA	**2034.87**	2171.74	2369.48	96.33
	MPA	2043.36	2191.46	2397.55	74.85
	GWO	2298.02	2445.06	2658.77	108.72
	DE	2042.79	2173.95	2398.05	93.53
	SSA	2391.56	2620.64	3050.56	142.70
	TLBO	2175.93	2346.24	2538.82	105.62
	PSO	2231.21	2553.75	2867.15	135.26
	LSHADE-cnEpSin	2061.00	**2125.15**	**2365.07**	**73.06**

Table 10. Results of composition benchmark functions (D=30).

Functions	Algorithms	Best	Mean	Worst	Std.
F21	TLMPA	**2325.65**	**2342.28**	**2359.06**	**9.89**
	MPA	2360.59	2410.77	2457.90	45.29
	GWO	2386.47	2480.59	2537.60	22.45
	DE	2453.06	2421.01	2436.71	7.15
	SSA	2365.04	2527.05	2565.55	20.05
	TLBO	2395.11	2405.85	2459.38	26.49
	PSO	2382.79	2441.32	2493.43	28.45
	LSHADE-cnEpSin	2334.58	2363.38	2382.87	14.12
F22	TLMPA	**2300.00**	3377.69	7938.46	2156.88
	MPA	2408.85	2476.42	2593.93	50.28
	GWO	2718.12	5824.30	8845.78	2618.08
	DE	2430.91	5090.69	7928.37	2483.25
	SSA	2550.35	7770.29	9856.78	2781.48
	TLBO	**2300.00**	2639.68	6539.35	1054.75
	PSO	**2300.00**	5837.70	7506.35	1077.88
	LSHADE-cnEpSin	**2300.00**	**2409.21**	**2498.75**	**37.82**
F23	TLMPA	**2400.00**	**2677.34**	**2709.98**	67.89
	MPA	2637.19	2749.12	2802.82	39.81
	GWO	2852.26	2878.28	2920.24	17.23
	DE	2756.43	2772.96	2785.73	**10.44**
	SSA	2847.97	2894.49	2944.82	28.01
	TLBO	2755.95	2845.00	3080.52	69.92
	PSO	2755.24	2826.90	2894.98	42.90
	LSHADE-cnEpSin	2673.36	2742.36	2847.35	45.44
F24	TLMPA	**2842.17**	**2867.08**	**2892.04**	**13.17**
	MPA	2912.35	2960.03	2991.72	22.20
	GWO	3016.32	3055.13	3105.99	27.17
	DE	2928.05	2968.37	2990.98	13.93
	SSA	3021.42	3051.54	3078.36	17.60
	TLBO	2878.36	2976.41	3087.08	53.25
	PSO	2910.14	3006.00	3156.88	54.80
	LSHADE-cnEpSin	2865.36	2932.69	3050.86	46.39
F25	TLMPA	**2883.44**	2886.67	**2888.24**	1.54
	MPA	2921.51	2969.72	3048.02	32.10
	GWO	2952.28	3048.51	3336.89	107.15
	DE	2886.84	2887.72	2888.73	**0.63**
	SSA	2967.03	3048.43	3164.29	49.65
	TLBO	2883.81	2917.80	2947.95	23.67
	PSO	2875.21	**2879.89**	2898.91	5.05
	LSHADE-cnEpSin	2883.76	2903.31	2967.33	23.02
F26	TLMPA	2900.00	3718.49	4353.77	587.31
	MPA	3379.85	3707.27	4708.74	429.31
	GWO	4150.01	5775.59	6611.84	491.37
	DE	4522.55	4735.17	4946.21	117.96
	SSA	5708.05	6249.01	7092.82	355.45
	TLBO	**2800.00**	5836.38	8324.18	1629.29
	PSO	4191.27	5434.26	6474.24	614.47
	LSHADE-cnEpSin	**2800.00**	**3680.43**	**4169.71**	**98.36**
F27	TLMPA	**3182.70**	3202.47	3213.52	7.73
	MPA	3203.98	3221.74	3249.77	11.53
	GWO	3244.41	3284.04	3357.98	32.82
	DE	3197.98	3209.42	3216.51	4.52
	SSA	3259.37	3283.29	3329.09	20.51
	TLBO	3218.40	3271.11	3389.73	41.83
	PSO	3200.01	**3200.01**	**3200.01**	**0.00**
	LSHADE-cnEpSin	3218.81	3269.20	3319.18	26.50
F28	TLMPA	**3100.00**	3147.18	**3213.98**	51.62
	MPA	3245.99	3337.53	3401.18	41.49
	GWO	3399.55	3499.69	3675.86	74.87
	DE	3182.83	3209.62	3257.45	19.61
	SSA	3321.01	3402.61	3471.63	35.52
	TLBO	3203.11	3237.56	3359.59	37.90
	PSO	3300.01	3300.01	3300.01	**0.00**
	LSHADE-cnEpSin	**3100.00**	**3130.84**	3332.05	77.83
F29	TLMPA	**3309.11**	**3443.20**	**3554.83**	**63.64**
	MPA	3317.46	3544.66	3801.36	132.48
	GWO	3767.02	3989.66	4225.02	148.42
	DE	3424.39	3574.70	3734.53	78.87
	SSA	3806.41	4249.43	4570.40	194.48
	TLBO	3681.04	4029.68	4374.03	223.91
	PSO	3451.70	3735.66	4110.12	172.96
	LSHADE-cnEpSin	3401.05	3640.10	3918.82	159.03
F30	TLMPA	5062.62	**5419.53**	**6333.27**	**384.87**
	MPA	7395.12	1.06E+04	1.45E+04	2082.83
	GWO	9.03E+06	2.19E+07	4.47E+07	1.00E+07
	DE	9333.19	1.54E+04	3.17E+04	6242.66
	SSA	3.68E+06	1.55E+07	3.55E+07	9.23E+06
	TLBO	5450.79	1.02E+04	1.58E+04	3147.32
	PSO	**3255.58**	9384.12	3.13E+04	7971.92
	LSHADE-cnEpSin	5229.21	6874.69	2.10E+04	3612.04

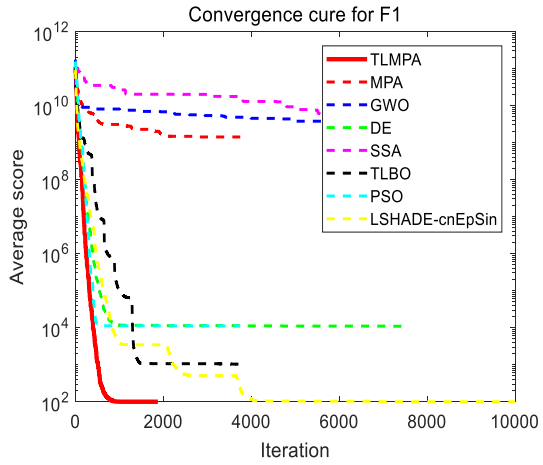

Figure 20. D=30, convergence cure for F1.

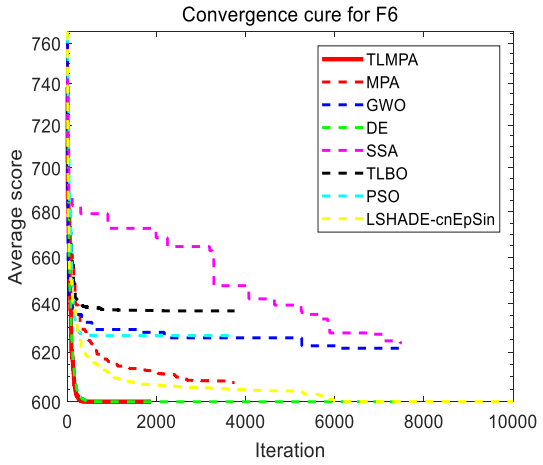

Figure 21. D=30, convergence cure for F6.

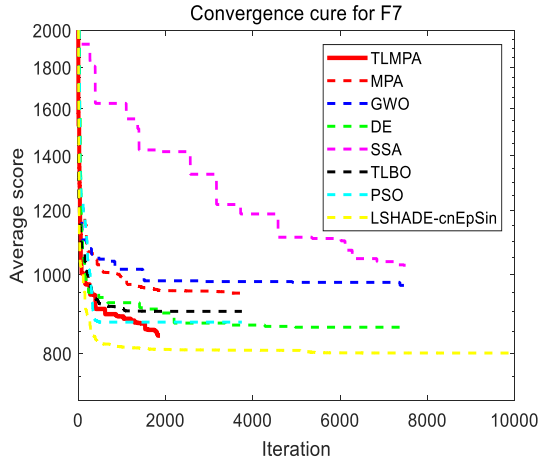

Figure 22. D=30, convergence cure for F7.

