Computers, Materials & Continua
DOI: 10.32604/cmc.2022.028942
Article

Tech Science Press

A Novel Metaheuristic Algorithm: The Team Competition and Cooperation Optimization Algorithm

Tao Wu[1], Xinyu Wu[1], Jingjue Chen[1], Xi Chen[2,*] and Amir Homayoon Ashrafzadeh[3]

[1]School of Computer Science, Chengdu University of Information Technology, Chengdu, 610225, China
[2]School of Computer Science and Engineering, Southwest Minzu University, Chengdu, 610041, China
[3]CSIT Department, School of Science, RMIT University, Melbourne, 3058, Australia
*Corresponding Author: Xi Chen. Email: cx@swun.edu.cn
Received: 21 February 2022; Accepted: 07 April 2022

Abstract: Metaheuristic algorithm is a generalization of heuristic algorithm that can be applied to almost all optimization problems. For optimization problems, metaheuristic algorithm is one of the methods to find its optimal solution or approximate solution under limited conditions. Most of the existing metaheuristic algorithms are designed for serial systems. Meanwhile, existing algorithms still have a lot of room for improvement in convergence speed, robustness, and performance. To address these issues, this paper proposes an easily parallelizable metaheuristic optimization algorithm called team competition and cooperation optimization (TCCO) inspired by the process of human team cooperation and competition. The proposed algorithm attempts to mathematically model human team cooperation and competition to promote the optimization process and find an approximate solution as close as possible to the optimal solution under limited conditions. In order to evaluate the performance of the proposed algorithm, this paper compares the solution accuracy and convergence speed of the TCCO algorithm with the Grasshopper Optimization Algorithm (GOA), Seagull Optimization Algorithm (SOA), Whale Optimization Algorithm (WOA) and Sparrow Search Algorithm (SSA). Experiment results of 30 test functions commonly used in the optimization field indicate that, compared with these current advanced metaheuristic algorithms, TCCO has strong competitiveness in both solution accuracy and convergence speed.

Keywords: Optimization; metaheuristic; algorithm

1 Introduction

Metaheuristic algorithm combines the advantages of random search algorithm and local search algorithm. Compared with the optimization algorithm that gives a clear optimal solution, the metaheuristic algorithm gives an optimal solution or approximate solution to the optimization problem under limited conditions. In recent years, traditional optimization algorithms can hardly meet the accuracy requirements of various fields such as engineering, business and economics for optimization problems under limited conditions [1]. Compared with traditional optimization algorithms, metaheuristic algorithms have the following advantages. First, the algorithm is simple and easy to implement [2,3]; second, it requires less time and space, and can be adjusted according to the user's accuracy requirements [4]; third, the algorithm can jump out of the local optimal solution to a certain extent and approach the global optimal solution as close as possible. Generally, the metaheuristic optimization algorithm includes two parts, exploration and exploitation. Due to the randomness of the metaheuristic algorithm, finding a proper balance between these two parts is a challenging task [5].

The No Free Lunch theorem (NFL) [6] shows that there is no metaheuristic algorithm that can solve all optimization problems at the same time. The NFL theorem makes this field of study highly active. It allows researchers to improve existing algorithms or propose new algorithms to better address various optimization problems. In this paper, the experiments show that compared with the latest excellent optimization algorithms, such as Whale Optimization Algorithm (WOA) [7], Sparrow Search Algorithm (SSA) [8], Seagull Optimization Algorithm (SOA) [9], Grasshopper Optimization Algorithm (GOA) [10], TCCO has made significant progress.

The rest of the paper is structured as follows. Section 2 presents a literature review of metaheuristic algorithms. Section 3 presents the details about TCCO and its pseudo-code implementation. Section 4 provides the comparative statistical analysis of results on benchmark functions. Section 5 concludes the work and suggests some directions for future studies.

2 Related Work

In the past few decades, researchers have developed a series of metaheuristic algorithms inspired by nature to solve optimization problems under limited conditions. They can be roughly divided into the following four classes:

The metaheuristic algorithm based on Darwinian theory of evolution. For instance, the Genetic Algorithm (GA) [11] proposed by Holland et al. seeks the optimal solution by imitating the natural selection, genetic and mutation mechanisms in the biological evolution; the Biogeography-Based Optimizer proposed by Simon (BBO) [12], which is inspired by biogeography regarding the migration of species between different habitats, as well as the evolution and extinction of species. These algorithms have been widely used in process control, signal processing, image processing, flexible job shop scheduling, machine learning and other fields. Excellent traits have a greater probability of being inherited, which is one of the main advantages of evolutionary algorithms. Besides, the algorithms are scalable and easy to combine with other algorithms. The disadvantage is that they may fall into a local optimal solution and lead to premature convergence.

The second class is physics-based algorithms. This type of algorithm seeks the optimal solution by imitating the physical rules that are common in the real world. For example, the annealing idea proposed by Metropolis et al. was introduced into the field of optimization problems by Kirkpatrick et al. and designed the simulated annealing algorithm (SA) [13]. Although the simulated annealing algorithm has a simple calculation process, it is universal, and has strong robustness. It can be used to solve complex non-linear problems. But there are also disadvantages such as slow convergence speed, long execution time, performance related to initial values and parameter sensitivity; different from SA, the Gravitational Search Algorithm (GSA) [14] proposed by Esmat et al. is an optimization algorithm based on the law of universal gravitation. It finds the optimal solution by moving the particle position of the population; furthermore, the artificial electric field algorithm (AEFA) [15] designed by Yadav, which is inspired by the Coulomb law, finds optimal solution by simulating the movement of charged particles in an electrostatic field.

Algorithms based on swarm intelligence is the third class, they find the optimal solution by simulating the activities of biological swarms. The Particle Swarm optimization algorithm (PSO) [16] designed by Kennedy et al. It is inspired by the social behavior of a flock of birds. Individuals are abstracted into particles, and all particles have a fitness value determined by a user defined fitness function. Each particle also has a speed that determines the direction and distance of their flight, and then the particles follow the current optimal particle to search in the solution space. The ant colony optimization (ACO) [17] proposed by Dorigo et al. is inspired by the social behavior of ants. In fact, the social intelligence of ants in finding the closest path from the nest and a source of food is the main inspiration of this algorithm. The Whale Optimization Algorithm (WOA) [7] designed by Mirjalili et al., which mimics the hunting behavior of humpback whales. In the WOA algorithm, the position of each humpback whale represents a feasible solution. The sparrow search algorithm (SSA) [8] designed by Xue et al. finds the optimal solution by simulating the strategy of predation and avoiding natural enemies of sparrow groups. This kind of swarm intelligence algorithm also has problems such as easy to fall into local optimal solution and slow convergence speed. The Rock Hyraxes Swarm Optimization (RHSO) [18] designed by Al-Khateeb et al., which mimics the collective behavior of rock hyraxes to find their eating and their special way of looking at this food. RHSO is very effective in solving real issues with constraints.

