A meta-inspired termite queen algorithm for global optimization and engineering design problems

Link to publication record in Ulster University Research Portal

Document Version
Author Accepted version

A meta-inspired termite queen algorithm for global optimization and engineering design problems

Peng Chen^a, Shihua Zhou^{a,*}, Qiang Zhang^{a,b} and Nikola Kasabov^{c,d}

^{a}*Key Laboratory of Advanced Design and Intelligent Computing, Ministry of Education, Dalian University, Dalian 116622, China*
^{b}*School of Computer Science and Technology, Dalian University of Technology, Dalian 116024, China*
^{c}*Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, Auckland 1010, New Zealand*
^{d}*Intelligent Systems Research Center, Ulster University, Londonderry BT52 1SA, UK*

ARTICLE INFO

ABSTRACT

This paper proposes a novel bio-inspired termite queen algorithm (TQA) to solve optimization problems by simulating the division of labor in termite populations under a queen's rule. TQA is benchmarked on a set of 23 functions to test its performance at solving global optimization problems, and applied to six real-world engineering design problems to verify its reliability and effectiveness. Comparative simulation studies with other algorithms are conducted, from whose results it is observed that TQA satisfactorily solves global optimization problems and engineering design problems.

1. Introduction

Optimization is common in practical problems such as scheduling (Cai et al., 2020), feature selection (Hancer, 2020), engineering design (Zhang et al., 2019), image segmentation (Elaziz et al., 2021), and DNA sequence design (Cao et al., 2020; Wu et al., 2021; Yin et al., 2021), whose traditional solutions often incur a considerable computational cost. Therefore, the metaheuristic algorithm with lower computational cost has become a hot research topic. These explore feasible domains by random search in early iterations and develop feasible solutions using local search in later iterations. The optimal solution is iteratively updated by evaluating the adaptation values of individual populations. However, according to the theorem of no free lunches (NFL), no algorithm can well solve every optimization problem (Wolpert and Macready, 1997). Therefore, it is still of great practical significance to study metaheuristic algorithms for specific problems.

Metaheuristic algorithms can be classified as either evolution-, swarm-, physics-, or human-behavior-based. Evolution-based algorithms draw on genetic knowledge that offspring inherit certain traits from their parents. Elements in two solutions are typically exchanged to produce new solutions. Representative algorithms include differential evolution (DE) (Storn and Price, 1997), evolutionary programming (EP) (Yao et al., 1999), evolutionary strategies (ESs) (Amoretti, 2014), genetic algorithms (GAs) (Holland, 1992), and genetic programming (GP) (Koza, 1990).

Swarm-based algorithms are generally inspired by animal behavior, with the most representative being particle swarm optimization (PSO) (Eberhart and Kennedy, 1995), as developed by Eberhart and Kennedy, inspired by the foraging behavior of birds and fish. New solutions are often produced through the guidance of the current optimal solution. As the largest branch of metaheuristic algorithms, these include the artificial bee colony (ABC) (Karaboga and Basturk, 2007), ant colony optimization (ACO) (Colorni et al., 1991), barnacle mating optimizer (BMO) (Sulaiman et al., 2020), black widow optimization (BWO) (Hayyolalam and Kazem, 2020), dragonfly algorithm (DA) (Mirjalili, 2016a), grasshopper optimization algorithm (GOA) (Saremi et al., 2017), Harris hawk optimization (HHO) (Heidari et al., 2019), marine predator algorithm (MPA) (Faramarzi et al., 2020b), manta ray foraging optimization (MRFO) (Zhao et al., 2020), satin bowerbird optimizer (SBO) (Moosavi and Bardsiri, 2017), and sailfish optimizer (SFO) (Shadravan et al., 2019).

Physics-based algorithms mimic physical rules or mathematical formulas, inspired by physical phenomena or mathematical models. For example, the gravitational search algorithm (GSA) (Rashedi et al., 2009) mimics Newton's law of gravity; the lightning search algorithm (LSA) (Shareef et al., 2015) was inspired by the natural phenomenon of lightning and the mechanism of stepped leader propagation; multiverse optimization (MVO) (Mirjalili et al., 2016) draws on the concept of white holes, black holes, and wormholes in cosmology; and Yin-Yang-pair optimization (YYPO) (Punnathanam and Kotecha, 2016) embodies the traditional Chinese dichotomous philosophy called Yin and Yang.

Human-behavior-based algorithms imitate human learning and social and competitive behavior and often reflect the psychological and physiological activities of humans in specific contexts. For example, the Fibonacci indicator algorithm (FIA) (Etminaniesfahani et al., 2018) is inspired by the Fibonacci index in the stock market, the harmony search algorithm (HSA) (Lee and Geem, 2005) imitates the process of jazz musicians' impromptu creation of beautiful music, the political optimizer (PO) (Askari et al., 2020) imitates the election campaigns of politicians, student psychology-based optimization (SPBO) (Das et al., 2020) is inspired by students' psychological activities in pursuit of high scores, learning-based optimization (TLBO) (Rao et al., 2011) imitates the behavior of classroom teaching, and the volleyball

*Corresponding author
✉ zhoushihua@dlu.edu.cn (S. Zhou)

premier league algorithm (VPA) (Moghdani and Salimi-fard, 2018) imitates the competition of teams in volleyball matches.

In addition to the original algorithms mentioned above, there is a large body of literature in which excellent research work has been done on algorithm improvement and application. For example, Leboucher integrated the ideas of game theory into the particle swarm algorithm, which has greatly improved population diversity and the exploration ability of the algorithm (Leboucher et al., 2018). Ghoumari proposed a maximum a posteriori principle to decide the best update strategy within a certain stage, and integrated it into evolutionary algorithm to enable learning (Ghoumari et al., 2018). Bojan Dragos utilized the GWO to adjust the parameters and rules of type-1 and type-2 fuzzy controllers to improve the performance of the control system (Bojan-Dragos et al., 2021). Precup successfully applied SMA to the optimization and tuning of interval type-2 fuzzy controllers, where the objective function is the sum of time multiplied by the square of the control error (Precup et al., 2021). Zamfirache presented an optimal control scheme for reference trace control using GSA combined with popular deep Q-learning of DRL variants (Zamfirache et al., 2022). However, no matter what kind of meta-heuristic algorithm is used to solve what kind of optimization problem, the biggest difficulty is to balance the exploration behavior and exploitation behavior of the algorithm in the process of solving the problem. Most explore first and exploit later, seeking the optimal solution neighborhood in early iterations, and exploiting it in later iterations to find the optimum. In reality, this often leads to partial optimization. The proposed TQA divides search agents into different identities to explore and exploit concurrently throughout iterations. Most optimization algorithms mimic a single behavior. An example is foraging, in which case food location indicates the optimal solution. TQA models the behavior of entire populations and presents the current most powerful individual location as the optimal solution. The validity of TQA is verified and evaluated through 23 mathematical test functions and six real-world engineering design problems, and experimental results show that it performs more satisfactorily than the most advanced algorithms in solving global optimization problems and engineering design problems.