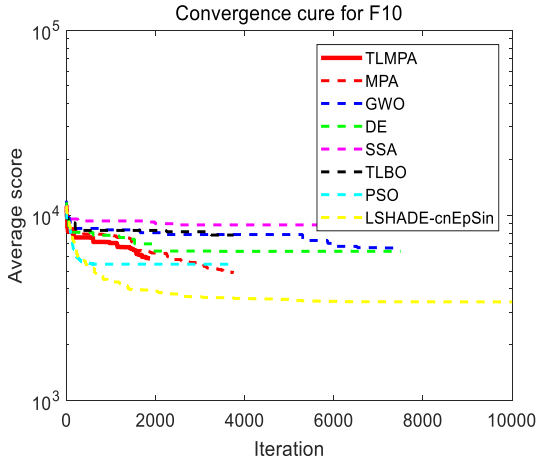

Figure 23. D=30, convergence cure for F10.

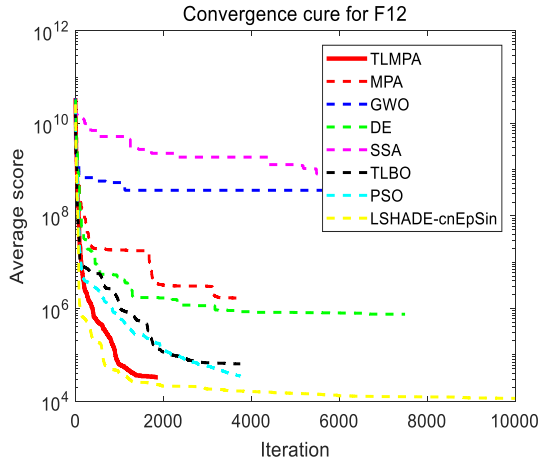

Figure 24. D=30, convergence cure for F12.

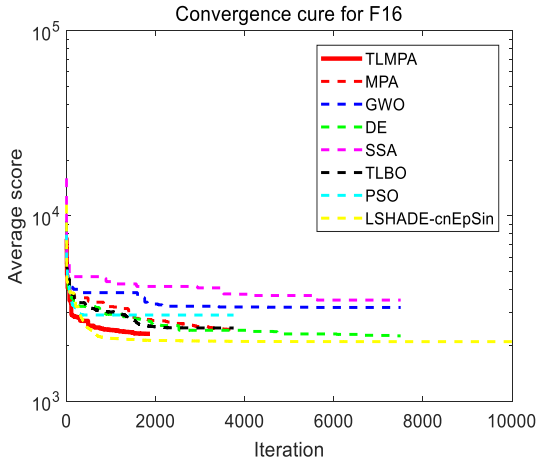

Figure 25. D=30, convergence cure for F16.

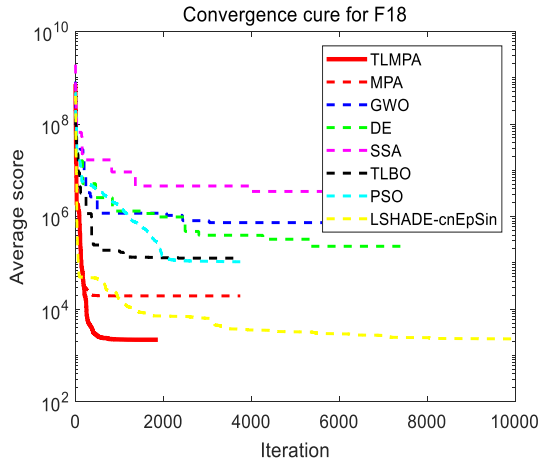

Figure 26. D=30, convergence cure for F18.

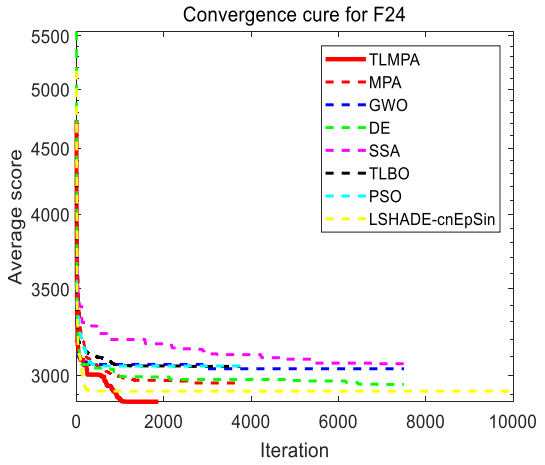

Figure 27. D=30, convergence cure for F24.

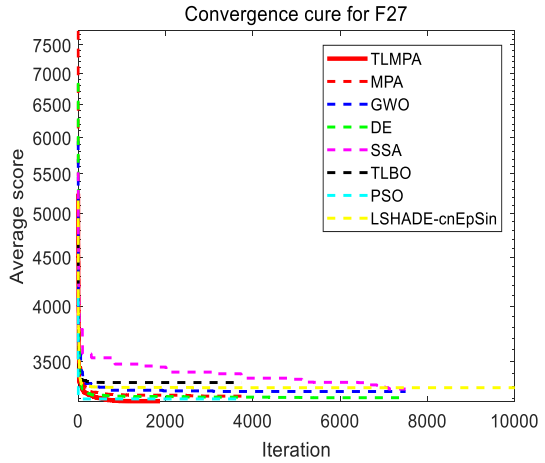

Figure 28. D=30, convergence cure for F27.

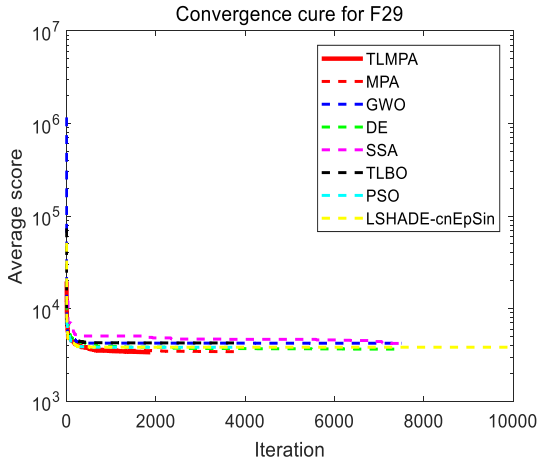

Figure 29. D=30, convergence cure for F29.

4.2. Statistical tests

Through the analysis of the previous experimental data, the test result of TLMPA is satisfactory, and it can be preliminarily considered that the proposed algorithm has strong competitiveness among all the comparison algorithms by virtue of its superior search performance. Statistics on the 10-dimensional and 30-dimensional experimental results are shown in Table 11. This table shows the average performance of the proposed algorithm is better than, equal to, or worse than other comparison algorithms.

It can be seen from the data in the table that in the case of 10 dimensions, the proposed algorithm is better than MPA, GWO, DE, SSA, TLBO, PSO, and LSAHDE-cnEpSin in 17, 29, 19, 29, 27, 28 and 12 benchmark functions respectively; TLMPA is equal to MPA, GWO, DE, SSA, TLBO, PSO, LSAHDE-cnEpSin in 0, 0, 2, 0, 1, 1, and 4 benchmark functions; the proposed algorithm is inferior to the comparison algorithm with 12, 0, 8, 0, 1 and 0 benchmark functions respectively. In the 30-dimensional test results, the number of TLMPA's performance better than the comparison algorithm is

25, 29, 26, 29, 28, 25, and 12 functions; the number of the proposed algorithm equal to the comparison algorithm is 0, 0, 0, 0, 0, 0, 0 and 2 benchmark functions; the number of TLMPA worse than the comparison algorithm is 4, 0, 3, 0, 1, 4 and 15 benchmark functions respectively.

Table 11. Statistical results of the IEEE CEC-2017 problems.

Algorithms	Dimension	Better	Equal	Less than
MPA	D=10	17	0	12
	D=30	25	0	4
GWO	D=10	29	0	0
	D=30	29	0	0
DE	D=10	19	2	8
	D=30	26	0	3
SSA	D=10	29	0	0
	D=30	29	0	0
TLBO	D=10	27	1	1
	D=30	28	0	1
PSO	D=10	28	1	0
	D=30	25	0	4
LSHADE-cnEpSin	D=10	12	4	13
	D=30	12	2	15

* Where "Better", "Equal" and "Less than" represent the number of problems that the performance of TLMPA is significantly better than, almost equal to, and significantly worse than the corresponding algorithm, respectively.

In addition, in order to verify that the results obtained are not accidental, a non-parametric wilcoxon non-parametric statistical test was conducted in this paper. Wilcoxon non-parametric statistical test returns a parameter called p-value. When p is less than 0.05, it means that there is a significant difference between the two algorithms in solving the problem; when p is greater than 0.05, it means that there is no significant difference between the two algorithms in solving the problem. Table 12 shows the results of the p-value test with a dimension of 10. From the table, in functions F10, F12, F13, F16, F21, F25, F26, F27 and F30, the proposed algorithm has no significant difference from MPA; in F16 and F29, the proposed algorithm has no significant difference with LSHADE-cnEpSin. In F11, F16, F20, F22, F23 and F29, there is no significant difference from DE; in F7 and F10, there is no significant difference from PSO. Overall, the proposed algorithm is significantly different from the comparison algorithm.