The Fourth subclass is the algorithms based on human behavior, such as CAs [19] and ICA [20]. In the Cultural Algorithms (CAs) [19], the belief space serves as a knowledge database in which people's past experience is stored, so that future generations can learn from the knowledge. As a result, the evolution speed of the population surpasses the evolution speed of purely relying on biological genetic inheritance, and has good global optimization performance. The Imperialist Competitive Algorithm (ICA) [20] designed by Atashpaz et al. is an optimization method formed by simulating the colony assimilation mechanism and the imperial competition mechanism.

3 The Team Competition and Cooperation Optimization Algorithm

In this section, we will elaborate on the inspiration of TCCO, the mathematical model and pseudo-code.

3.1 Algorithm Inspiration

Competition and cooperation exist everywhere in human society. The development of human society is largely driven by competition and cooperation. Therefore, this paper attempts to mathematically model cooperation and competition to promote the optimization process and find an approximate solution as close as possible to the optimal solution under limited conditions. At the same time, the concept of team and intra-team update in algorithm naturally support parallel computation, the simple and efficient inter-team cooperation also allow the algorithm to be parallelized with a small communication cost. In addition, considering the massive computing power of the parallel system, this algorithm also designs a process of judging the advantage of team members, which improves the performance and convergence speed of the algorithm to a certain extent.

The competition and cooperation process in this paper can be simplified into the following steps. First, people are divided into several teams. In order to achieve a same goal, everyone proposes their own pre-solution to this problem. Each team selects a leader according to the quality of their solution. The cooperation is advanced under the organization of the leader, trying to find a better solution.

After all the team members updated their own solution, a new leader is selected, and then the best solution is used to participate in the team competition. The team with the best solution wins and becomes the dominant team. Next, each team randomly finds their own partners to work together to optimize their solution. However, the dominant team has the right to choose more partners than other teams. If your team's solution is better than partner's, they will follow you, otherwise thing reversed. After the cooperation, people of each team should re-elect the leader and participate in the team competition. When all the teams have completed these steps, the current round of competition and cooperation process ends. After multiple rounds of this process, the solution given by the dominant team will become the final solution.

3.2 Mathematical Model and Pseudo-Code Implementation

In order to simplify the mathematical model, this paper is based on the following assumptions:

Assumption 1: Each member is identical except for his own solution.

Assumption 2: Each ordinary team's partner count is partnerNorm, but the dominant team is partnerBest.

Assumption 3: The population is equally divided into teamN teams (teamN $>$ 1) and satisfies (teamN $-$ partnerBest $-$ 1) mod (partnerNorm $+$ 1) $=$ 0. Besides, the number of members in each team is bigger than 1(memberN $>$ 1).

Assumption 4: Pre-solution and random grouping are implemented by random initialization.

Assumption 5: The solution is a feasible solution of the objective function, and its quality is measured by the fitness function. This paper defines that the solution with lower fitness value is better.

Assumption 6: When the solution proposed by ordinary members and leaders, ordinary team and dominant team have the same fitness values, the solution of the leader and dominant team is given priority.

Assumption 7: The team leader's solution is defined as the team's solution.

Assumption 8: The process of advantages judgment is simplified to the member replaces the corresponding items of others with their own items in turn. Items are defined as advantages only if the solution is better after be replaced.

In this paper, partnerNorm $=$ 1, partnerBest $=$ 2, the number of team is set to 7 (teamN $=$ 7), the number of team members is also 7 (memberN $=$ 7) and the population size is 49. Fig. 1 shows the basic flow chart of TCCO.

3.2.1 Grouping and Pre-Solution

There are 7 teams in this paper, and each team contains 7 members. It can be seen from assumptions 1 and 5 that members can be abstracted as feasible solutions, and the members of a team can be specified by a matrix. Each row of the matrix identifies a member. See Eq. (1) for details, where d represents the dimension of the objective function.

$$M_n = \begin{bmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,d} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{memberN,1} & m_{memberN,2} & \cdots & m_{memberN,d} \end{bmatrix} (n = 0, 1, \ldots, teamN - 1) \tag{1}$$

$$m_i = \text{borderL} + \text{uniform}(0, 1) * (\text{borderH} - \text{borderL}) \tag{2}$$

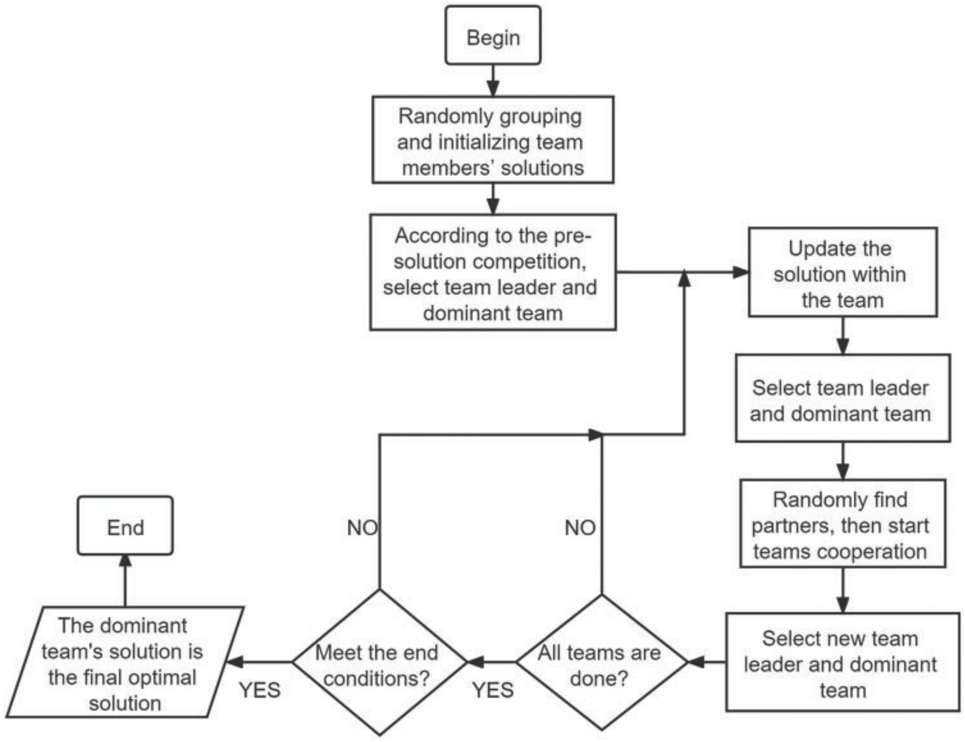
Begin
Randomly grouping and initializing team members' solutions
According to the pre-solution competition, select team leader and dominant team
Update the solution within the team
Select team leader and dominant team
Randomly find partners, then start teams cooperation
Select new team leader and dominant team
All teams are done?
NO
YES
Meet the end conditions?
NO
YES
The dominant team's solution is the final optimal solution
End

Figure 1: TCCO algorithm flow chart

Eq. (2) gives the initialization method of each pre-solution, where m_i represents a feasible solution, borderL and borderH are lower and upper bounds of objective function. The uniform (0, 1) is a uniformly distributed random d-dimension vector in the range [0, 1].