The rest of this paper is organized as follows: Section 2 discusses the inspiration and model of TQA. Tests on benchmark functions and engineering design problems are described in sections 3 and 4, respectively. Section 5 relates our conclusions.

2. Termite queen algorithm

2.1. Inspiration

The termite is a widely distributed completely social insect, distributed across two-thirds of the earth's land area, concentrated in tropical and subtropical regions. Termites have a division of labor including a queen, workers, and soldiers. The queen lays eggs in the nest all her life; workers find food, serve the queen, and repair the nest; and soldiers

with huge jaws keep the group safe. Termites with different identities are shown in Figure 1.
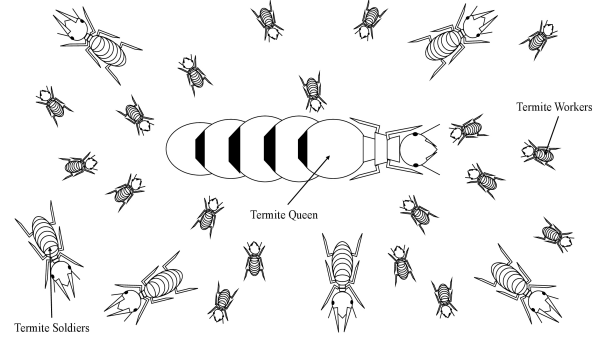

Figure 1: Termite family

For the continuation of the group, termite populations exhibit the behavior of marriage flight. From April through June each year, a large number of termites grow their wings under the chemical instructions of the queen, flying out of their nest in search of a new one. This behavior expands termite territory and avoids inbreeding, which can produce weak individuals. Inspired by the division of labor and marriage flight of termites, we have developed a new metaheuristic algorithm.

2.2. Local mapping principle and local mapping initialization

We propose a local mapping principle to transform a point to an area in multidimensional space. The idea is to determine which part of the divided area the point belongs to by its relative position in each dimension, as follows.

Step 1: Calculate the midpoint of each dimension, the formula can be found in (1):

$$Center_j = \frac{ub_j + lb_j}{2} \tag{1}$$

where ub_j and lb_j represent the upper and lower bounds, respectively, of the j-th dimension; and $Center_j$ represents its midpoint.

Step 2: Calculate the local index for each dimension, the formula can be found in (2):

$$local j = \begin{cases} 1, & Center_j \leq x_j \leq ub_j \\ -1 & lb_j \leq x_j \leq Center_j \end{cases} \tag{2}$$

where x_j and $Local_j$ are the x-coordinate and local index, respectively of the j-th dimension.

The upper and lower bounds of local mapping initialization are calculated based on the corresponding local index. For $Local_j = 1$, the upper bound of the coordinates of the j-th dimension is the upper bound of the whole space, and the lower bound is the midpoint. The local mapping principle and local mapping initialization are shown in Figure 2.
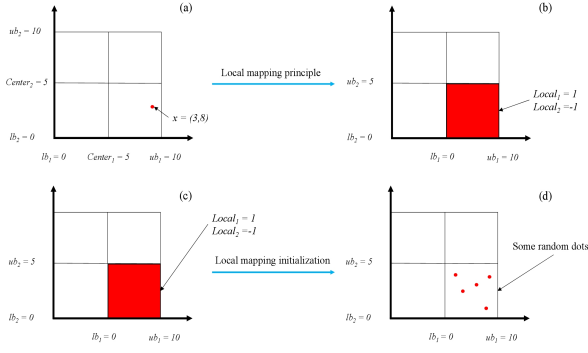
Figure 2: Local mapping, (a) and (b): local mapping principle; (c) and (d): local mapping initialization

2.3. TQA
2.3.1. Initialization

In TQA, each candidate solution is assumed to be a termite, whose population can be expressed as (3):

$$X = \begin{bmatrix} X_{1,1} & \cdots & X_{1,dim} \\ \vdots & \ddots & \vdots \\ X_{N,1} & \cdots & X_{N,dim} \end{bmatrix}_{N \times dim} \tag{3}$$

where N is the number of termites and dim is the number of dimensions in the search space. The upper and lower bound vectors of the search space are (4):

$$\begin{cases} \overrightarrow{ub} = [ub_1, ..., ub_{dim}] \\ \overrightarrow{lb} = [lb_1, ..., lb_{dim}] \end{cases} \tag{4}$$

where $\overrightarrow{ub}$ and $\overrightarrow{lb}$ respectively represent the upper and lower bounds of the search space. After initialization, X is evaluated, and the termite with the optimal fitness value is used as the current optimal solution.

2.3.2. Division

TQA is inspired by the division of labor in termite populations, which differentiates individuals by identity and updates their locations in different ways. TQA has five identities: queen, flying worker, foraging worker, serving worker, and soldier. The best solution represents the queen, and other candidate solutions are differentiated by the remaining four identities in each iteration. The division meets the following rules:

(1) To ensure a food source, $\frac{N}{2}$ foraging workers are maintained in each iteration;

(2) The remaining termites are divided into serving workers and soldiers in a 4:1 ratio;

(3) If marriage flight occurs, all individuals other than foraging workers become flying workers.

2.3.2.1. Foraging workers

If $Fitness_i$ of the i-th termite is less than the median fitness, $Fitness_t$, the termite is differentiated as a foraging worker. Foraging behavior is a back-and-forth movement, either away from or near the queen. The location of foraging workers is updated as (5):

$$\overrightarrow{X_i} = \overrightarrow{Target} \times r_1 + \overrightarrow{X_i} \times (1 - r_1) \tag{5}$$

where $\overrightarrow{Target}$ indicating the destination of the current movement, and it is chosen with equal probability between $\overrightarrow{BestX}$ (the queen position) and $\overrightarrow{UBestX}$ (the opposite position of the queen position). The formula for calculating $\overrightarrow{UBestX}$ can be found in (6):

$$\overrightarrow{UBestX} = \overrightarrow{ub} + \overrightarrow{lb} - \overrightarrow{BestX} \tag{6}$$

and r_1 is a random number in [0,1], which is used to control the search step of foraging workers. When r_1 is greater than 0.5, the updated point is closer to $\overrightarrow{Target}$. When r_1 is less than 0.5, the updated point is closer to $\overrightarrow{X_i}$.

2.3.2.2. Soldiers

A termite with fitness greater than $Fitness_t$ is differentiated as a soldier or serving worker with respective probabilities of 20 percent and 80 percent. A soldier exhibits a surrounding behavior centered on the queen, with its position updated as (7):

$$\overrightarrow{X_i} = (2\overrightarrow{x_t} - \overrightarrow{X_i}) \times r_2 + \overrightarrow{X_i} \times (1 - r_2)$$
$$\overrightarrow{x_t} = rand \times (\overrightarrow{ub} - \overrightarrow{lb}) + \overrightarrow{lb} \tag{7}$$

where $rand$ and r_2 are random numbers in [0,1]. $\overrightarrow{x_t}$ is a point on the axis of symmetry of the search space. In the search behavior, we first find the symmetry point of $\overrightarrow{X_i}$ with respect to $\overrightarrow{x_t}$, and then make a search behavior between the symmetry point and $\overrightarrow{X_i}$, in which the search step is controlled by r_2. When r_2 is greater than 0.5, the updated point is closer to the symmetry point When r_2 is less than 0.5, the updated point is closer to $\overrightarrow{X_i}$.