Table 12. Results of the p-value IEEE CEC-2017 sets (D = 10).

ID	MPA VS TLMPA	GWO VS TLMPA	DE VS TLMPA	SSA VS TLMPA	TLBO VS TLMPA	PSO VS TLMPA	LSHADE-cnEpSin VS TLMPA
F1	2.60E-06	1.73E-06	1.73E-06	1.73E-06	1.73E-06	1.73E-06	5.22E-06
F3	0.006232	1.73E-06	1.73E-06	1.73E-06	1.72E-06	1.72E-06	1.72E-06
F4	0.003162	1.73E-06	1.73E-06	1.73E-06	1.73E-06	2.60E-06	1.73E-06
F5	0.025637	1.73E-06	4.29E-06	1.73E-06	0.002105	0.001593	3.52E-06
F6	1.73E-06	1.73E-06	1	1.73E-06	1.73E-06	2.56E-06	1.73E-06
F7	1.02E-05	1.73E-06	4.73E-06	1.73E-06	**0.599936**	**0.404835**	5.75E-06
F8	1.73E-06	1.73E-06	3.72E-05	1.73E-06	1.73E-06	0.001965	6.89E-05
F9	1.73E-06	1.73E-06	1	1.73E-06	1.73E-06	3.94E-05	1.78E-05
F10	**0.975387**	0.0038542	1.92E-06	1.73E-06	**0.228880**	**0.926255**	2.35E-06
F11	**0.031603**	1.73E-06	**0.093676**	1.73E-06	1.73E-06	3.52E-06	5.75E-06
F12	**0.861213**	1.73E-06	1.73E-06	1.73E-06	1.73E-06	1.73E-06	1.92E-06
F13	**0.158855**	1.73E-06	1.92E-06	1.73E-06	1.73E-06	1.73E-06	2.35E-06
F14	**0.035009**	1.73E-06	7.51E-05	1.73E-06	1.73E-06	1.73E-06	1.73E-06
F15	**0.071903**	1.73E-06	2.35E-06	1.73E-06	1.73E-06	1.73E-06	1.73E-06
F16	**0.452807**	1.73E-06	**0.120445**	1.73E-06	0.00016	5.22E-06	**0.734325**
F17	1.73E-06	1.73E-06	0.011748	1.73E-06	1.73E-06	1.73E-06	6.89E-05
F18	1.73E-06	1.73E-06	1.92E-06	1.73E-06	1.73E-06	1.73E-06	1.73E-06
F19	3.72E-05	1.73E-06	4.07E-05	1.73E-06	1.73E-06	1.73E-06	1.73E-06
F20	1.73E-06	1.73E-06	**0.135296**	1.73E-06	1.73E-06	1.73E-06	1.23E-05
F21	**0.571646**	6.34E-06	7.69E-06	1.36E-05	0.000453	1.92E-06	0.000115
F22	0.001593	1.73E-06	**0.53044**	1.73E-06	0.000174	2.35E-06	0.001709
F23	**0.280214**	1.73E-06	**0.085896**	1.73E-06	0.000332	5.31E-05	0.14704
F24	0.024308	1.73E-06	6.34E-06	6.34E-06	1.64E-05	1.92E-06	6.89E-05
F25	**0.221022**	2.35E-06	0.002957	5.75E-06	0.000174	3.72E-05	0.000664
F26	**0.271155**	1.73E-06	0.001004	0.000189	0.000712	2.91E-05	0.013741
F27	**0.360039**	1.73E-06	0.000115	1.73E-06	1.92E-06	0.000616	3.88E-06
F28	0.000261	1.73E-06	4.29E-06	1.8E-05	5.79E-05	5.79E-05	0.000241
F29	0.006424	2.35E-06	**0.071903**	3.88E-06	0.000388	5.22E-06	**0.440522**
F30	**0.130592**	1.73E-06	5.22E-06	1.73E-06	1.73E-06	1.73E-06	0.000359

* Bold numbers represent the $p > 0.05$

Table 13 shows the results of the p-value test with a dimension of 30. From the table, only a few functions show that the proposed algorithm is not significantly different from the comparison algorithm. For example, in functions F4, F12, F16, F18, F22 and F28, the proposed algorithm has no significant difference from LSHADE-cnEpSin; in function F30, there is no significant difference from PSO; in F4 and F22, there is no significant difference from TLBO; in F16, F17, F20 and F26, there is no significant difference from MPA. In the remaining functions, TLMPA and other algorithms are significantly different. Therefore, the two-dimensional p-value test results show that TLMPA has superior performance.

In order to further verify the reliability of the experimental results, this paper also carries out the Friedman rank test [37] on the experimental results, and the test results are shown in Table 14. From the data in the table, it is easy to observe that whether it is 10-dimensional or 30-dimensional, the score of TLMPA can rank second, second only to LSHADE-cnEpSin. This also shows that the performance of TLMPA is better than most comparison algorithms.

Table 13. Results of the p-value IEEE CEC-2017 sets (D = 30).

ID	MPA VS TLMPA	GWO VS TLMPA	DE VS TLMPA	SSA VS TLMPA	TLBO VS TLMPA	PSO VS TLMPA	LSHADE-cnEpSin VS TLMPA
F1	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
F3	8.86E-05	8.86E-05	8.86E-05	8.86E-05	0.040044	8.86E-05	**0.1454**
F4	8.86E-05	8.86E-05	0.003592	8.86E-05	**0.331723**	0.043804	**0.681322**
F5	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	0.000103	0.025094
F6	8.86E-05	8.86E-05	0.015625	8.86E-05	8.86E-05	8.86E-05	8.86E-05
F7	8.86E-05	8.86E-05	0.000189	8.86E-05	8.86E-05	0.000338	0.03334
F8	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	0.004045
F9	8.86E-05	8.86E-05	6.1E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
F10	0.001162	0.00078	1	8.86E-05	0.003185	0.000219	8.86E-05
F11	8.86E-05	8.86E-05	0.001162	8.86E-05	8.86E-05	8.86E-05	8.86E-05
F12	8.86E-05	8.86E-05	8.86E-05	8.86E-05	0.043804	0.000189	**0.073138**
F13	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
F14	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
F15	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
F16	**0.1454**	0.000103	**0.550292**	8.86E-05	0.03334	0.000892	**0.350656**
F17	**0.525653**	0.000103	**0.295878**	8.86E-05	0.000189	8.86E-05	**0.82276**
F18	0.002495	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	**0.455273**
F19	0.000103	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05
F20	**0.433048**	8.86E-05	**0.82276**	8.86E-05	0.00039	8.86E-05	**0.079322**
F21	0.001325	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	0.000254
F22	**0.247145**	0.002495	0.01524	0.000293	**0.331723**	0.001162	**0.488708**
F23	0.000338	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	0.000338
F24	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	0.00014
F25	8.86E-05	8.86E-05	0.00455	8.86E-05	0.000254	0.001162	0.000892
F26	**0.654159**	8.86E-05	8.86E-05	8.86E-05	0.000449	8.86E-05	0.000449
F27	8.86E-05	8.86E-05	0.001162	8.86E-05	8.86E-05	**0.067355**	8.86E-05
F28	8.86E-05	8.86E-05	0.000293	8.86E-05	8.86E-05	8.86E-05	**0.501591**
F29	0.025094	8.86E-05	0.001162	8.86E-05	8.86E-05	0.000254	0.00078
F30	8.86E-05	8.86E-05	8.86E-05	8.86E-05	8.86E-05	**0.313463**	0.002495

* Bold numbers represent the $p > 0.05$

Table 14. Friedman rank test.

Algorithms	Dimensions	Mean rank	Rank	Dimensions	Mean rank	Rank
TLMPA	D = 10	2.45	2	D = 30	2.24	2
MPA	D = 10	2.97	3	D = 30	4.13	4
GWO	D = 10	6.98	8	D = 30	6.94	7
DE	D = 10	4.13	4	D = 30	4.05	3
SSA	D = 10	6.93	7	D = 30	7.27	8
TLBO	D = 10	4.87	5	D = 30	4.6	5
PSO	D = 10	5.32	6	D = 30	4.76	6
LSHADE-cnEpSin	D = 10	2.33	1	D = 30	2.01	1

4.3. CPU runtime comparison of algorithms

This section compares the CPU runtime of the proposed algorithm and the comparison algorithm in both dimensions. The CPU time is calculated by each algorithm under the same function evaluation times. The results are shown in Tables 15 and 16. Among them, Table 15 is the result of 10 dimensions, and Table 16 is the result of 30 dimensions. It should be noted that all times are measured in seconds.