3.2.2 Fitness Function and Competition

Generally, under the condition of assumption 5, if the objective function f(x) is a maximization problem and its value range is non-negative, then the fitness function $\mathrm{Fit}(x) = \frac{1}{f(x)}$. If f(x) is a minimization problem, it can be simply mapped to $\mathrm{Fit}(x) = f(x)$.

$$\mathrm{Fit}(x) = f(x) \tag{3}$$

$$\mathrm{Fit}(M_{n)} = \begin{bmatrix} f\left(m_{1,1},\ m_{1,2},\ldots,m_{1,d}\right) \\ f\left(m_{2,1},\ m_{2,2},\ldots,m_{2,d}\right) \\ \vdots \\ f\left(m_{memberN,1}, m_{memberN,2},\ldots,m_{memberN,d}\right) \end{bmatrix} \tag{4}$$

$$\mathrm{leader}_n = M_n\left[argmin\left(Fit\left(M_n\right)\right)\right] \tag{5}$$

$$\mathrm{leaderV} = \mathrm{Fit}\left(\left[\mathrm{leader}_1, \mathrm{leader}_2, \ldots, \mathrm{leader}_n\right]^T\right) \tag{6}$$

$$\mathrm{leaderTeam} = M_{argmin(\mathrm{leaderV})} \tag{7}$$

Eq. (5) gives the selection method of the team leader, where leader$_n$ represents the solution of the team leader, that also is, the solution of the team. $argmin(x)$ represents the index of the minimum value of vector x, and the leader of each team follows the assumption 6 responsible by the best solution within the contemporary team. Eq. (7) gives the selection method of the leading team, and the dominant team is also defined as the team with the best solution among all teams according to Assumption 6. The selection of the team leader and the dominant team together constitutes the competition in this algorithm.

3.2.3 Update Solutions within the Team

For ordinary members of the team, they have three ways to update their solution. In this paper, there is a probability $P_{leader} = 0.6$ that members will follow the team leader. Members in one team can share their solutions without reservation, so that members can learn from leader to improve their own shortcomings, at the same time, the leader also can absorb advantages of their members; besides, there is a probability $P_{\text{dominant}Team} = 0.3$ for all members to follow the leader of the dominant team, in order to surpass the leader of their own team. But because they are not in the same team, they can only learn and improve from the leader of the dominant team in the general direction, and they cannot communicate with each other; furthermore, there is also a probability of $1 - P_{leader} - P_{leaderTeam} = 0.1$, members can choose not to follow and freely explore their own solutions. In general, the free exploration probability can affect the exploration and exploitation capabilities of the algorithm. The greater the probability, the stronger the exploration ability of the algorithm. *errorRange* is the following error, which is positively related to the distance between itself and the following target, and is limited by the number of current iteration. As the number of iteration increases, it gradually approaches zero. The expression is given by Eq. (8), where abs is the absolute value function, the tMax is the maximum iteration and the t is current iteration.

$$\text{errorRange} = \text{uniform}(0,1) * \frac{(\text{tMax} - \text{t})}{tMax} * \text{abs}(m_i - \text{dest}) \tag{8}$$

$$\text{exploreRange} = \text{uniform}(0,1) * \frac{(borderH - borderL) * d * (\text{tMax} - \text{t})}{teamN * memberN * tMax} \tag{9}$$

1. Follow the team leader, update method is given in Eq. (15). T$_i$ in Eq. (10) is a real symmetric matrix, the diagonal elements are the elements of the member to be updated, and the remaining elements are the same as the team leader. The weakness$_i$ vector given in Eq. (11) represents the disadvantage items of the member compared to the leader, and the strength$_i$ given in Eq. (12) represents the advantage items. The errorRange and exploreRange are given by Eqs. (8) and (9), in this case, dest is the team leader, and p is a random number in the interval [0, 1] that obeys a uniform distribution.

$$\text{T}_i = \begin{bmatrix} m_{i,1} & m_{leader,2} & \cdots & m_{leader,d} \\ m_{leader,1} & m_{i,2} & \cdots & m_{leader,d} \\ \vdots & \vdots & \ddots & \vdots \\ m_{leader,1} & m_{leader,2} & \cdots & m_{i,d} \end{bmatrix} \tag{10}$$

$$\text{weakness}_i = \text{Fit}(\text{T}_i) > fit_{leader} \tag{11}$$

$$\text{strength}_i = \text{Fit}(\text{T}_i) \le fit_{leader} \tag{12}$$

$$m_i^{t+1}[\mathrm{dim}] = \begin{cases} leader_n^t[dim] + errorRange, & p < 0.5 \\ leader_n^t[dim] - errorRange, & p \geq 0.5 \end{cases} \tag{13}$$

$$m_i^{t+1}[\mathrm{dim}] = \begin{cases} m_i^t[dim] + exploreRange, & p < 0.5 \\ m_i^t[dim] - exploreRange, & p \geq 0.5 \end{cases} \tag{14}$$

$$m_i^{t+1} = \begin{cases} (13), & \mathrm{dim}\ \ in\ wekness_i \\ (14), & \mathrm{dim}\ \ in\ strength_i \end{cases} \tag{15}$$

2. Follow the leader of the dominant team, the *errorRange* is given by Eq. (8). In this case, dest is dominant team's leader, and p is a random number in the interval [0, 1] that obeys a uniform distribution.

$$m_i^{t+1} = \begin{cases} leaderTeam_n^t + uniform(0,1) * errorRange, & p < 0.5 \\ leaderTeam_n^t - uniform(0,1) * errorRange, & p \geq 0.5 \end{cases} \tag{16}$$

3. Same update method as Eq. (2) for free exploration. $m_i^{t+1} = \mathrm{borderL} + \mathrm{uniform}(0,\ 1) * (\mathrm{borderH} - \mathrm{borderL})$

For the team leader, he will learn the advantages of all other members, and Eq. (17) gives the update method. In a word, it is to gather the advantages of his team members.

$$leader_n^{t+1}[\mathrm{dim}] = \mathrm{M}_n[argmin(Fit([T_1[dim], T_2[dim], \ldots, T_{memberN}[dim]))][dim] \tag{17}$$

In particular, for the first and second update methods of the above-mentioned ordinary members, when the feasible solution exceeds the range of the solution space, this paper simply adopts the method of directly taking the boundary value. After a round of update solutions are completed, a new team leader will be selected according to Assumption 6 and Eq. (5), and a new dominant team will be selected according to Assumption 6 and Eq. (7).

3.2.4 Teamwork to Update Solutions

After all members updated, the team randomly search for partners according to Assumption 2. In the two teams that cooperate, all members will follow the better solution. The way to update the solution is given by Eq. (18). Step is the number of times that each team continuously proposes better solutions in all collaborations. In other words, once the team solution is worse than cooperation team, step will be reset to 1. The errorRange is given by Eq. (8), dest is the solution of the better team in this case, and p is a random number with uniform distribution in the interval [0, 1].

$$m_i = \begin{cases} dest + step * errorRange, & p < 0.5 \\ dest - step * errorRange, & p \geq 0.5 \end{cases} \tag{18}$$

When the feasible solution exceeds the solution space, the method of directly taking the boundary value is also simply adopted in this paper. After a round of collaborative update solutions between teams, a new team leader will be selected according to Assumption 6 and Eq. (5), and a new dominant team will be selected according to Assumption 6 and Eq. (7).