2.3.2.3. Serving workers

The queen moves her huge body with difficulty, and serving workers are responsible for helping her eat and clean her body. The process can be seen as serving workers approaching the queen from different directions, with positions updated as (8):

$$X_{i,j} = \begin{cases} BestX_j + c_1(bestx_j - BestX_j)c_2, & r_3 > 0.5 \\ BestX_j - c_1(bestx_j - BestX_j)c_2, & r_3 \leq 0.5 \end{cases} \tag{8}$$

where $\overrightarrow{bestx}$ is a random point in region s_1, which is a region calculated based on the local mapping principle in Section 2.2 and $\overrightarrow{bestx}$. $\overrightarrow{bestx_j}$ is the jth dimension of $\overrightarrow{bestx}$, and $\overrightarrow{BestX_j}$ is the jth dimension of $\overrightarrow{BestX}$. c_1 and c_2 control the exploitation intensity. The sigmoid function, c_1, and c_2 are calculated as (9)-(11):

$$Sigmoid(x) = \frac{1}{1 + e^{-x}} \tag{9}$$

$$c_1 = (1 - (Sigmoid(tan(-\frac{\pi}{2} + \pi(\frac{t}{T})^7)))^{\frac{1}{1000}} \times (\cos(\frac{\pi t}{T}) + 1) \quad (10)$$

$$c_2 = rand^{2-c_1} \quad (11)$$

where *rand* is a random number in [0,1], t and T are respectively the current and maximum number of iterations. The conventional decline factor is strictly decreasing, but in this paper, the product of c_1 and c_2 is used as a random decline factor, in which c_1 provides decline and c_2 provides randomness. The search behavior is developed with $\overrightarrow{BestX}$ as the starting point and $\overrightarrow{bestx} - \overrightarrow{BestX}$ as the step. When the product of c_1 and c_2 is less than 1, the updated point falls between $\overrightarrow{bestx}$ and $\overrightarrow{BestX}$. When the product of c_1 and c_2 is greater than 1, the updated point falls between $\overrightarrow{ubestx}$ and $\overrightarrow{BestX}$. Where $\overrightarrow{ubestx}$ is the symmetry point of $\overrightarrow{bestx}$ with respect to $\overrightarrow{BestX}$. r_3 is a random number in [0,1], which controls the direction of the search behavior.

2.3.2.4. Flying workers

To fall into local optima is similar in some ways to in-breeding in termite populations. Termites avoid this through marriage flight, and TQA incorporates this practice. In marriage flight, a large number of termite flying-workers fly to a new nest, with their positions updated as (12):

$$X_{i,j} = \begin{cases} LevyX_j + c_1(levyx_j - LevyX_j)c_2, & r_4 > 0.5 \\ LevyX_j - c_1(levyx_j - LevyX_j)c_2, & r_4 \leq 0.5 \end{cases} \quad (12)$$

where r_4 is a random number in [0,1], and c_1 and c_2 are determined by (10) and (11), respectively. The update mechanisms for (12) and (8) are essentially the same, except that $levyx_j$, and $BestX_j$ in (8) are replaced by $levyx_j$, and $LevyX_j$, respectively, in (12). Where $\overrightarrow{levyx}$ is a random point in region s_2, which is a region calculated based on the local mapping principle in Section2.2 and $\overrightarrow{levyx}$. $\overrightarrow{levyx_j}$ is the jth dimension of $\overrightarrow{levyx}$, and $\overrightarrow{LevyX_j}$ is the jth dimension of $\overrightarrow{LevyX}$.

Lévy flight is a special random movement (Magdziarz and Szczotka, 2016) whose steps can be generated by (13):

$$L(\lambda) = \frac{\lambda\beta\Gamma(\lambda)sin(\frac{\pi\lambda}{2})}{\pi s^{1+\lambda}} \quad (s \to \infty), s = \frac{U}{|V|^{\frac{1}{\lambda}}} \quad (13)$$

where $V \sim N(0,1)$, $U \sim N(0, \sigma^2)$, and σ^2 is calculated by (14):

$$\sigma^2 = \frac{\Gamma(1+\lambda)}{\lambda\Gamma(\frac{1+\lambda}{2})} \times \frac{sin(\frac{\pi\lambda}{2})}{2^{\frac{\lambda-1}{2}}} \quad (14)$$

where Γ is the gamma function and λ is 1.5. Lévy flight is updated as (15):

$$LevyX_j = BestX_j + s_j \quad (15)$$

Lévy flight random motion is essentially a large-scale jitter strategy, which helps to increase the exploration ability of the algorithm and avoid falling into local optima. It is worth noting that marriage flight rarely occurs, we control whether a marriage flight occurs by the value of *count*. When the ratio of $BestFitness_t$ to $BestFitness_{t-1}$ is greater than 0.999, where $BestFitness_t$ and $BestFitness_{t-1}$ are the optimal values for iterations t and t-1, respectively, *count* is incremented by 1. When *count* accumulates to 20, a marriage flight occurs and *count* is cleared to 0.

Figure 3 shows the pseudocode of TQA. The optimization process begins with the creation of a set of random candidate solutions. Termites are differentiated through fitness thresholds and randomness, and their locations are updated by formulas (5), (7), (8), and (12). In each iteration, the fitness values of all termites are evaluated, and the individual with the optimal fitness is the queen. When the optimal fitness value does not change significantly for a long time, half the termites will carry out a marriage flight to break through local optimality. There are six parameters in this algorithm, which are c_1, c_2, r_1, r_2, r_3 and r_4. Among them, c_1 and c_2 are linear decreasing factors, while r_1, r_2, r_3 and r_4 are random numbers in [0,1]. To reduce the complexity, there is no user parameters in the algorithm.

The Termite Queen Algorithm

Initialization {initialize the termite population $\overrightarrow{X_i}$ (i=1, 2, ..., N) considering $\overrightarrow{ub}$ and $\overrightarrow{lb}$
define c_1 and *count*
calculate the fitness of each termite
find $\overrightarrow{BestX}$ and *BestFitness*}

Main loop **while** ($t < T$)
    for each termite
        divided into four kinds of identity
        update all positions according to (5), (7), (8) and (12)
        amend the termite based on the $\overrightarrow{ub}$ and $\overrightarrow{lb}$
    end for
    update c_1 and *count*
    calculate the fitness of each termite
    update $\overrightarrow{BestX}$ and *BestFitness*
    $t = t + 1$
end while

Figure 3: Pseudocode of TQA

3. Results and discussion

Several sets of tests were conducted to determine the effectiveness of TQA. First, the performance of TQA was evaluated using 23 benchmark functions (Mirjalili et al., 2014), which can be categorized as single-peak (F1–F7), multi-peak (F8–F13), and composite (F14–F23). A single-peak function has only one global optimal value, which is used to evaluate the local exploitation ability of the algorithm, and multi-peak and composite functions with two or more optimal values are used to evaluate global exploration ability.