TLMPA ranks fifth and sixth in most functions of the 10-dimensional running time, mainly because it is a hybrid algorithm, the structure is much more complicated than other algorithms, and the cost of running is higher than the original version. It is larger than other algorithms, but it is not the worst. The gap with the original algorithm is not very large, which is within an acceptable range. It can be seen from Table 15 that the average running time of TLMPA in F19, F20 and F21 is less than that of the original MPA; the average running time of TLMPA is slightly shorter than that of TLBO and significantly shorter than that of DE. This also shows that the TLMPA running time cost obtained by hybrid MPA with TLBO and DE is reasonable. Overall, the cost of TLMPA is acceptable.

Table 15. The result of CPU running time (D = 10).

Functions	TLMPA	MPA	GWO	DE	SSA	TLBO	PSO	LSHADE-cnEpSin	Rank
F1	2.0045	1.9448	1.6524	3.6529	0.9768	2.4742	1.1277	2.7826	5
F3	1.8265	1.7562	1.4420	3.2651	0.9440	2.1703	1.0440	2.2843	5
F4	1.8257	1.8160	1.4027	3.2735	0.9476	2.3389	1.0258	2.3751	5
F5	1.8370	1.7130	1.4072	3.0908	0.9632	2.1303	1.0572	2.4406	5
F6	2.0963	1.9066	1.7426	3.3781	1.0664	2.3397	1.2202	2.3848	5
F7	1.8145	1.7582	1.4427	3.2802	0.9415	2.1140	1.0250	2.2709	5
F8	1.9088	1.8904	1.4979	3.5158	0.9453	2.4594	1.1134	2.3320	5
F9	1.9214	1.8286	1.4777	3.2764	0.9539	2.3252	1.0707	2.3221	5
F10	1.8075	1.7143	1.3983	3.0979	0.9349	2.1012	1.0208	2.2393	5
F11	1.6556	1.5557	1.2634	2.8612	0.8771	1.9918	0.9214	2.1090	5
F12	1.6627	1.5669	1.2782	2.8669	0.8834	1.9225	0.9181	2.1635	5
F13	1.6770	1.5723	1.2789	2.8637	0.8918	1.9573	0.9189	2.1538	5
F14	2.1745	2.1033	1.6848	3.9154	0.9194	2.5549	1.2011	2.1110	6
F15	2.6890	2.6001	2.1402	4.6359	0.8875	3.0009	1.4732	2.1981	6
F16	2.7087	2.6316	2.1113	4.6410	0.8965	3.0325	1.4516	2.1424	6
F17	2.8771	2.7961	2.2817	4.8326	0.9604	3.1827	1.5847	2.1808	6
F18	2.6993	2.6522	2.1419	4.6401	0.8909	3.0064	1.4503	2.1889	6
F19	3.1028	3.2768	2.8262	5.7835	1.2897	3.6451	2.0889	2.5091	6
F20	2.5466	2.7114	2.1790	4.7039	0.9688	3.1292	1.5313	2.2257	5
F21	2.7497	2.7569	2.2391	4.8208	0.9662	3.2338	1.5668	2.1825	5
F22	2.9590	2.8552	2.3526	4.9141	1.0072	3.3462	1.6625	2.1708	6
F23	3.0003	2.8358	2.4046	4.9170	1.0210	3.3437	1.6943	2.2384	6
F24	3.0112	2.8589	2.4183	4.8843	1.0269	3.3324	1.6753	2.2206	6
F25	2.9275	2.8284	2.3143	4.9224	0.9936	3.3075	1.6408	2.1319	6
F26	3.0474	2.9017	2.4451	5.0104	1.0602	3.4365	1.7538	2.1998	6
F27	3.0569	2.9633	2.4566	5.1313	1.0667	3.4388	1.7321	2.3006	6
F28	3.0008	2.8554	2.4309	4.9435	1.0347	3.3709	1.7105	2.2524	6
F29	3.0287	2.8980	2.4049	5.0382	1.0374	3.3291	1.7002	2.2648	6
F30	3.3735	3.3505	2.9086	5.4548	1.3297	3.7801	2.1543	2.6331	6

Table 16. The result of CPU running time (D = 30).

Functions	TLMPA	MPA	GWO	DE	SSA	TLBO	PSO	LSHADE-cn EpSin	Rank
F1	7.6323	7.8573	6.5477	12.8939	3.0008	8.7418	4.3609	24.0413	4
F3	8.9862	11.6044	10.8899	20.4836	3.4399	13.0999	6.9188	36.5824	3
F4	9.9125	13.1458	11.8356	22.2012	3.7366	15.3958	8.1409	42.3000	3
F5	11.8052	14.9568	12.9631	24.4922	3.4429	16.5668	8.9527	42.1545	3
F6	8.3555	10.4811	10.2049	17.5995	3.9172	11.0654	7.1083	29.1355	3
F7	7.4896	9.7797	8.6954	15.6796	3.1928	10.1980	5.7511	27.4862	3
F8	7.6358	9.8837	8.7459	15.9103	3.2140	10.3569	5.9559	27.7127	3
F9	7.6250	9.9744	9.2025	16.3137	3.2325	10.4066	6.0711	28.6323	3
F10	9.7661	13.0718	11.2039	20.9995	3.5378	12.0954	7.5865	33.2026	3
F11	10.2840	12.8474	11.2483	20.4498	3.0795	12.9829	7.6781	35.6709	3
F12	8.9658	12.4452	11.8844	19.8800	3.1871	12.9300	7.7268	36.6939	3
F13	10.4096	13.7960	12.1947	20.7861	3.0662	13.5547	7.7410	37.5436	3
F14	10.6064	15.1606	15.2805	26.4041	3.3904	16.4911	10.3368	45.2841	3
F15	10.7887	14.0818	13.6080	23.4470	3.0047	15.3276	8.5517	41.5351	3
F16	11.1420	14.1826	13.8365	23.6066	3.0408	15.3722	8.8247	40.5553	3
F17	13.2182	19.3079	19.4721	32.8018	3.6738	21.3092	12.7583	52.4288	3
F18	15.6245	24.1582	22.2119	39.0985	3.4507	24.2287	13.3299	64.0582	3
F19	20.0394	29.5269	38.7540	53.6594	6.2993	30.1586	27.0481	86.6345	2
F20	24.1959	34.2016	32.1317	52.7352	4.1005	33.8648	21.0786	85.3409	3
F21	58.6689	65.8781	60.1733	100.9663	4.1584	68.8475	43.7507	159.7990	3
F22	78.7354	88.4615	80.5821	138.2244	4.3875	98.9447	59.5139	213.6537	3
F23	7.4137	7.4702	6.5517	11.0608	4.5385	7.8377	4.8477	19.7776	3
F24	8.3268	7.9334	7.7774	11.3711	4.8828	8.6093	5.5660	20.7303	5
F25	7.2586	7.1815	6.4466	10.4546	4.4719	7.6758	4.6563	19.0208	5
F26	8.2982	7.9846	7.5041	11.2277	5.1874	8.1912	5.6288	20.0805	6
F27	8.3007	8.0337	7.5452	11.3073	5.3941	8.1172	5.5275	19.9938	6
F28	7.9623	7.6016	7.1746	11.0299	4.9340	7.9203	5.0781	19.4779	6
F29	7.1416	7.0002	6.2695	10.1380	4.3787	7.3404	4.4758	18.7736	5
F30	9.8156	8.8998	8.6211	12.1276	6.7368	9.3287	6.8733	20.8718	6

At 30-dimensional, TLMPA performs better than 10-dimensional, ranking third in most functions, and the running time in most functions is smaller than the original version, and it is also less than the running time of most algorithms. This also just shows that TLMPA can fully reflect the stability of its algorithm performance when dealing with higher-dimensional problems. In F19, TLMPA even ranked second with good results; both ranked second in mixed functions. Generally speaking, the running time of TLMPA in 30 dimensions is shorter than that of most algorithms, which reflects its reliable and stable algorithm performance.

4.4. Result analysis

There are several well-evaluated reasons for the superior performance of the proposed TLMPA algorithm compared to other competitors. First, use the effective mutation crossover scheme of the DE algorithm and the fast search ability of the TLBO algorithm, and find a better solution in each iteration process to replace the worst agent. This factor significantly enhances the global search performance of the algorithm. Secondly, in the second phase of the proposed algorithm, the two learning behaviors of "teacher" and "learner" of the TLBO algorithm are carried out, which makes the mid-stage of the proposed algorithm a good balance between exploration ability and exploitation ability. After the cross mutation strategy of the DE algorithm, the diversity of the population is increased, and premature convergence is also effectively avoided. Finally, the proposed algorithm

retains the *FADs* effect in the original MPA, which has a certain auxiliary effect on the algorithm to avoid falling into the local optimum.