3.2.5 TCCO Pseudo-Code Implementation

Tab. 1 shows the team competition and cooperation optimization algorithm's main pseudo-code.

Table 1: TCCO pseudo-code

```
Algorithm
01: Begin
02:     Randomly grouping and initializing team members' solutions.
03:     Select team leader and dominant team.
04:     While (stop condition is not met):
05:         For (team in teams):
06:             Team members update solution by itself.
07:             Select new team leader.
08:             Select new dominant team.
09:             Randomly find partners.
10:             Start team cooperation.
11:             Select new team leader.
12:             Select new dominant team.
13:         End
14:     End
15:     The dominant team's solution is the final optimal solution.
16: End
```

4 Experiment and Analysis

In this section, the TCCO is benchmarked on 30 classic benchmark functions [1,21,22] compared with the Whale Optimization Algorithm (WOA), Sparrow Search Algorithm (SSA), Seagull Optimization Algorithm (SOA) and Grasshopper Optimization Algorithm (GOA) to analysis it's performance. All algorithms are basic version, and the parameter settings are shown in Tab. 2. In this paper, 30 classic benchmark functions are divided into 4 categories: low-dimensional unimodal, low-dimensional multimodal, high-dimensional unimodal, and high-dimensional multimodal according to dimensions and modals. Four sets of experiments are performed to test and compare the performance of the above algorithms. In order to compare the performance fairly, the population size of all algorithms is set to 49, and the maximum number of iterations is 500. At the same time, in order to make the comparison experiment more general, each algorithm will run the benchmark function 30 times to compare the average, best, and worst solutions of 30 classic benchmark functions.

4.1 The Experiments of Low-Dimensional Unimodal Functions

This section will conduct experiments on the first set of eight low-dimensional unimodal test functions to compare the performance of the above five optimization algorithms. The name, expression, minimum value, range and dimension of the functions are shown in Tab. 3.

Table 2: Algorithm parameter settings

Parameters\ Algorithm	GOA	SOA	WOA	SSA	TCCO
l	1.5				
f	0.5				
C_{max}	1				
C_{min}	0.00004				
fc		2			
a			2		
b			1		
$P_{percent}$				0.2	
R2				0.8	
P_{leader}					0.6
$P_{leaderTeam}$					0.3
partnerNorm					1
partnerBest					2

Table 3: Low-dimensional unimodal test functions

Function name	Expression	Minimum	Range	d
Mccormick	$F_1 = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1$	−1.9133	[−3, 4]	2
Easom	$F_2 = -\cos(x_1) * \cos(x_2) * e^{-(x_1-\pi)^2-(x_2-\pi)^2}$	−1	[−100, 100]	2
Matyas	$F_3 = 0.26\left(x_1^2 + x_2^2\right) - 0.48x_1x_2$	0	[−10, 10]	2
Zakharov	$F_4 = \sum_{i=1}^{d} x_i^2 + \left(\sum_{i=1}^{d} 0.5i * x_i\right)^2 + \left(\sum_{i=1}^{d} 0.5i * x_i\right)^4$	0	[−5, 10]	10
Bohachevsky1	$F_5 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	0	[−100, 100]	2
Booth	$F_6 = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$	0	[−10, 10]	2
Bohachevsky2	$F_7 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) * \cos(4\pi x_2) + 0.3$	0	[−100, 100]	2
Bohachevsky3	$F_8 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1 + 4\pi x_2) + 0.3$	0	[−100, 100]	2

Tab. 4 shows the experiment results of 8 low-dimensional unimodal test functions and advantages are shown in bold. In the average solution comparison, SOA won 3 items, WOA won 4 items and TCCO won 6 items. In the comparison of minimum values, GOA got 1 item, SOA got 2 items, WOA got 3 items and SSA got 4 items, while TCCO has 7 wins. In the maximum value item, SOA won 3 items, WOA won 4 items and TCCO won 6 items. Fig. 2 shows the convergence performance of each algorithm under eight test functions (run once, maximum iteration set to 20). The results show that TCCO has a slight lead in convergence speed and accuracy.

Table 4: Experiment results of low-dimensional unimodal test functions

		GOA	SOA	WOA	SSA	TCCO
F1	Best	**−1.913E+00**	−1.913E+00	−1.913E+00	−1.913E+00	**−1.913E+00**
	Worst	−1.913E+00	−1.721E+00	−1.913E+00	−1.913E+00	**−1.913E+00**
	Mean	−1.913E+00	−1.884E+00	−1.913E+00	−1.913E+00	**−1.913E+00**
F2	Best	−1.000E+00	−1.000E+00	−1.000E+00	−1.000E+00	**−1.000E+00**
	Worst	−1.000E+00	−8.088E−05	−1.000E+00	−1.000E+00	**−1.000E+00**
	Mean	−1.000E+00	−8.310E−01	−1.000E+00	−1.000E+00	**−1.000E+00**
F3	Best	1.193E−14	0.000E+00	3.527E−213	0.000E+00	**−1.000E+00**
	Worst	1.426E−13	**0.000E+00**	1.012E−174	2.263E−18	9.011E−47
	Mean	5.726E−14	**0.000E+00**	3.374E−176	1.373E−19	3.932E−48
F4	Best	1.194E−05	7.815E−02	1.093E−26	**0.000E+00**	1.162E−15
	Worst	3.023E−05	1.709E+02	**7.918E−20**	2.493E−16	4.030E−12
	Mean	2.011E−05	3.405E+01	**3.685E−21**	9.677E−18	3.629E−13
F5	Best	3.064E−10	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
	Worst	2.148E−09	**0.000E+00**	**0.000E+00**	4.223E−11	**0.000E+00**
	Mean	9.757E−10	**0.000E+00**	**0.000E+00**	1.408E−12	**0.000E+00**
F6	Best	9.943E−13	1.645E−06	4.489E−09	6.558E−07	**0.000E+00**
	Worst	5.328E−12	2.231E−01	5.972E−06	1.887E−04	**0.000E+00**
	Mean	2.628E−12	1.584E−02	7.115E−07	4.054E−05	**0.000E+00**
F7	Best	3.621E−10	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
	Worst	2.375E−09	**0.000E+00**	**0.000E+00**	2.731E−14	**0.000E+00**
	Mean	1.346E−09	**0.000E+00**	**0.000E+00**	9.363E−16	**0.000E+00**
F8	Best	1.303E−10	6.181E−12	**0.000E+00**	**0.000E+00**	**0.000E+00**
	Worst	3.472E−10	1.820E−06	**0.000E+00**	8.116E−14	**0.000E+00**
	Mean	2.065E−10	1.160E−07	**0.000E+00**	3.364E−15	**0.000E+00**

4.2 The Experiments of Low-Dimensional Multimodal Functions

This section will conduct experiments on the second set of 12 low-dimensional multimodal test functions to compare the performance of the above five optimization algorithms. The name, expression, minimum value, range and dimension of the function are shown in Tab. 5, and the results of experiment are shown in Tab. 6. Fig. 3 shows the convergence performance of each algorithm under twelve test functions (run once, maximum iteration set to 20).