In addition, a large number of excellent meta-heuristic algorithms are used as comparison algorithms in the tests.

To make a fair comparison, the parameters of all comparison algorithms are set to the values suggested in the literature. See Table.1 for specific parameters.

Table 1
Controlling parameters values of the tested algorithms

Algorithms	Parameters	Values
GWO	a	$\in [0, 2]$
MFO	a	$\in [0, 2]$
MVO	WEPmax WEPmin	1 0.2
SCA	A r_1	2 $\in [0, 2]$
SSA	c_1	$\in (0, 2]$
WOA	a a2	$\in [0, 2]$ $\in [-1, 2]$
EO	a_1 a_2 GP	2 1 0.5
STOA	C_f S_A	2 $\in [0, 2]$
TSA	Pmax Pmin	4 1
TQA	c_1	$\in [0, 2]$

3.1. Qualitative results and discussion

The first set of tests was performed on the 2D of the first two variables for the benchmark functions to observe the behavior of TQA. Figure 4 describes the test functions selected from three categories, which qualitatively demonstrate the performance of TQA.

The experiments were conducted with $N = 10$ termites, and the maximum number of iterations $T = 200$. Qualitative results included the search history, average fitness curve, and convergence curve. From the search history in Figure 4, it can be seen that the search agents of TQA are well distributed in the search space, which indicates that TQA has a satisfactory exploration ability. It is worth noting that from the search history of F8, TQA has been exploited for a long time in the lower-left area, but the obtained best value is in the upper-right corner; this means that TQA successfully jumped out of the local optimal solution. The ability to jump out of the local optimum is mainly aided by Levy flight and restart idea. Some search agents would happen Levy flights when the best fitness obtained so far has not changed greatly for a long time, which helps the algorithm jump out of the local optimal solution to some extent. From the mean fitness curves in Figure 4, it can be seen that TQA maintained good population diversity throughout the iteration. This population diversity results primarily from division mechanisms and also helps the algorithm jump out of local optima to some extent. From the convergence curves in Figure 4, it can be seen that TQA has a fast convergence speed on all six functions, which indicates that

TQA has a good convergence. The convergence is mainly supported by the exploitation ability resulting from the local mapping principle. Hence TQA balances exploitation and exploration, and solves the problem of partial optimization to a certain extent.

To further illustrate the convergence of TQA, TQA was performed on test functions F1–F23 and compared to nine classic and advanced algorithms: grey wolf optimizer (GWO) (Mirjalili et al., 2014), moth flame optimization (MFO) (Mirjalili, 2015), multiverse optimizer (MVO) (Mirjalili et al., 2016), sine cosine algorithm (SCA) (Mirjalili, 2016b), salp swarm algorithm (SSA) (Mirjalili et al., 2017), whale optimization algorithm (WOA) (Mirjalili and Lewis, 2016), equilibrium optimizer (EO) (Faramarzi et al., 2020a), sooty tern optimization algorithm (STOA) (Dhiman and Kaur, 2019), and tunicate swarm algorithm (TSA) (Kaur et al., 2020). For a fair comparison, all algorithms used the same experimental conditions: population size N=10; the maximum number of iterations T=100; the number of dimensions dim=30; and an Intel Core i5-10500 with a 3.10-GHz CPU and 16 GB RAM, programmed using MATLAB R2020b. The experimental results can be found in Figure 5. As can be seen from Figure 5, TQA has the best convergence effect and fastest convergence speed on F1- F7 and F9-13. This shows that TQA has satisfactory convergence performance on unimodal and multimodal functions. TQA has a slightly slower convergence speed on F8 and F14-23. According to the analysis, this may result from the fact that the search space of the composite function is more complicated than other functions. It is worth noting that the slower convergence speed in the composite function can avoid the algorithm from getting trapped in a local optimum to some extent.

3.2. Quantitative results and discussion

In the second set of tests, TQA was performed on test functions F1–F23 and compared to nine classic and advanced algorithms: grey wolf optimizer (GWO) (Mirjalili et al., 2014), moth flame optimization (MFO) (Mirjalili, 2015), multiverse optimizer (MVO) (Mirjalili et al., 2016), sine cosine algorithm (SCA) (Mirjalili, 2016b), salp swarm algorithm (SSA) (Mirjalili et al., 2017), whale optimization algorithm (WOA) (Mirjalili and Lewis, 2016), equilibrium optimizer (EO) (Faramarzi et al., 2020a), sooty tern optimization algorithm (STOA) (Dhiman and Kaur, 2019), and tunicate swarm algorithm (TSA) (Kaur et al., 2020). For a fair comparison, all algorithms used the same experimental conditions: population size N=30; the maximum number of iterations T= 500; the number of dimensions dim=30; and an Intel Core i5-10500 with a 3.10-GHz CPU and 16 GB RAM, programmed using MATLAB R2020b. Each algorithm ran 30 times independently, and the mean ($Mean$) and standard deviation (Std) were recorded as a basis to evaluate its performance and robustness.