In the initial phase of the proposed algorithm, each individual is learning from teachers (top predators), which makes it easy for the population to gather closer to the teacher, and the search speed is fast, but it also causes the diversity of the population to be easily lost prematurely. And then fall into local search. After adding the crossover mutation strategy, the diversity of the population is increased and the diversity of the lost population is made up for. In the mid-stage, both exploration and exploitation are underway. After each individual learns the experience and knowledge imparted by the top predator (this is regarded as exploration behavior), they communicate and interact with other individuals (this is regarded as exploitation behavior), and learn from each other's strengths, making their own prey information more reliable and accurate. In this phase, the proposed algorithm uses the enhanced diversification trend to start searching for trends, and then smoothly transfers the initial search task to exploitation. In the later phase of the proposed algorithm, "learners" exerted their own learning ability, coupled with an effective crossover mutation mechanism, so that the algorithm can maintain the characteristics of population diversity and continue to search for the optimal solution.

5. Engineering design problem

The above experimental parts are all unconstrained function optimization problems, but many optimization problems in the real world are often accompanied by complex constraints; especially engineering structural design optimization has a large number of constraints. In order to verify the performance of the algorithm in constrained optimization problems, four engineering optimization problems in the structural field were tested, namely welded beam design problems, multi-disc clutch brake design problems, pressure vessel design problems, Tension/compression spring design problems. And compare with several other algorithms.

5.1. Welded beam design problem

The optimization goal of the welded beam design problem is the seven constraints related to shear stress (τ), beam bending stress (θ), bar buckling load (P_c), beam end deflection (δ), normal stress (σ) and boundary under the conditions, the welded beam is designed with the minimum manufacturing cost. Welded beam structure as shown in Figure 30. Variable x_1 refers to the thickness of the welded beam, x_2 refers to the length of the welded joint, x_3 refers to the width of the welded beam, x_4 refers to the thickness of the beam, and the mathematical formula can be expressed as:

$$\text{Minimize } f(X) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14 + x_2)$$

$$s.t.g_1(X) = \tau(X) - \tau_{\max}$$

$$g_2(X) = \sigma(X) - \sigma_{\max}$$

$$g_3(X) = x_1 - x_4 \leq 0$$

$$g_4(X) = 0.125 - x_1 \leq 0$$

$$g_5(X) = \delta(X) - 0.25 \leq 0$$

$$g_6(X) = P - P_c(X) \leq 0$$

$$g_7(X) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x_2) - 5 \leq 0$$

$$0.1 \leq x_1 \leq 2; 0.1 \leq x_2 \leq 10; 0.1 \leq x_3 \leq 10; 0.1 \leq x_4 \leq 2$$

where, $\tau_{\max}$ is the maximum acceptable shear stress, $\sigma_{\max}$ is the maximum acceptable normal stress, and P is the load. Calculated as follows:

$$\tau(X) = \sqrt{\tau_1^2 + 2\tau_1\tau_2\left(\frac{x_2}{2R}\right) + \tau_2^2} \tag{18}$$

$$\tau_1 = \frac{P}{\sqrt{2}x_1x_2} \tag{19}$$

$$\tau_2 = \frac{MR}{J} \tag{20}$$

$$M = P\left(L + \frac{x_2}{2}\right) \tag{21}$$

$$J(X) = 2\left\{\sqrt{2}\, x_1x_2\left[\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2\right]\right\} \tag{22}$$

M in Eq 21 and $J(X)$ in Eq 22 represent moment of inertia and polarity, respectively, and the remaining parameters are shown in Eqs 23–27.

$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2} \tag{23}$$

$$\sigma(X) = \frac{6PL}{x_4x_3^2} \tag{24}$$

$$\delta(X) = \frac{6PL^3}{Ex_3^3x_4} \tag{25}$$

$$P_c(X) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right) \tag{26}$$

$$G = 12 \times 10^6 psi, E = 30 \times 10^6 psi, P = 6000lb, L = 14in \tag{27}$$

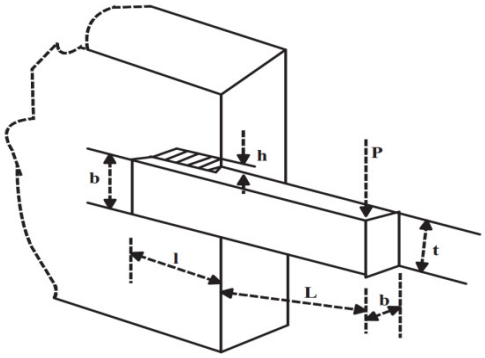

Figure 30. The structure of welded beam design.

Table 17 records the comparative experimental results of the optimal solution of the welded beam design problem. From the table, the proposed algorithm shows superior performance in solving the welded beam design problem, and the optimal value obtained by the test is lower than the previous research result. All tests are run independently 30 times, and finally the average value is taken as the test result. The best fitness value found by the TLMPA is $f(X) = 1.724852$, and the corresponding optimal solution is $X = [0.20572964, 3.470488666, 9.03662391, 0.20572964]$. It can be explained that TLMPA has better optimization accuracy in solving spring pressure design problems.

Table 17. Comparative results for the welded beam design problem.

Algorithms	Optimum variables				Optimum cost
	h	l	t	b	
TLMPA	0.20572964	3.470488666	9.03662391	0.20572964	**1.724852**
SSA [38]	0.205700	3.471400	9.036600	0.205700	1.724910
GA3 [39]	0.205986	3.471328	9.020224	0.206480	1.728226
GSA [40]	0.182129	3.856979	10	0.202376	1.879952
WCA [41]	0.205728	3.470522	9.036620	0.205729	1.724856
CPSO [42]	0.202369	3.544214	9.048210	0.205723	1.728024
Random [43]	0.457500	4.731300	5.085300	0.660000	4.118500
David [43]	0.243400	6.255200	8.291500	0.244400	2.384110
Simple [43]	0.279200	5.625600	7.751200	0.279600	2.530730
ACO [45]	0.205700	3.471131	9.036683	0.205731	1.724918
ESs [68]	0.199742	3.61206	9.0375	0.206082	1.7373
CDE [69]	0.203137	3.542998	9.033498	0.206179	1.733462

5.2. Multi-plate disc clutch brake design problem

In this discrete benchmark task, the goal is to optimize the total weight of the multi-disc clutch brake, involving five variables: driving force (F), inner and outer radius ($r1$ and $r0$), number of friction surfaces (Z) and disc thickness (t). Figure 31 shows a multi-disc clutch brake. Since this problem contains 8 different constraints, the difficulty of solving the optimization problem is increased, and the feasible region in the solution space only accounts for 70%. The mathematical formula can be expressed as:

$$\text{Minimize } f(x) = \pi\left(x_2^2 - x_1^2\right) x_3 (x_5 + 1)\rho$$

$$s.t. g_1(x) = x_2 - x_1 - \Delta R \geq 0$$

$$g_2(x) = L_{\max} - (x_5 + 1)(x_3 + \delta) \geq 0$$

$$g_3(x) = P_{\max} - P_{rz} \geq 0, g_4(x) = P_{\max} v_{sr\max} - P_{rz} v_{sr} \geq 0,$$

$$g_5(x) = v_{sr\max} - v_{sr} \geq 0, g_6(x) = T_{\max} - T \geq 0$$

$$g_7(x) = M_h - sM_s \geq 0, g_8(x) = T \geq 0$$

$$\text{where } M_h = \frac{2}{3}\mu x 4 x 5 \frac{x_2^3 - x_1^3}{x_2^2 - x_1^2}, w = \frac{\pi n}{30} rad/s$$

$$A = \pi\left(x_2^2 - x_1^2\right) mm^2,$$

$$P_{rz} = \frac{x_4}{A} N/mm^2, V_{sr} = \frac{\pi R_{sr} n}{30} mm/s, R_{sr} = \frac{2}{3}\frac{x_2^2 - x_1^2}{x_2^2 x_1^2} mm$$

$$T = \frac{I_z \pi n}{30\left(M_h + M_f\right)} mm, \Delta r = 20mm, L_{\max} = 30mm, \mu = 0.6$$

$$T_{\max} = 15s, \mu = 0.5, s = 1.5, M_s = 40Nm,$$

$$p_{\max} = 1Mp_a, \rho = 0.0000078kg/mm^3,$$

$$v_{sr\max} = 10m/s, \delta = 0.5mm, s = 1.5$$

$$T_{\max} = 15s, n = 250rpm, I_z = 55kg/m^2,$$

$$M_s = 40Nm, M_f = 3Nm$$

$$60 \leq x_1 \leq 80, 90 \leq x_2 \leq 110, 1 \leq x_3 \leq 3,$$

$$60 \leq x_4 \leq 1000, 2 \leq x_5 \leq 9, i = 1, 2, 3, 4, 5$$

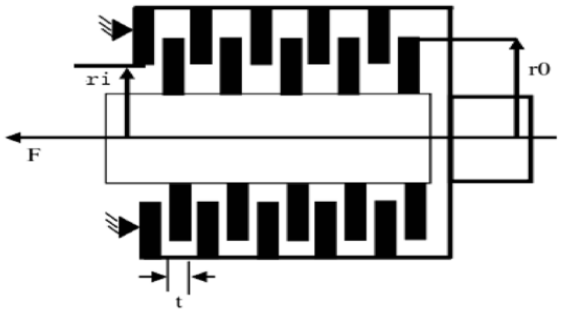

Figure 31. Multi-plate disc clutch brake problem.