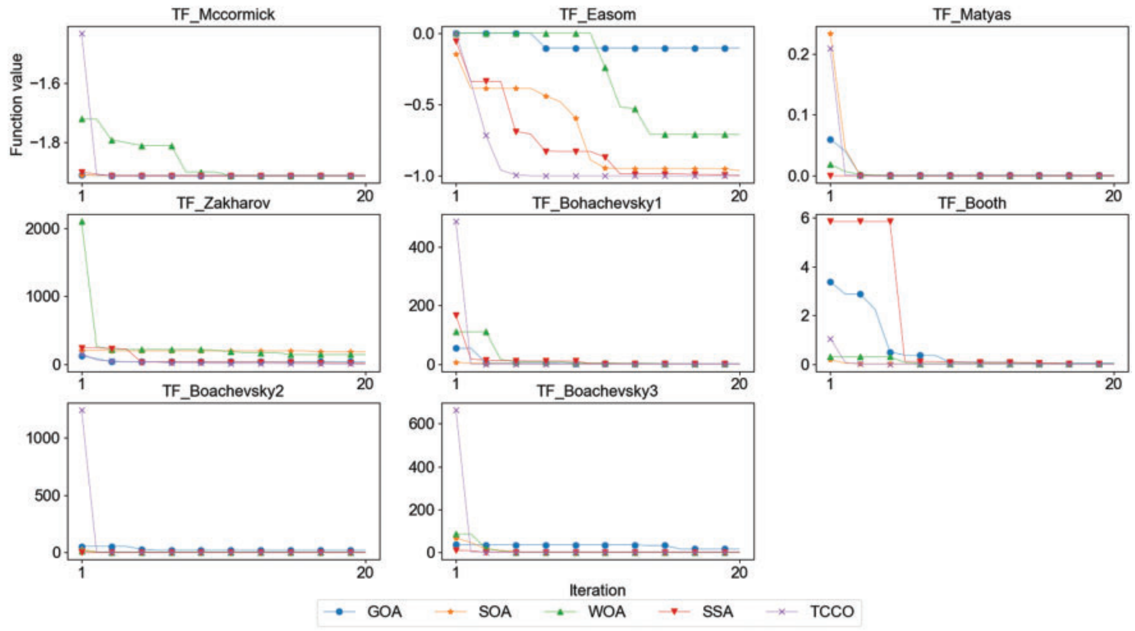

Figure 2: Convergence speed comparison of low-dimensional unimodal test functions

Table 5: Low-dimensional multimodal test functions

Function name	Expression	Minimum	Range	d
Beale	$F_9 = (1.5 - x_1 + x_1x_2)^2 + (2.25 - x_1 + x_1x_2^2)^2 + (2.625 - x_1 + x_1x_2^3)^2$	0	[−4.5, 4.5]	2
Michalewicz2	$F_{10} = -\sum_{j=1}^{d} \sin(x_j)\left(\sin(jx_j^2/\pi)\right)^{20}$	−1.8013	[0, π]	2
Michalewicz5	$F_{11} = -\sum_{j=1}^{d} \sin(x_j)\left(\sin(jx_j^2/\pi)\right)^{20}$	−4.6877	[0, π]	5
Michalewicz10	$F_{12} = -\sum_{j=1}^{d} \sin(x_j)\left(\sin(jx_j^2/\pi)\right)^{20}$	−9.6602	[0, π]	10
Schaffer	$F_{13} = 0.5 + \frac{sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left(1 + 0.001\left(x_1^2 + x_2^2\right)\right)^2}$	0	[−100, 100]	2
Six_Hump_ Camel_Back	$F_{14} = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	−1.03163	[−5, 5]	2
Shubert	$F_{15} = \left(\sum_{j=1}^{5} jcos(j+1)x_1 + j\right) * \left(\sum_{j=1}^{5} jcos((j+1)x_2 + j)\right)$	−186.73	[−10, 10]	2

(Continued)

Table 5: Continued

Function name	Expression	Minimum	Range	d
Cross_in_tray	$F_{16} = -0.0001\left(\left\lvert \sin(x_1)\sin(x_2)\, e^{\left\lvert 100 - \frac{\sqrt{x_1^2+x_2^2}}{\pi}\right\rvert}\right\rvert + 1\right)^{0.1}$	−2.06261	[−10, 10]	2
Drop_Wave	$F_{17} = -\frac{1+\cos\left(12\sqrt{x_1^2+x_2^2}\right)}{0.5\left(x_1^2+x_2^2\right)+2}$	−1	[−5.12, 5.12]	2
Eggholder	$F_{18} = -(x_2+47)\sin\left(\sqrt{\left\lvert x_2+\frac{x_1}{2}+47\right\rvert}\right) - x_1 sin\left(\sqrt{\lvert x_1-(x_2+47)\rvert}\right)$	−959.647	[−512, 512]	2
Goldstein_Price	$F_{19} = [1+(x_1+x_2+1)^2 (19-14x_1+3x_1^2-14x_2+ 6x_1x_2+3x_2^2)] * [30+(2x_1-3x_2)^2 (18-32x_1+12x_1^2+48x_2-36x_1x_2+ 27x_2^2)]$	3	[−2, 2]	2
Colville	$F_{20} = 100\left(x_1^2-x_2\right)^2+(x_1-1)^2+(x_3-1)^2 + 90\left(x_3^2-x_4\right)^2+10.1(x_2-1)^2+(x_4-1)^2 + 19.8\,(x_2-1)\,(x_4-1)$	0	[−10, 10]	4

Table 6: Experiment results of low-dimensional multimodal test functions

		GOA	SOA	WOA	SSA	TCCO
F9	Best	8.096E−14	1.589E−10	2.811E−10	1.145E−08	**0.000E+00**
	Worst	7.621E−01	7.621E−01	7.990E−07	2.611E−06	**2.773E−32**
	Mean	5.080E−01	2.286E−01	1.413E−07	5.153E−07	**1.849E−33**
F10	Best	−1.801E+00	−1.801E+00	−1.801E+00	−1.801E+00	**−1.801E+00**
	Worst	−1.000E+00	−1.000E+00	−1.313E+00	−1.801E+00	**−1.801E+00**
	Mean	−1.534E+00	−1.668E+00	−1.730E+00	−1.801E+00	**−1.801E+00**
F11	Best	−3.550E+00	−4.585E+00	−3.504E+00	−4.631E+00	**−4.688E+00**
	Worst	−2.658E+00	−2.514E+00	−1.896E+00	−3.462E+00	**−4.688E+00**
	Mean	−3.141E+00	−3.303E+00	−2.548E+00	−4.233E+00	**−4.688E+00**
F12	Best	−6.490E+00	−5.936E+00	−4.304E+00	−8.046E+00	**−9.660E+00**
	Worst	−4.811E+00	−4.200E+00	−2.803E+00	−5.842E+00	**−9.660E+00**
	Mean	−5.898E+00	−5.109E+00	−3.602E+00	−6.670E+00	**−9.660E+00**
F13	Best	2.491E−12	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
	Worst	8.427E−11	**0.000E+00**	**0.000E+00**	2.495E−13	9.716E−03
	Mean	3.148E−11	0.000E+00	0.000E+00	8.330E−15	7.449E−03
F14	Best	−1.032E+00	−1.032E+00	−1.032E+00	−1.032E+00	**−1.032E+00**
	Worst	−1.032E+00	−1.032E+00	−1.032E+00	−1.032E+00	**−1.032E+00**
	Mean	−1.032E+00	−1.032E+00	−1.032E+00	−1.032E+00	**−1.032E+00**
F15	Best	−1.867E+02	−1.867E+02	−1.867E+02	−1.867E+02	**−1.867E+02**