The experimental results can be found in Table.2-4, where the best results for each function are bolded. According to Table.2: On one hand, TQA produced the best
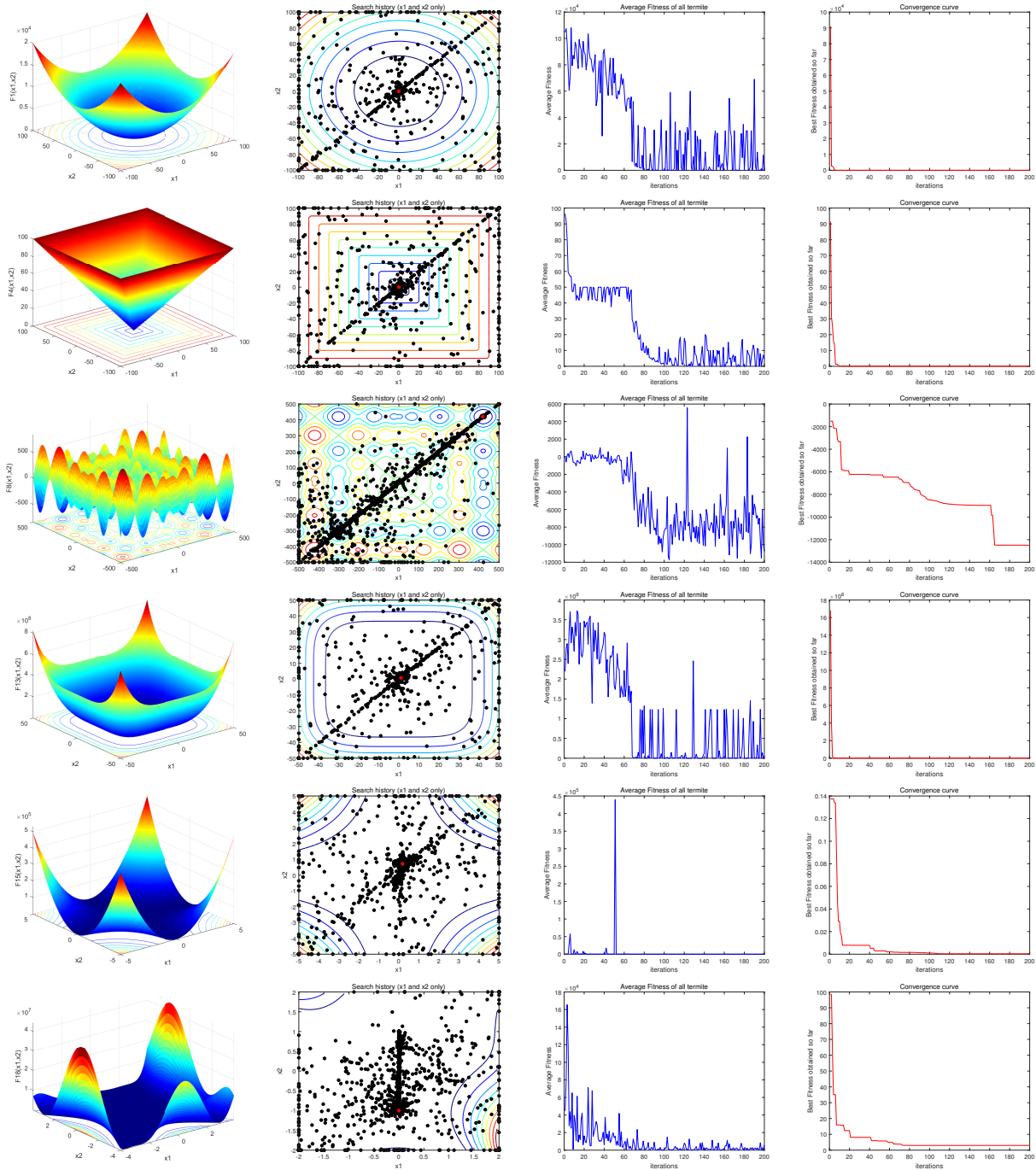
Figure 4: Performance of TQA on unimodal, multimodal, and composite test functions

results on six single-peak functions, and was only slightly worse than SSA on F6. This indicates that TQA have a satisfactory exploitation ability. On the other hand, TQA performed well on all unimodal functions, suggesting that the optimization of single extreme values can be better solved by TQA. According to Table.3, TQA found the global optimal value on F8–F11 and obtained the best results on F12 and F13. This indicates that TQA has satisfactory exploration ability to better solve optimization problems with multiple extreme values. According to Table.4, TQA produced the

best results of the comparison algorithms on both F14-23, TQA achieved optimum values on the multimodal functions except F15 and F20. To sum up, the good performance and robustness of TQA in solving all three types of optimization problems suggest that TQA has generality in solving optimization problems.

Although the superiority of TQA has been demonstrated by the comparison of averages, a significance test is required to verify the significant difference between TQA and its competitors. There are both parametric and nonparametric
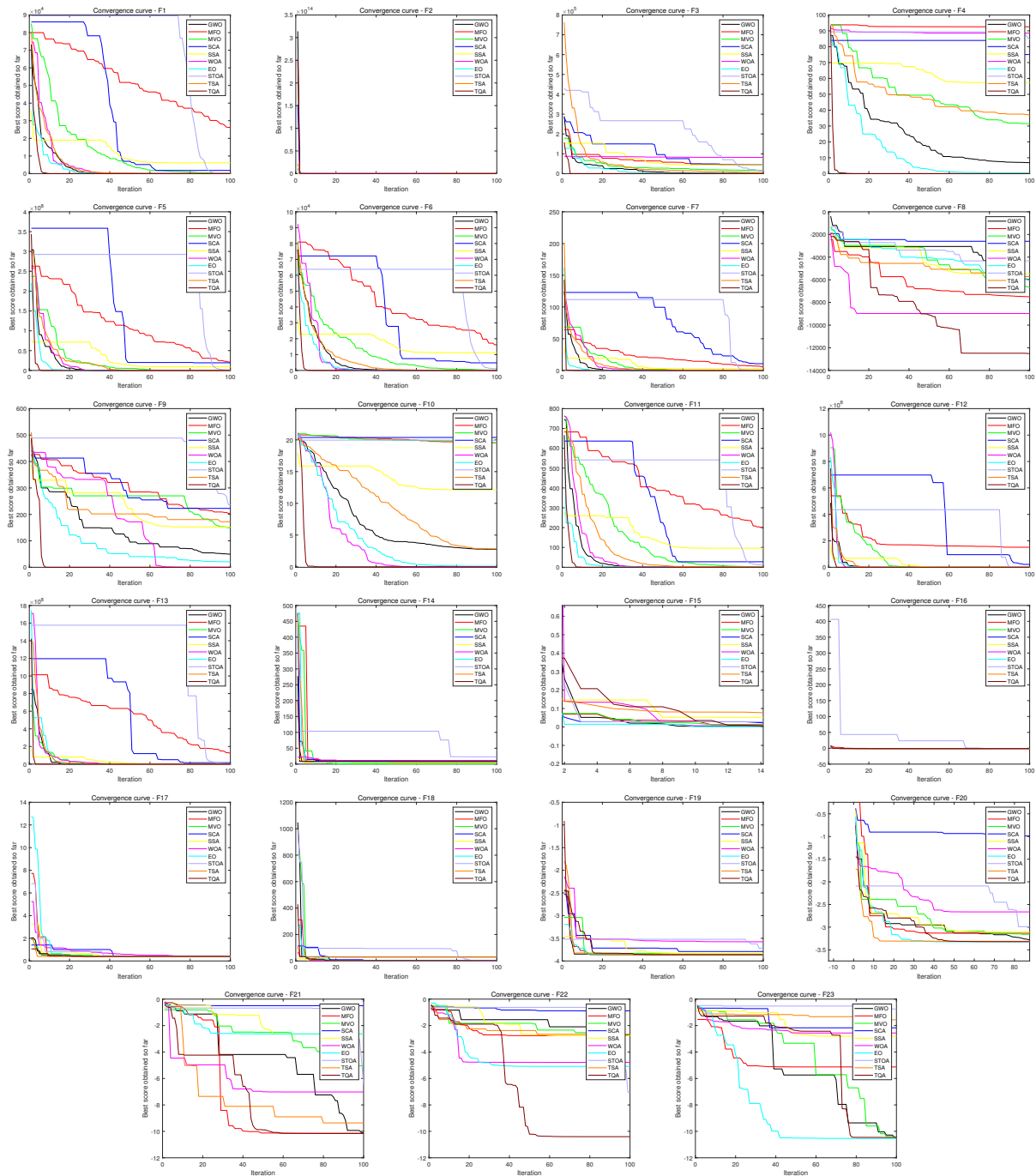
Figure 5: Comparative analysis of convergence

significance tests, but a parametric test must meet the conditions of normality and independence. Therefore, nonparametric tests are generally used to assess the differences between metaheuristic optimization algorithms(Derrac et al., 2011).