Table 18 records the comparison experiment of the optimal solution of the multi-disc clutch separator. Judging from the table, the proposed algorithm shows superior performance in solving the design problem of multi-disc clutch separator, and the test results are better than previous research results. All tests are run independently 30 times, and finally the average value is taken as the test result. The best fitness value found by the TLMPA algorithm is f(X) = 0.235242458, and the corresponding optimal solution is X= [70, 90, 1, 703.3836952, 2]. It can be explained that TLMPA has better optimization accuracy in solving the design problem of multi-disc clutch separator.

Table 18. Comparative results for multi-plate disc clutch brake problem.

Algorithms	ri(x1)	r0(x2)	t(x3)	F(x4)	Z(x5)	Optimal Cost
TLMPA	70	90	1	703.3836952	2	**0.235242458**
WCA [41]	70	90	1	910	3	0.313656
TLBO [11]	70	90	1	810	3	0.313656
PVS [44]	70	90	1	980	3	0.31366

5.3. Pressure vessel design problem

The optimization goal of the pressure vessel problem is to minimize the total cost. Constraints include material costs, molding costs, and welding costs. There are lids on both ends of the container, and it has a hemispherical head. The design structure of the pressure vessel is shown in Figure 32. The four variables in this problem: shell thickness (Ts), head thickness (Th), inner diameter (R), and cylindrical section length (L) of the vessel [73]. The mathematical model of pressure vessel design is as follows:

$$\text{Consider } Z = [z_1, z_2, z_3, z_4] = [h, l, t, b]$$

$$\text{Minimize } f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_2^3 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

$$s.t.g_1(x) = -x_1 + 0.0193x_3 \leq 0$$

$$g_2(x) = -x_3 + 0.00954z_3 \leq 0$$

$$g_3(x) = -\pi x_3^2 x_4 - \tfrac{4}{3}\pi x_3^3 + 1,296,00 \leq 0$$

$$g_4(x) = x_4 - 240 \leq 0$$

Variable range $0 \leq x_1, x_2 \leq 99, 0 \leq x_3, x_4 \leq 200$

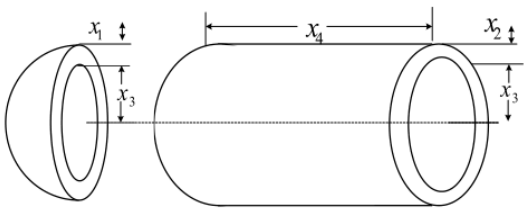

Figure 32. Pressure vessel design and its features.

Table 19 records the comparison experiment of the optimal value of the pressure vessel design problem. From the table, the proposed algorithm shows strong optimization performance in solving pressure vessel design problems, and the optimal value obtained by the test is lower than the result value of other algorithms. All tests are run independently 30 times, and finally the average value is taken as the test result. The optimal fitness value found by the TLMPA algorithm is f(X) = 5885.332774, and the corresponding optimal solution is X= [0.778168641, 0.384649163, 40.31961872, 200]. This shows that TLMPA has better optimization accuracy in solving pressure vessel design problems.

Table 19. Comparative results for the pressure vessel design problem.

Algorithms	Optimum variables				Optimum cost
	T_s	T_h	R	L	
TLMPA	0.778169	0.384649	40.319618	200	**5885.332774**
ACO [45]	0.812500	0.437500	42.098353	176.637751	6059.7258
WOA [46]	0.812500	0.437500	42.0982699	176.638998	6059.7410
SCA [47]	0.8125	0.4378	42.0883699	176.648998	6058.2907
ES [48]	0.812500	0.437500	42.098087	176.640518	6059.7456
VPL [49]	0.815200	0.426500	42.0912541	176.742314	6043.986
PSO-DE [50]	0.812500	0.437500	42.098446	176.636600	6059.71433
RCSA [51]	0.9803	0.4854	50.7236	92.7062	6335.4270
GAS [52]	0.937500	0.50000	48.329	112.679	6410.3811
MFO [62]	0.8125	0.437500	42.098445	176.636596	6059.7143
BA [65]	0.812500	0.437500	42.098445	176.636595	6059.7143
IACO [66]	0.812500	0.437500	42.098353	176.637751	6059.7258
G-QPSO [67]	0.812500	0.437500	42.0984	176.6372	6059.7208
BIANCA [70]	0.812500	0.437500	42.096800	176.658000	6059.9384
CSS [72]	0.812500	0.437500	42.103624	176.572656	6059.0888

5.4. Tension/compression spring design problem

The optimization goal of the tension/compression spring (TCS) design problem is to minimize the weight of the spring. The constraints are the minimum deflection (g_1(X)), shear stress (g_2(X)), impact frequency (g_3(X)) and outer diameter limit (g_4(X)). For design drawings, see Figure 33. Setting variable x_1 refers to the diameter of the coil, x_2 refers to the diameter of the coil, and x_3 refers to the number of coils. The mathematical formula can be expressed as:

$$\text{Minimize } f(X) = (x_3 + 2)\, x_2 x_1^2$$

$$s.t. g_1(X) = 1 - \frac{x_2^3 x_3}{71785 x_1^4} \leq 0$$

$$g_2(X) = \frac{4x_2^2 - x_1 x_2}{12566\left(x_2 x_1^3 - x_1^4\right)} + \frac{1}{5180 x_1^2} - 1 \leq 0$$

$$g_3(X) = 1 - \frac{140.45 x_1}{x_2^2 x_3} \leq 0$$

$$g_4(X) = \frac{x_1 + x_2}{1.5} - 1 \leq 0$$

$$0.05 \leq x_1 \leq 2, 0.25 \leq x_2 \leq 1.3, 2 \leq x_3 \leq 15$$

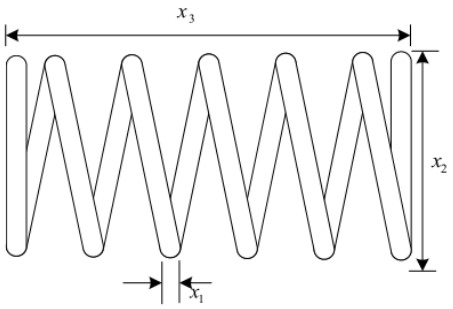

Figure 33. Structure of the TCS design.

Table 20 records the comparison experiment of the optimal value of the tension/compression spring design problem. It can be seen from the table that the optimal value obtained by the proposed algorithm in solving the pressure vessel design problem is lower than the previous research results. All tests are run independently 30 times, and finally the average value is taken as the test result. The optimal fitness value found by the TLMPA algorithm is f(X) = 0.012665236, and the corresponding optimal solution is X= [0.356532715, 11.29982336, 0.05168137]. It can be explained that TLMPA has better optimization accuracy in solving tension/compression spring design problems.

Table 20. Comparative results for the tension/compression spring problem.

Algorithms	Optimum variables			Optimum weight
	D	N	d	
TLMPA	0.356532715	11.29982336	0.05168137	**0.0126652**
IHS [53]	0.349871	12.076432	0.051154	0.0126706
SSA [38]	0.345215	12.004032	0.051207	0.0126763
SES [54]	N/A	N/A	N/A	0.012732
Ray and Saini [55]	0.050417	3.979915	0.321532	0.013060
Mathematical optimization [56]	0.399918	9.018540	0.053396	0.0127303
Ray and Liew [57]	0.3681587	10.648442	0.0521602	0.012669249
CMSSA [58]	0.393380	9.423987	0.053169	0.0127043
RO [59]	0.349096	11.76279	0.051370	0.0126788
CEDE [60]	0.354714	11.410831	0.051609	0.0126702
Montes and Coello [61]	0.051643	11.397926	0.355360	0.012698
Arora [63]	0.399180	9.185400	0.053396	0.012730
CWCA [64]	0.35710734	11.270826	0.051709	0.012672
GA2 [71]	0.351661	11.632201	0.051480	0.012704

6. Research limitations and future work

Although the results shown above all show that TLMPA performs better than most comparison algorithms, it also has some shortcomings. First of all, there is still room for improvement in the performance of TLMPA. For example, in the test of the multimodal function, the number of theoretical optimal values found is small, indicating that the balance of exploration and development capabilities in multimodal functions with multiple optimal values needs to be further improved. Secondly, the test results of TLMPA in 10 dimensions are slightly worse than those in 30 dimensions, which indicate that TLMPA is good at optimization problems with higher dimensions. This study only tests 10 and 30 dimensional problems, but also needs to be tested in higher dimensional problems, such as 50 and 100 dimensions. Finally, because TLMPA hybrid the ideas of three algorithms, its algorithm structure are a bit more complicated than the original version, so its flexibility and lightness are worse than the original algorithm.