(Continued)

Table 6: Continued

		GOA	SOA	WOA	SSA	TCCO
	Worst	−1.867E+02	−1.864E+02	−1.864E+02	−1.867E+02	**−1.867E+02**
	Mean	−1.867E+02	−1.867E+02	−1.867E+02	−1.867E+02	**−1.867E+02**
F16	Best	−2.063E+00	−2.063E+00	−2.063E+00	**−2.063E+00**	**−2.063E+00**
	Worst	−2.063E+00	−2.063E+00	−2.063E+00	−2.063E+00	**−2.063E+00**
	Mean	−2.063E+00	−2.063E+00	−2.063E+00	−2.063E+00	**−2.063E+00**
F17	Best	−1.000E+00	**−1.000E+00**	**−1.000E+00**	**−1.000E+00**	**−1.000E+00**
	Worst	−9.362E−01	**−1.000E+00**	**−1.000E+00**	−1.000E+00	−9.362E−01
	Mean	−9.575E−01	**−1.000E+00**	**−1.000E+00**	−1.000E+00	−9.957E−01
F18	Best	−7.865E+02	−9.596E+02	−9.596E+02	−9.596E+02	**−9.596E+02**
	Worst	−7.182E+02	−8.889E+02	−9.595E+02	**−7.865E+02**	−8.889E+02
	Mean	−7.410E+02	−9.181E+02	−9.596E+02	**−9.480E+02**	−9.554E+02
F19	Best	3.000E+00	3.000E+00	3.000E+00	3.000E+00	**3.000E+00**
	Worst	3.000E+00	3.271E+01	3.000E+00	3.000E+00	**3.000E+00**
	Mean	3.000E+00	9.479E+00	3.000E+00	3.000E+00	**3.000E+00**
F20	Best	3.000E+00	3.000E+00	3.000E+00	3.000E+00	**3.000E+00**
	Worst	3.000E+00	3.255E+01	3.000E+00	3.000E+00	**3.000E+00**
	Mean	3.000E+00	9.467E+00	3.000E+00	3.000E+00	**3.000E+00**

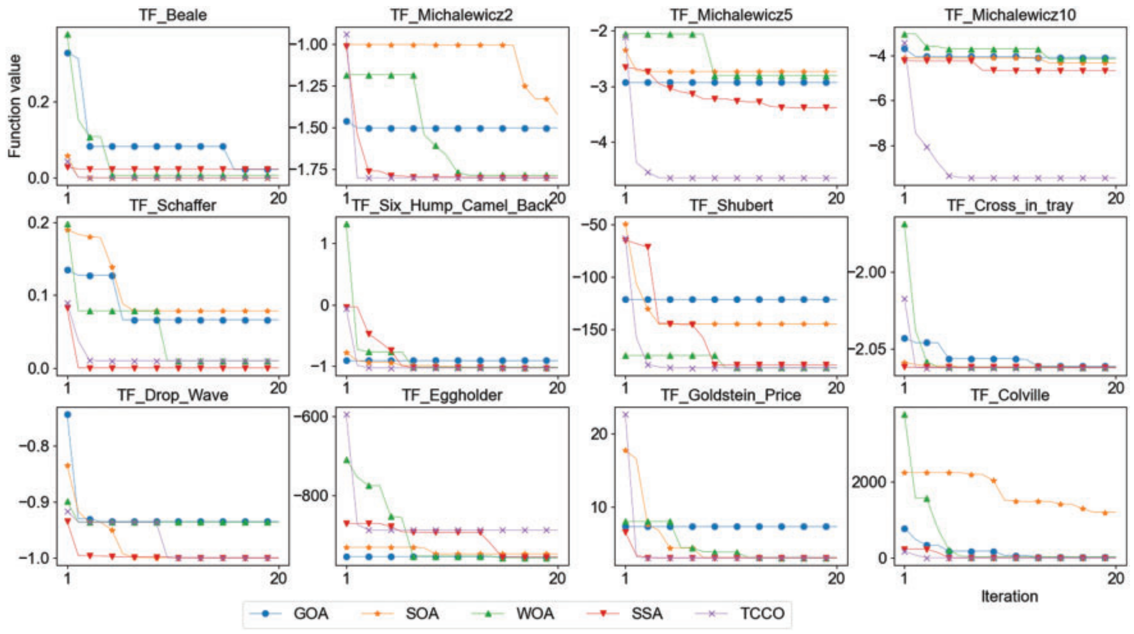

Figure 3: Convergence speed comparison of low-dimensional multimodal test functions

4.3 The Experiments of High-Dimensional Unimodal Functions

This section will conduct experiments on the third group of 7 high-dimensional unimodal test functions to compare the performance of the above 5 optimization algorithms. The name, expression, minimum value, range and dimension of the function are shown in Tab. 7; the results of experiment are shown in Tab. 8. Fig. 4 shows the convergence performance of each algorithm under seven test functions (run once, maximum iteration set to 20).

Table 7: High-dimensional unimodal test functions

Function name	Expression	Minimum	Range	d
Step	$F_{21} = \sum_{j=1}^{d} (x_j + 0.5)^2$	0	[−5.12, 5.12]	30
Trid	$F_{22} = \sum_{j=1}^{d} (x_j - 1)^2 - \sum_{j=2}^{d} x_j x_{j-1}$	−4930	[−900, 900]	30
Quartic	$F_{23} = \sum_{j=1}^{d} j * x_j^4 + rand$	0	[−1.28, 1.28]	30
Schwefel2_22	$F_{24} = \sum_{j=1}^{d} \lvert x_j \rvert + \prod_{j=1}^{d} \lvert x_j \rvert$	0	[−10, 0]	30
Schwefel1_2	$F_{25} = \sum_{j=1}^{d} \left(\sum_{k=1}^{j} x_k \right)^2$	0	[−100, 100]	30
Rosenbrock	$F_{26} = \sum_{j=1}^{d-1} \left[100 \left(x_{j+1} - x_j^2 \right)^2 + \left(x_j - 1 \right)^2 \right]$	0	[−30, 30]	30
Dixon_Price	$F_{27} = (x_1 - 1)^2 + \sum_{j=2}^{d} j \left(2x_j^2 - x_j - 1 \right)^2$	0	[−10, 10]	30