We used the Wilcoxon test method (Derrac et al., 2011), whose results can be found in Table.5, to statistically analyze the comparative results, where $R^=$, R^+, and R^- respectively indicate the number of functions for which TQA is similar to, worse than, and superior to other algorithms. The Wilcoxon

test uses TQA as a control algorithm. The significance level of statistical detection was set to 0.05. So, a p-value less than 0.05, which is bolded, indicates a significant difference between algorithms. According to the results in Table.5, the p-values in nine pair of comparisons were all less than 0.05, indicating that there were obvious differences between TQA and other nine comparison algorithms. And the number of the functions in which TQA occupied the dominant position was more than 78% (18 / 23) in nine pairs of comparisons.

Table 2
Results and comparison of algorithms on F1–F7 benchmark functions with Dim = 30

Function		GWO	MFO	MVO	SCA	SSA	WOA	EO	STOA	TSA	TQA
F1	Mean	1.9562E-27	1.3448E+03	1.1434E+00	9.7521E+00	2.0984E-07	3.2105E-72	1.5728E-40	4.3988E-07	1.0605E-21	**0.0000E+00**
	Std	2.7956E-27	3.4537E+03	3.0478E-01	1.3965E+01	3.8216E-07	1.2602E-71	6.2401E-40	1.2357E-06	1.5708E-21	0.0000E+00
F2	Mean	9.0627E-17	3.3416E+01	9.3185E-01	2.4431E-02	2.5680E+00	3.6782E-51	6.1210E-24	1.1618E-05	1.1899E-13	**3.9762E-176**
	Std	4.5868E-17	1.6635E+01	6.2934E-01	4.1505E-02	1.8681E+00	1.0665E-50	5.5702E-24	8.5255E-06	1.2515E-13	0.0000E+00
F3	Mean	4.2951E-05	2.5267E+04	2.1394E+02	8.5688E+03	1.5343E+03	4.1707E+04	6.4027E-08	1.1994E-01	3.5614E-04	**0.0000E+00**
	Std	1.3474E-04	1.3103E+04	9.4735E+01	5.3325E+03	9.3251E+02	1.6373E+04	2.8118E-07	3.3477E-01	8.5557E-04	0.0000E+00
F4	Mean	7.5946E-07	7.1353E+01	2.0398E+00	3.9145E+01	1.1102E+01	4.6662E+01	3.0488E-10	5.4052E-02	3.3920E-01	**1.3088E-173**
	Std	7.3955E-07	7.1185E+00	6.1462E-01	1.0200E+01	3.6860E+00	2.6534E+01	3.7387E-10	8.0566E-02	7.4562E-01	0.0000E+00
F5	Mean	2.6887E+01	5.3512E+06	6.0042E+02	5.2985E+04	2.1702E+02	2.7977E+01	2.5322E+01	2.8405E+01	2.8736E+01	**8.8965E-01**
	Std	8.3203E-01	2.0289E+07	8.3737E+02	9.6851E+04	3.3875E+02	4.3860E-01	1.6719E-01	4.6580E-01	3.8427E-01	3.1368E+00
F6	Mean	7.9532E-01	1.6608E+03	1.3068E+00	1.9740E+01	**1.3179E-07**	3.6813E-01	1.0012E-05	2.5604E+00	3.7259E+00	5.2682E-06
	Std	4.0530E-01	3.7671E+03	3.5908E-01	3.0238E+01	1.1347E-07	1.7132E-01	8.0897E-06	4.5907E-01	6.1839E-01	4.0589E-06
F7	Mean	2.0165E-03	2.7666E+00	3.4178E-02	1.6134E-01	1.8610E-01	3.3415E-03	1.2837E-03	6.6170E-03	9.4419E-03	**1.1069E-04**
	Std	1.1451E-03	5.4732E+00	1.7503E-02	3.3388E-01	7.7992E-02	4.2141E-03	7.1285E-04	4.4729E-03	3.6333E-03	1.4585E-04

Table 3
Results and comparison of algorithms on F8–13 benchmark functions with Dim = 30

Function		GWO	MFO	MVO	SCA	SSA	WOA	EO	STOA	TSA	TQA
F8	Mean	-5.8148E+03	-8.2322E+03	-7.8483E+03	-3.7265E+03	-7.6365E+03	-1.1128E+04	-8.9168E+03	-5.0798E+03	-5.9514E+03	**-1.2569E+04**
	Std	8.3653E+02	9.7825E+02	6.4318E+02	3.1245E+02	9.1923E+02	1.5937E+03	6.8890E+02	5.0103E+02	5.3532E+02	6.3522E-06
F9	Mean	2.7226E+00	1.6961E+02	1.3117E+02	4.1630E+01	4.8521E+01	**0.0000E+00**	3.3166E-02	6.0971E+00	1.8778E+02	**0.0000E+00**
	Std	3.6098E+00	2.6407E+01	2.7533E+01	3.9655E+01	1.4198E+01	0.0000E+00	1.8166E-01	5.0903E+00	4.6939E+01	0.0000E+00
F10	Mean	9.6930E-14	1.3734E+01	1.8635E+00	1.1933E+01	2.5855E+00	4.5593E-15	8.2305E-15	1.9960E+00	1.4727E+00	**8.8818E-16**
	Std	1.9447E-14	7.7487E+00	7.5545E-01	9.2950E+00	8.7928E-01	2.7174E-15	1.5979E-15	1.4713E-03	1.6082E+00	0.0000E+00
F11	Mean	3.1903E-03	2.5021E+01	8.4409E-01	9.8394E-01	1.9761E-02	**0.0000E+00**	**0.0000E+00**	4.0537E-02	7.9026E-03	**0.0000E+00**
	Std	8.8711E-03	4.6953E+01	1.0224E-01	3.2306E-01	1.5999E-02	0.0000E+00	0.0000E+00	5.7336E-02	7.9461E-03	0.0000E+00
F12	Mean	4.0046E-02	1.7067E+07	2.4153E+00	5.0273E+04	6.3453E+00	2.4478E-02	6.0473E-07	2.8967E-01	7.1167E+00	**8.9470E-08**
	Std	1.5629E-02	6.4949E+07	1.3898E+00	1.9127E+05	3.0695E+00	2.0836E-02	3.9036E-07	1.6360E-01	4.5914E+00	7.2218E-08
F13	Mean	6.7088E-01	2.7520E+02	1.7198E-01	5.9457E+04	1.5944E+01	4.7197E-01	1.8155E-02	1.9155E+00	3.0472E+00	**2.5336E-03**
	Std	2.1229E-01	1.3504E+03	7.6771E-02	1.3375E+05	1.2467E+01	2.6565E-01	3.5221E-02	1.9481E-01	6.8099E-01	5.4255E-03

Therefore, TQA is significantly better than the other algorithms at solving F1–F23. To sum up, we infer that TQA is a satisfactory optimization algorithm, with good performance and robustness at solving global optimization problems.