In the future, we will evaluate the proposed TLMPA algorithm on more complex benchmark problems. TLMPA can be applied to many exciting scenarios, such as applying the binary version of TLMPA to discrete and binary problems [73–76]; applying TLMPA to optimization problems such as scheduling [77–79], neural network training [80–82], and feature selection [83–85]. Various constraint processing techniques can also be integrated into TLMPA to introduce its effective constraint version. In addition, multi-objective optimization is a combinatorial optimization problem, which requires the algorithm to maintain diversity as much as possible, while TLMPA can increase population diversity, and it is easier to find Pareto optimal solution. Moreover, the search speed and accuracy of TLMPA algorithm are better than most of the comparison algorithms. Therefore, using TLMPA algorithm to solve the multi-objective optimization problem will produce good results.

7. Conclusions

This article proposes a hybrid version of MPA and TLBO, TLMPA, which aims to maintain the proper synergy between exploration and exploitation in the search zone. First, the update mechanism of the original algorithm was improved, and the "teachers" and "learners" of TLBO were introduced to

establish a better balance between exploration and exploitation. In addition, in order to avoid making up for the loss of population diversity in the early phase of the algorithm, the effective mutation crossover scheme of the differential evolution algorithm is added to the algorithm, which greatly enhances the algorithm's global search performance. In terms of performance evaluation, the extremely challenging CEC-2017 benchmark test set was used to study the robustness of the TLMPA algorithm in terms of scalability in two dimensions with dimensions of 10 and 30. The analysis of statistical test and result comparison shows that the search performance of TLMPA is better than the original MPA and other optimization methods. In addition, the results of engineering design problems such as welded beam design, multi-disc clutch brake, pressure vessel design, tension/compression spring design were compared, which also proved that TLMPA has better search performance than other algorithms.

Acknowledgments

This work is supported by National Science Foundation of China under Grant 62066005, 61563008, and by the Project of Guangxi Natural Science Foundation under Grants No. 2018GXNSFAA138146.

Conflict of interest

The author declares that they has no conflict of interest.

References

1. J. H. Holland, Genetic algorithms, *Sci. Am.,* **267** (1992), 66–72.
2. J. Kennedy, R. Eberhart, *Particle swarm optimization,* Perth, WA, Australia: Proceedings of IEEE International Conference on Neural Networks, 1995.
3. R. Storn, K. Price, Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optim.,* **11** (1997), 341–359.
4. K. V. Price, *Differential evolution: A fast and simple numerical optimizer,* Berkeley, CA, USA: Proceedings of North American Fuzzy Information Processing, 1996.
5. D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm, *J. Global Optim.,* **39** (2007), 459–471.
6. C. R. Hwang, Simulated annealing: Theory and applications, *Acta Appl. Math.,* **12** (1988), 108–111.
7. A. Faramarzi, M. Heidarinejad, S. Mirjalili, A. H. Gandomi, Marine Predators algorithm: A nature-inspired metaheuristic, *Expert Syst. Appl.,* **152** (2020), 113377.
8. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Software,* **69** (2014), 46–61.
9. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Software,* **95** (2016), 51–67.
10. S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, S. M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, *Adv. Eng. Software*, **114** (2017), 163–191.

11. R. V. Rao, V. J. Savsani, D. P. Vakharia, Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems, *Comput.-Aided Des.,* **43** (2011), 303–315.
12. D. H. Wolpert, W. G. Macready, No free lunch theorems for optimization, *IEEE Trans. Evol. Comput.,* **1** (1997), 67–82.
13. M. Liu, X. Yao, Y. Li, Hybrid whale optimization algorithm enhanced with Lévy flight and differential evolution for job shop scheduling problems, *Appl. Soft Comput.,* **87** (2020), 105954.
14. D. Tansui, A. Thammano, Hybrid nature-inspired optimization algorithm: Hydrozoan and sea turtle foraging algorithms for solving continuous optimization problems, *IEEE Access,* **8** (2020), 65780–65800.
15. H. Garg, A hybrid GSA-GA algorithm for constrained optimization problems, *Inf. Sci.,* **478** (2019), 499–523.
16. D. T. Le, D. K. Bui, T. D. Ngo, Q. H. Nguyen, H. Nguyen-Xuan, A novel hybrid method combining electromagnetism-like mechanism and firefly algorithms for constrained design optimization of discrete truss structures, *Comput. Struct.,* **212** (2019), 20–42.
17. N. E. Humphries, N. Queiroz, J. R. M. Dyer, N. G. Pade, M. K. Musyl, K. M. Schaefer, et al., Environmental context explains Lévy and Brownian movement patterns of marine predators, *Nature,* **465** (2010), 1066–1069.
18. F. Bartumeus, J. Catalan, U. L. Fulco, M. L. Lyra, G. M. Viswanathan, Erratum: Optimizing the encounter rate in biological interactions: Lévy versus brownian strategies, *Phys. Rev. Lett.,* **89** (2002), 109902.
19. M. A. A. Al-qaness, A. A. Ewees, H. Fan, L. Abualigah, M. A. Elaziz, Marine Predators algorithm for forecasting confirmed cases of COVID-19 in Italy, USA, Iran and Korea, *Int. J. Environ. Res. Public Health,* **17** (2020), 3520.
20. D. Yousri, T. S. Babu, E. Beshr, M. B. Eteiba, D. Allam, A robust strategy based on marine predators algorithm for large scale photovoltaic array reconfiguration to mitigate the partial shading effect on the performance of PV system, *IEEE Access,* **8** (2020), 112407–112426.
21. M. Abdel-Basset, R. Mohamed, M. Elhoseny, R. K. Chakrabortty, M. Ryan, A hybrid COVID-19 detection model using an improved Marine Predators algorithm and a ranking-based diversity reduction strategy, *IEEE Access,* **8** (2020), 79521–79540.
22. M. A. Elaziz, A. A. Ewees, D. Yousri, H. S. N. Alwerfali, Q. A. Awad, S. Lu, et al., An improved Marine Predators algorithm with fuzzy entropy for multi-level thresholding: Real world example of COVID-19 CT image segmentation, *IEEE Access,* **8** (2020), 125306–125330.
23. R. V. Rao, V. Patel, Multi-objective optimization of heat exchangers using a modified teaching-learning-based optimization algorithm, *Appl. Math. Modell.,* **37** (2013), 1147–1162.
24. R. V. Rao, V. Patel, An improved Teaching-learning-based optimization algorithm for solving unconstrained optimization problems, *Sci. Iran.,* **20** (2013), 710–720.
25. A. R. Yildiz, Optimization of multi-pass turning operations using hybrid teaching learning-based approach, *Int. J. Adv. Manuf. Technol.,* **66** (2013), 1319–1326.
26. K. Yu, X. Wang, Z. Wang, An improved Teaching-learning-based optimization algorithm for numerical and engineering optimization problems, *J. Intell. Manuf.,* **27** (2016), 831–843.

27. E. Uzlu, M. Kankal, A. Akpınar, T. Dede, Estimates of energy consumption in Turkey using neural networks with the Teaching-learning-based optimization algorithm, *Energy,* **75** (2014), 295–303.

28. V. Toğan, Design of planar steel frames using teaching-learning based optimization, *Eng. Struct.,* **34** (2012), 225–232.

29. R. V. Rao, V. D. Kalyankar, Parameter optimization of modern machining processes using teaching-learning-based optimization algorithm, *Eng. Appl. Artif. Intell.,* **26** (2013), 524–531.

30. M. Singh, B. K. Panigrahi, A. R. Abhyankar, S. Das, Optimal coordination of directional over-current relays using informative differential evolution algorithm, *J. Comput. Sci.,* **5** (2014), 269–276.

31. H. Bouchekara, M. A. Abido, M. Boucherma, Optimal power flow using Teaching-learning-based optimization technique, *Electr. Power Syst. Res.,* **114** (2014), 49–59.

32. G. M. Viswanathan, V. Afanasyev, S. V. Buldyrev, E. J. Murphy, P. A. Prince, H. E. Stanley, Lévy flight search patterns of wandering albatrosses, *Nature,* **381** (1996), 413–415.

33. J. D. Filmalter, L. Dagorn, P. D. Cowley, M. Taquet, First descriptions of the behavior of silky sharks, Carcharhinus falciformis, around drifting fish aggregating devices in the Indian Ocean, *Bull. Mar. Sci.,* **87** (2011), 325–337.