Table 8: Experiment results of high-dimensional unimodal test functions

		GOA	SOA	WOA	SSA	TCCO
F21	Best	1.251E−04	2.565E−09	2.521E−01	6.127E−05	**0.000E+00**
	Worst	6.057E−04	7.382E−03	1.492E+00	6.969E−03	**0.000E+00**
	Mean	4.035E−04	1.941E−03	9.511E−01	2.158E−03	**0.000E+00**
F22	Best	−3.370E+03	−1.288E+03	−1.469E+03	−4.779E+03	**−4.930E+03**
	Worst	−1.914E+03	−8.707E+02	−5.811E+01	−3.747E+03	**−4.366E+03**
	Mean	−2.523E+03	−9.554E+02	−3.827E+02	−4.343E+03	**−4.742E+03**
F23	Best	4.087E−02	**3.930E−05**	2.526E−04	5.325E−05	2.907E−03
	Worst	9.642E−02	7.706E−03	**1.851E−02**	1.839E−03	4.602E−01
	Mean	6.544E−02	2.352E−03	**3.756E−03**	6.062E−04	1.407E−01
F24	Best	2.428E+00	**1.350E−142**	1.276E−17	1.289E−81	7.975E−40
	Worst	3.816E+00	**3.829E−124**	1.102E−14	3.456E−07	2.270E−36

(Continued)

Table 8: Continued

		GOA	SOA	WOA	SSA	TCCO
	Mean	3.282E+00	**1.317E−125**	9.780E−16	1.928E−08	3.739E−37
F25	Best	6.035E+02	9.671E+03	**6.054E−07**	0.000E+00	4.318E+00
	Worst	2.631E+03	7.689E+04	**3.109E−02**	1.137E−13	6.298E+01
	Mean	1.680E+03	3.966E+04	**2.819E−03**	4.532E−15	2.167E+01
F26	Best	1.913E+02	**2.795E−09**	2.531E+01	3.266E−08	1.262E−02
	Worst	3.933E+02	1.588E+00	**2.877E+01**	9.306E−01	2.147E+01
	Mean	2.694E+02	2.417E−01	**2.717E+01**	1.419E−01	1.069E+01
F27	Best	4.807E+00	7.686E−07	2.376E+01	1.278E−01	**0.000E+00**
	Worst	8.006E+00	1.484E+00	1.942E+02	5.166E+01	**0.000E+00**
	Mean	6.620E+00	1.477E−01	1.239E+02	1.220E+01	**0.000E+00**

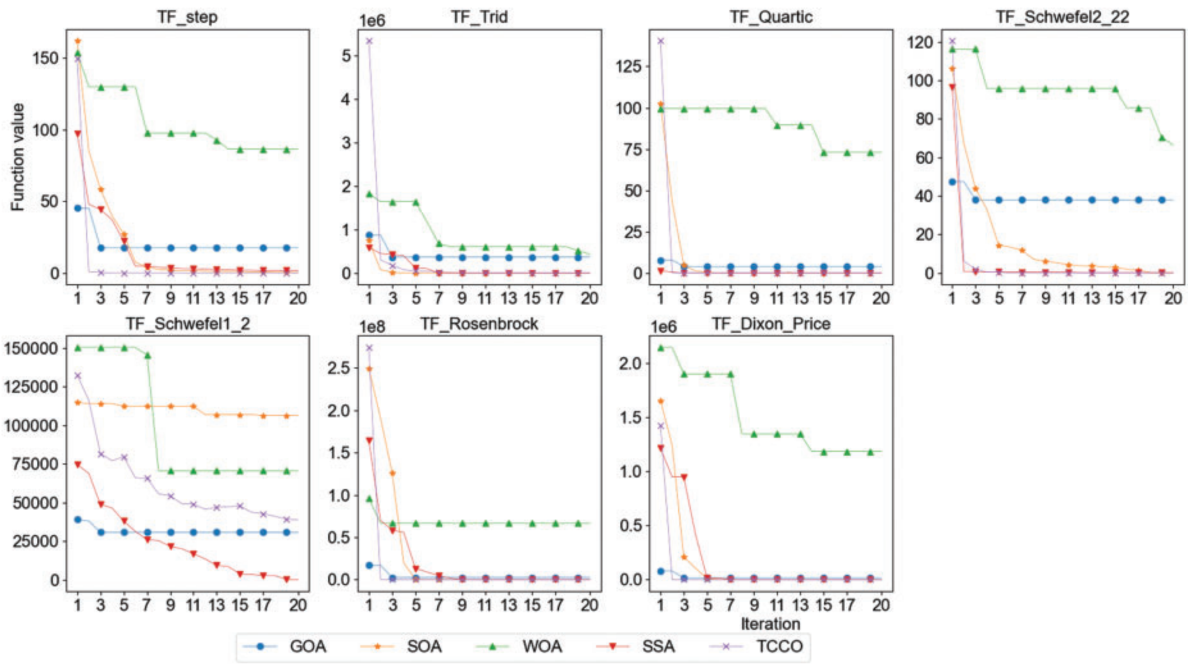

Figure 4: Convergence speed comparison of high-dimensional unimodal test functions

4.4 The Experiments of High-Dimensional Multimodal Functions

This section will conduct experiments on the fourth group of three high-dimensional multimodal test functions to compare the performance of the above five optimization algorithms. The name, expression, minimum value, range and dimension of the function are shown in Tab. 9; the results of experiment are shown in Tab. 10. Fig. 5 shows the convergence performance of each algorithm under three test functions (run once, maximum iteration set to 20).

Table 9: High-dimensional multimodal test functions

Function name	Expression	Minimum	Range	d
Rastrigin	$F_{28} = \sum_{j=1}^{d} \left(x_j^2 - 10\cos\left(2\pi x_j\right) + 10\right)$	0	[−5.12, 5.12]	30
Griewank	$F_{29} = \frac{1}{4000}\left(\sum_{j=1}^{d}\left(x_j - 100\right)^2\right) - \left(\prod_{j=1}^{d}\cos\left(\frac{x_j - 100}{\sqrt{j}}\right)\right) + 1$	0	[−600, 600]	30
Ackley	$F_{30} = -20e^{-0.2\sqrt{\frac{1}{d}\sum_{j=1}^{d}x_j^2}} - e^{\frac{1}{d}\sum_{j=1}^{d}\cos\left(2\pi x_j\right)} + 20 + e$	0	[−32, 32]	30

Table 10: Experiment results of high-dimensional multimodal test functions

		GOA	SOA	WOA	SSA	TCCO
F28	Best	4.688E+01	**0.000E+00**	**0.000E+00**	**0.000E+00**	**0.000E+00**
	Worst	1.214E+02	**0.000E+00**	5.684E−14	1.124E−08	**0.000E+00**
	Mean	8.530E+01	**0.000E+00**	1.895E−15	3.772E−10	**0.000E+00**
F29	Best	7.120E−01	8.424E−03	3.609E+00	1.346E−02	**0.000E+00**
	Worst	1.029E+00	1.029E+00	2.341E+01	1.017E+00	**8.303E−02**
	Mean	8.947E−01	5.336E−01	1.150E+01	8.238E−01	**1.038E−02**
F30	Best	2.748E+00	**4.441E−16**	3.952E−14	**4.441E−16**	3.997E−15
	Worst	3.582E+00	**4.441E−16**	1.611E−09	4.682E−07	1.465E−14
	Mean	3.227E+00	**4.441E−16**	5.533E−11	3.487E−08	7.550E−15