4. Termite queen algorithm for engineering problems

TQA was tested on six widely used engineering problems to verify its ability to solve real-world problems, and the results were compared with those of algorithms such as GWO, MVO, SCA, SSA, WOA, STOA, and TSA. For each problem, the population size was set to $N = 50$, and the maximum number of iterations was $T = 1000$. For the convenience of calculation, the dimensions solved in the metaheuristic algorithm corresponded to the variables in the problem. Constraints were imposed through a penalty function (Yang, 2010). All experiments were performed on an Intel Core i5-10500 CPU at 3.10 GHz with 16 GB RAM, and programmed using MATLAB R2020b. Each algorithm ran independently 30 times. The optimal and average fitness value, standard deviation, and optimal solution were recorded, and performance was measured by the optimal and average fitness values.

4.1. Three-bar truss design problem

The three-bar truss design problem is a classic minimization problem in engineering (Pelusi et al., 2020), whose

structure can be found in Figure 6, and whose model can be found in (16)-(18). It has buckling, deflection, and stress constraints. The goal is to reduce or minimize the weight of the three trusses.
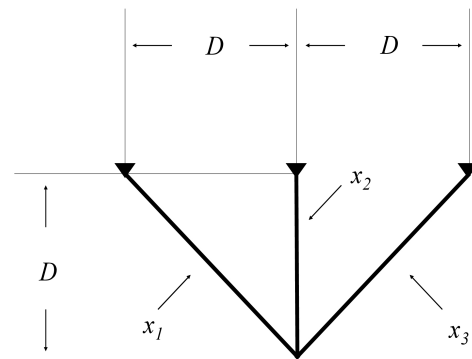
Figure 6: Three-bar truss design problem

Fuction:

$$minf(x) = (2\sqrt{2}x_1 + x_2) \times l \tag{16}$$

Table 4

Results and comparison of algorithms on F14–F23 benchmark functions with fixed Dim

Function		GWO	MFO	MVO	SCA	SSA	WOA	EO	STOA	TSA	TQA
F14	Mean	4.4644E+00	1.8877E+00	**9.9800E-01**	1.5961E+00	1.0974E+00	2.6071E+00	**9.9800E-01**	1.7918E+00	7.4077E+00	**9.9800E-01**
	Std	3.7820E+00	1.6196E+00	3.5200E-11	9.2294E-01	3.0331E-01	2.5095E+00	1.7973E-16	9.5371E-01	4.8491E+00	1.3191E-10
F15	Mean	6.3691E-03	2.2314E-03	4.7620E-03	1.0634E-03	2.8436E-03	7.0010E-04	2.3210E-03	2.3652E-03	8.8125E-03	**5.2927E-04**
	Std	9.3181E-03	4.9372E-03	7.9392E-03	3.7743E-04	5.9459E-03	4.0346E-04	6.1169E-03	4.9015E-03	1.7503E-02	3.0942E-04
F16	Mean	**-1.0316E+00**	**-1.0316E+00**	**-1.0316E+00**	**-1.0316E+00**	**-1.0316E+00**	**-1.0316E+00**	**-1.0316E+00**	**-1.0316E+00**	-1.0264E+00	**-1.0316E+00**
	Std	2.1577E-08	6.7752E-16	3.9331E-07	6.4874E-05	2.9876E-14	5.7301E-10	6.4539E-16	1.3271E-06	1.1989E-02	4.3831E-16
F17	Mean	**3.9789E-01**	**3.9789E-01**	**3.9789E-01**	3.9996E-01	**3.9789E-01**	**3.9789E-01**	**3.9789E-01**	3.9816E-01	3.9792E-01	**3.9789E-01**
	Std	2.0213E-06	0.0000E+00	6.6213E-07	2.4910E-03	2.9859E-14	8.4274E-06	0.0000E+00	3.7851E-04	3.5979E-05	0.0000E+00
F18	Mean	**3.0000E+00**	**3.0000E+00**	**3.0000E+00**	3.0001E+00	**3.0000E+00**	3.0001E+00	**3.0000E+00**	**3.0000E+00**	1.4700E+01	**3.0000E+00**
	Std	3.2849E-05	2.5737E-15	3.3021E-06	1.2389E-04	3.3181E-13	9.9315E-05	6.4407E-16	1.0585E-04	2.5252E+01	5.6763E-15
F19	Mean	-3.8621E+00	**-3.8628E+00**	**-3.8628E+00**	-3.8544E+00	**-3.8628E+00**	-3.8545E+00	**-3.8628E+00**	-3.8556E+00	-3.8624E+00	**-3.8628E+00**
	Std	1.5177E-03	2.7101E-15	6.1616E-06	2.4602E-03	2.8516E-10	1.4880E-02	2.5094E-15	2.2802E-03	1.4168E-03	2.1362E-15
F20	Mean	-3.2777E+00	-3.2318E+00	-3.2493E+00	-2.9108E+00	-3.2271E+00	-3.1977E+00	-3.2741E+00	-2.9946E+00	-3.2519E+00	**-3.2822E+00**
	Std	7.8609E-02	6.3071E-02	6.0379E-02	3.3906E-01	5.8674E-02	1.5290E-01	5.9732E-02	3.5689E-01	8.5457E-02	5.7247E-02
F21	Mean	-9.2289E+00	-6.9757E+00	-6.4534E+00	-2.2254E+00	-7.9018E+00	-8.1977E+00	-8.4639E+00	-2.3496E+00	-5.4045E+00	**-1.0153E+01**
	Std	2.1362E+00	3.3463E+00	3.0056E+00	1.7588E+00	3.3015E+00	2.8599E+00	2.6827E+00	3.3925E+00	3.3977E+00	4.4050E-14
F22	Mean	-1.0401E+01	-8.8396E+00	-7.2070E+00	-3.5079E+00	-8.7598E+00	-7.7343E+00	-9.8260E+00	-4.5271E+00	-6.8235E+00	**-1.0403E+01**
	Std	1.3867E-03	2.9218E+00	3.3569E+00	2.5284E+00	2.8361E+00	3.1723E+00	1.7726E+00	4.1217E+00	3.6301E+00	3.8599E-14
F23	Mean	-1.0535E+01	-8.3292E+00	-8.5099E+00	-4.1933E+00	-8.1966E+00	-7.0573E+00	-1.0266E+01	-5.2456E+00	-6.6979E+00	**-1.0536E+01**
	Std	1.1256E-03	3.4561E+00	3.2245E+00	1.8242E+00	3.2062E+00	3.1828E+00	1.4815E+00	4.2821E+00	3.7094E+00	3.8417E-14

Table 5

Wilcoxon statistical test results (control algorithm: TQA)