34. E. Clark, Instrumental conditioning of lemon sharks, *Science (New York, N.Y.),* **130** (1959), 217–218.

35. L. A. Dugatkin, D. S. Wilson, The prerequisites for strategic behaviour in bluegill sunfish, Lepomis macrochirus, *Anim. Behav.,* **44** (1992), 223–230.

36. V. Schluessel, H. Bleckmann, Spatial learning and memory retention in the grey bamboo shark (Chiloscyllium griseum), *Zoology,* **115** (2012), 346–353.

37. D. W. Zimmerman, B. D. Zumbo, Relative power of the Wil-coxon test, the Friedman test, and repeated-measures ANOVA on ranks, *J. Exp. Educ.,* **62** (1993), 75–86.

38. S. Mirjalili, A. H. Gandomi, S. Z. Mirjalili, S. Saremi, H. Faris, S. M. Mirjalili, Salp swarm algorithm: A bio-inspired optimizer for engineering design problems, *Adv. Eng. Software,* **114** (2017), 163–191.

39. C. A. C. Coello, E. M. Montes, Constraint-handling in genetic algorithms through the use of dominance-based tournament selection, *Adv. Eng. Inf.,* **16** (2002), 193–203.

40. E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: A gravitational search algorithm, *Inf. Sci.,* **179** (2009), 2232–2248.

41. H. Eskandar, A. Sadollah, A. Bahreininejad, M. Hamdi, Water cycle algorithm-a novel metaheuristic optimization method for solving constrained engineering optimization problems, *Comput. Struct.,* **110** (2012), 151–166.

42. R. A. Krohling, L. dos Santos Coelho, Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems, *IEEE Trans. Syst., Man, Cybern., Part B (Cybernetics),* **36** (2006), 1407–1416.

43. K. M. Ragsdell, D. T. Phillips, Optimal design of a class of welded structures using geometric programming, *J. Eng. Ind.,* **98** (1976), 1021–1025.

44. P. Savsani, V. Savsani, Passing vehicle search (PVS): A novel metaheuristic algorithm, *Appl. Math. Model.,* **40** (2016), 3951–3978.

45. M. Dorigo, T. Stützle, *Ant colony optimization: Overview and recent advances,* 2Eds., Cham, Switzerland: Springer International Publishing, 2019.

46. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Software,* **95** (2016), 51–67.

47. S. Mirjalili, SCA: A sine cosine algorithm for solving optimization problems, *Knowl.-Based Syst.,* **96** (2016), 120–133.

48. H. Beyer, H. Schwefel, Evolution strategies-A comprehensive introduction, *Nat. Comput.,* **1** (2002), 3–52.

49. R. Moghdani, K. Salimifard, Volleyball premier league algorithm, *Appl. Soft Comput.,* **64** (2018), 161–185.

50. H. Liu, Z. Cai, Y. Wang, Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization, *Appl. Soft Comput.,* **10** (2010), 629–640.

51. K. Thirugnanasambandam, S. Prakash, V. Subramanian, S. Pothula, V. Thirumal, Reinforced cuckoo search algorithm-based multimodal optimization, *Appl. Intell.,* **49** (2019), 2059–2083.

52. K. Deb, *GeneAS: A robust optimal design technique for mechanical component design,* 2Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1997.

53. M. Mahdavi, M. Fesanghary, E. Damangir, An improved harmony search algorithm for solving optimization problems, *Appl. Math. Comput.,* **188** (2007), 1567–1579.

54. E. Mezura-Montes, C. A. C. Coello, R. Landa-Becerra, *Engineering optimization using simple evolutionary algorithm,* Sacramento, CA, USA: Proceedings. 15th IEEE International Conference on Tools with Artificial Intelligence, 2003.

55. T. Ray, P. Saini, Engineering design optimization using swarm with an intelligent information sharing among individuals, *Eng. Optim.,* **33** (2001),735–748.

56. A. D. Belegundu, J. S. Arora, A study of mathematical programming methods for structural optimization. Part I: Theory, *Int. J. Numer. Methods Eng.,* **21** (1985), 1583–1599.

57. T. Ray, K. M. Liew, Society and civilization: An optimization algorithm based on the simulation of social behavior, *IEEE Trans. Evol. Comput.,* **7** (2003), 386–396.

58. Q. Zhang, H. Chen, A. A. Heidari, X. Zhao, Y. Xu, P. Wang, et al., Chaos-induced and mutation-driven schemes boosting salp chains-inspired optimizers, *IEEE Access,* **7** (2019), 31243–31261.

59. A. Kaveh, M. Khayatazad, A new meta-heuristic method: Ray optimization, *Comput. Struct.,* **112** (2012), 283–294.

60. F. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, *Appl. Math. Comput.,* **186** (2007), 340–356.

61. E. M. Montes, C. A. C. Coello, An empirical study about the usefulness of evolution strategies to solve constrained optimization problems, *Int. J. Gen. Syst.*, **37** (2008), 443–473.

62. S. Mirjalili, Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm, *Knowl.-Based Syst.,* **89** (2015), 228–249.

63. J. S. Arora, *Introduction to optimum design*, New York: McGraw-Hill Book Co., 1989.

64. A. A. Heidari, R. A. Abbaspour, A. R. Jordehi, An efficient chaotic water cycle algorithm for optimization tasks, *Neural Comput. Appl.,* **28** (2017), 57–85.

65. A. H. Gandomi, X. S. Yang, A. H. Alavi, S. Talatahari, Bat algorithm for constrained optimization tasks, *Neural Comput. Appl.,* **22** (2013), 1239–1255.

66. H. Rosenbrock, An automatic method for finding the greatest or least value of a function, *Comput. J.,* **3** (1960), 175–184.

67. L. dos Santos Coelho, Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems, *Expert Syst. Appl.,* **37** (2010), 1676–1683.

68. E. Mezura-Montes, C. A. C. Coello, A simple multimembered evolution strategy to solve constrained optimization problems, *IEEE Trans. Evol. Comput.,* **9** (2005), 1–17.

69. F. Huang, L. Wang, Q. He, An effective co-evolutionary differential evolution for constrained optimization, *Appl. Math. Comput.,* **186** (2007), 340–356.

70. M. Montemurro, A. Vincenti, P. Vannucci, The automatic dynamic penalisation method (ADP) for handling constraints with genetic algorithms, *Comput. Methods Appl. Mech. Eng.,* **256** (2013), 70–87.

71. K. Deb, Optimal design of a welded beam via genetic algorithms, *AIAA J.,* **29** (1991), 2013–2015.

72. A. Kaveh, S. Talatahari, A novel heuristic optimization method: Charged system search, *Acta Mech.,* **213** (2010), 267–289.

73. B. K. Kannan, S. N. Kramer, An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design, *J. Mech. Des.*, **116** (1994), 405–411.

74. C. Ozturk, E. Hancer, D. Karaboga, Dynamic clustering with improved binary artificial bee colony algorithm, *Appl. Soft Comput.,* **28** (2015), 69–80.

75. X. Kong, L. Gao, H. Ouyang, S. Li, A simplified binary harmony search algorithm for large scale 0-1 knapsack problems, *Expert Syst. Appl.,* **42** (2015), 5337–5355.

76. M. Abdel-Basset, D. El-Shahat, H. Faris, S. Mirjalili, A binary multi-verse optimizer for 0-1 multidimensional knapsack problems with application in interactive multimedia systems, *Comput. Ind. Eng.,* **132** (2019), 187–206.

77. P. Brucker, R. Qu, E. K. Burke, Personnel scheduling: Models and complexity, *Eur. J. Oper. Res.,* **210** (2011), 467–473.

78. M. M. Solomon, Algorithms for the vehicle routing and scheduling problems with time window constraints, *Oper. Res.,* **35** (1987), 254–265.

79. P. V. Laarhoven, E. Aarts, J. K. Lenstra, Job shop scheduling by simulated annealing, *Oper. Res.*, **40** (1992), 113–125.

80. J. Zhang, J. Zhang, T. Lok, M. R. Lyu, A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training, *Appl. Math. Comput.,* **185** (2007), 1026–1037.

81. K. Socha, C. Blum, An ant colony optimization algorithm for continuous optimization: Application to feed-forward neural network training, *Neural Comput. Appl.,* **16** (2007), 235–247.

82. K. Y. Leong, A. Sitiol, K. S. M. Anbananthen, *Enhance neural networks training using GA with chaos theory,* 3Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

83. X. Wang, J. Yang, X. Teng, W. Xia, R. Jensen, Feature selection based on rough sets and particle swarm optimization, *Pattern Recognit. Lett.,* **28** (2007), 459–471.

84. M. Ghaemi, M. R. Feizi-Derakhshi, Feature selection using forest optimization algorithm, *Pattern Recognit.,* **60** (2016), 121–129.

85. P. R. Varma, V. V. Kumari, S. S. Kumar, Feature selection using relative fuzzy entropy and ant colony optimization applied to real-time intrusion detection system, *Procedia Comput. Sci.,* **85** (2016), 503–510.

AIMS Press