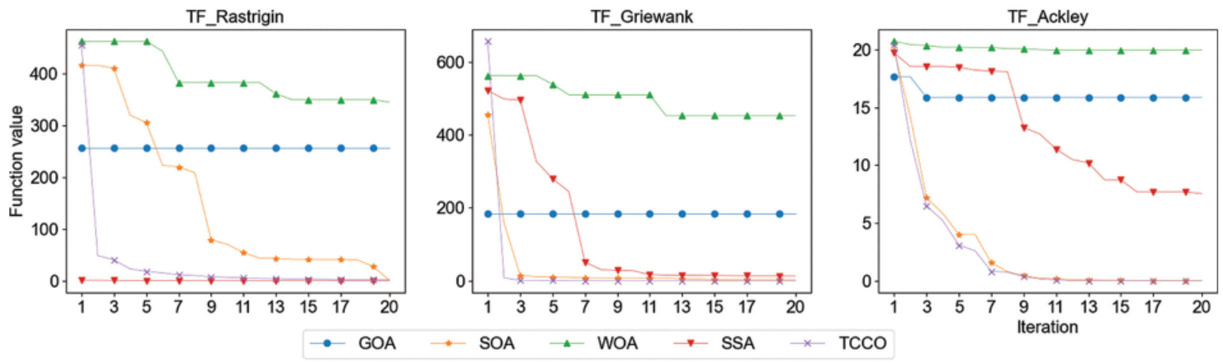

Figure 5: Convergence speed comparison of high-dimensional multimodal test functions

4.5 Analysis of Results

Unimodal function has only one global optimal solution in the solution space. This type of function can well evaluate the exploitation ability of metaheuristic algorithm. Experiment 1 and experiment 3 show that whether it is a low-dimensional unimodal function or a high-dimensional unimodal function, TCCO has good performance compared with other algorithms, and the convergence speed is faster than other algorithms.

Different from unimodal function, multimodal function has many local optimal solutions in the solution space. The number of local optimal solutions increases exponentially with the growth of the problem scale. Therefore, this type of test function is often used to evaluate the exploration capability of metaheuristic algorithm. Experiment 2 and 4 are designed for this purpose. The experiment results show that the performance of TCCO is better than other algorithms in both low-dimensional multimodal function and high-dimensional multimodal function.

5 Conclusion

This research is inspired by the competition and cooperation between human teams, and proposes a new metaheuristic optimization algorithm, Team Competition and Cooperation Optimization Algorithm (TCCO), which is easy to parallelize. TCCO includes two operators, which respectively simulate the update solution within the team and the update solution between teams. This paper conducts detailed experiments on 30 benchmark functions, compared and analyzed the exploration ability, exploitation ability and convergence speed of WOA, SSA, SOA, GOA and TCCO. Experiment results show that compared with other metaheuristic algorithms mentioned above, TCCO has stronger competitiveness.

Funding Statement: This research was partially supported by the National Key Research and Development Program of China (2018YFC1507005), Sichuan Science and Technology Program (2020YFG0189, 22ZDYF3494), China Postdoctoral Science Foundation (2018M643448), and Fundamental Research Funds for the Central Universities, Southwest Minzu University (2022101).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] M. Jain, V. Singh and A. Rani, "A novel nature-inspired algorithm for optimization: Squirrel search algorithm," *Swarm and Evolutionary Computation*, vol. 44, no. 1, pp. 148–175, 2019.
[2] E. Erdemir and A. A. Altun, "A new metaheuristic approach to solving benchmark problems: Hybrid salp swarm jaya algorithm," *Computers, Materials & Continua*, vol. 71, no. 2, pp. 2923–2941, 2022.
[3] H. Faris, I. Aljarah, M. A. Al-Betar and S. Mirjalili, "Grey wolf optimizer: A review of recent variants and applications," *Neural Computing and Applications*, vol. 30, no. 2, pp. 413–435, 2018.
[4] K. Hussain, M. N. M. Salleh, S. Cheng and Y. Shi, "Metaheuristic research: A comprehensive survey," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2191–2233, 2019.
[5] T. Dokeroglu, E. Sevinc, T. Kucukyilmaz and A. Cosar, "A survey on new generation metaheuristic algorithms," *Computers & Industrial Engineering*, vol. 137, no. 11, pp. 106040, 2019.
[6] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.
[7] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, no. 5, pp. 51–67, 2016.

[8] J. Xue and B. Shen, "A novel swarm intelligence optimization approach: Sparrow search algorithm," *Systems Science & Control Engineering*, vol. 8, no. 1, pp. 22–34, 2020.

[9] G. Dhiman and V. Kumar, "Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems," *Knowledge-Based Systems*, vol. 165, no. 3, pp. 169–196, 2019.

[10] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris and I. Aljarah, "Grasshopper optimization algorithm for multi-objective optimization problems," *Applied Intelligence*, vol. 48, no. 4, pp. 805–820, 2018.

[11] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[12] H. Ma, D. Simon, P. Siarry, Z. Yang, and M. Fei, "Biogeography-based optimization: A 10-year review," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 391–407, 2017.

[13] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[14] E. Rashed, E. Rashedi and A. Nezamabadi-pour, "A comprehensive survey on gravitational search algorithm," *Swarm and Evolutionary Computation*, vol. 41, no. 4, pp. 141–158, 2018.

[15] A. Yadav, "AEFA: Artificial electric field algorithm for global optimization," *Swarm and Evolutionary Computation*, vol. 48, no. 5, pp. 93–108, 2019.

[16] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. on Neural Networks Proc.*, Nagoya, NGO, JP, pp. 1942–1948, 1995.

[17] M. Dorigo, M. Birattari and T. Stutzle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[18] B. Al-Khateeb, K. Ahmed, M. Mahmood and D. Le, "Rock hyraxes swarm optimization: A new nature-inspired metaheuristic optimization algorithm," *Computers, Materials & Continua*, vol. 68, no. 1, pp. 643–654, 2021.

[19] A. Maheri, S. Jalili, Y. Hosseinzadeh, R. Khani and M. Miryahyavi, "A comprehensive survey on cultural algorithms," *Swarm and Evolutionary Computation*, vol. 62, no. 3, pp. 100846, 2021.

[20] G. E. Atashpaz and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *Proc. IEEE Congress on Evolutionary Computation CEC*, Singapore, SG, Singapore, pp. 661–4667, 2007.

[21] M. Jamil and X. S. Yang, "A literature survey of benchmark functions for global optimization problems," *International Journal of Mathematical Modelling and Numerical Optimisation (IJMMNO)*, vol. 4, no. 2, pp. 150–194, 2013.

[22] Y. H. Li, "A kriging-based global optimization method using multi-points infill search criterion," *Journal of Algorithms & Computational Technology*, vol. 11, no. 4, pp. 366–377, 2017.