Comparison	R^+	R^-	$R^=$	p-value
GWO vs TQA	0	20	3	**8.9000E-05**
MFO vs TQA	0	19	4	**1.3183E-04**
MVO vs TQA	0	18	5	**1.9644E-04**
SCA vs TQA	0	22	1	**4.0100E-05**
SSA vs TQA	1	18	4	**1.8218E-04**
WOA vs TQA	0	19	4	**1.3183E-04**
EO vs TQA	0	18	5	**1.9644E-04**
STOA vs TQA	0	21	2	**5.9570E-05**
TSA vs TQA	0	23	0	**2.7016E-05**

Subject to:

$$g_1(x) = \frac{\sqrt{2}x_1 + x_2}{(\sqrt{2}x_1^2 + 2x_1x_2)}P - \sigma \leq 0$$

$$g_2(x) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0$$

$$g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1}P - \sigma \leq 0$$

$$0 \leq x_i \leq 1, \quad i = 1, 2 \tag{17}$$

Parameter:

$$l = 100cm; \ P = 2kN/(cm^2); \ \sigma = 2kN/(cm^2) \tag{18}$$

Table.6 shows the statistical results of each algorithm for solving this problem, where the best results are shown in boldface. According to the results, TQA obtained the best optimal value, and its average value was second only to SSA.

4.2. Gear train design problem

The goal of the gear train design problem (Zhong et al., 2021) is to reduce or minimize the transmission-specific cost

of the gear train shown in Figure 7. The design variables are the numbers of gears N_a, N_b, N_d, and N_f, expressed respectively as x_1, x_2, x_3, and x_4. The mathematical model of the problem can be found in (19) and (20).
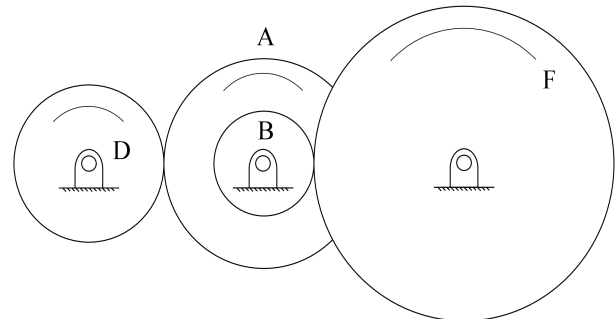

Figure 7: Gear train design problem

Fuction:

$$min f(x) = (\frac{1}{6.931} - \frac{x_2x_3}{x_1x_4})^2 \tag{19}$$

Subject to:

$$12 \leq x_i \leq 60, \quad i = 1, 2, 3, 4 \tag{20}$$

Table.7 shows the statistical results of each algorithm at solving the problem, where the best results are bolded, according to which TQA obtained the best optimal values and averages.

4.3. I-beam structure design problem

The object of the I-beam structure design problem is to minimize the vertical deviation of the chemical beam by

Table 6
Statistical results of the three-bar truss design problem

Function	GWO	MVO	SCA	SSA	WOA	STOA	TSA	TQA
Best	263.896109	263.895975	263.921243	263.895845	263.895908	263.897043	263.896667	**263.895843**
x_1	0.789201	0.788415	0.786924	0.788633	0.788379	0.788115	0.788158	0.788688
x_2	0.406765	0.408987	0.413456	0.408366	0.409088	0.409843	0.409718	0.408211
Mean	263.898476	263.897523	264.615867	**263.896421**	264.148125	267.707187	263.902906	263.897510
Std	2.2593E-03	1.9914E-03	3.4428E+00	1.0091E-03	3.4909E-01	7.6972E+00	5.4784E-03	2.6324E-03

Table 7
Statistical results of gear train design problem

Function	GWO	MVO	SCA	SSA	WOA	STOA	TSA	TQA
Best	1.3466E-16	5.3766E-16	4.5239E-13	2.3485E-23	1.0969E-16	5.1194E-15	5.1194E-15	**0.0000E+00**
x_1	32.840852	40.286228	55.358357	17.967248	60.000000	50.918059	50.918059	32.904544
x_2	19.744619	12.126915	17.049617	12.553056	43.283794	13.139311	13.139311	12.044375
x_3	13.457697	20.852128	17.213230	12.296074	12.000000	12.000000	12.000000	14.587879
x_4	56.079087	43.505092	36.744106	59.542939	60.000000	21.462371	21.462371	37.009774
Mean	5.0122E-13	1.9388E-13	1.2133E-10	1.2999E-20	3.5383E-11	9.0118E-13	9.0118E-13	**4.0228E-30**
Std	1.9700E-12	2.4629E-13	1.6799E-10	2.7013E-20	6.5390E-11	1.4072E-12	1.4072E-12	1.7077E-29

adjusting parameters $h(x_1)$, $b(x_2)$, $t_w(x_3)$, and $t_f(x_4)$ (Li et al., 2020), as shown in Figure 8. The model of the problem can be found in (21) and (22).
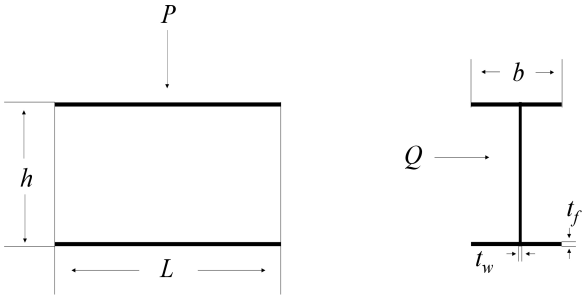

Figure 8: I-beam structure design problem

Fuction:

$$min f(x) = \frac{5000}{\frac{x_3(x_1-2x_4)}{12} + \frac{x_2 x_4^3}{6} + 2x_2 x_4 (\frac{x_1-x_4}{2})^2} \quad (21)$$

Subject to:

$$g_1(x) = 2x_2 x_3 + x_3(x_1 - 2x_4) \le 300$$

$$g_2(x) = \frac{(18x_1 \times 10^4)}{x_3(x_1 - 2x_4)^3 + 2x_2 x_3(4x_4^2 + 3x_1(x_1 - 2x_4))}$$

$$+ \frac{(15x_2 \times 10^3)}{(x_1 - 2x_4)x_3^3 + 2x_3 x_2^3} - 56$$

$$10 \le x_1 \le 80; \ 10 \le x_2 \le 50; \ 0.9 \le x_3, x_4 \le 5 \quad (22)$$

Table.8 shows the statistical results of each algorithm at solving the problem, where the best results are bolded, and which show that TQA obtained the best optimal values and averages.

4.4. Welded beam design problem

The purpose of this problem is to constrain the deflection of beam δ, buckling load of bar P_c, bending stress θ, and shear stress τ to minimize the cost of the welded beam (Karami et al., 2021). The structure can be found in Figure 9. The variables are weld thickness h (x_1), length of bar l (x_2), height t (x_3), and thickness of beam b (x_4). The model of the problem can be found in (23)–(25).
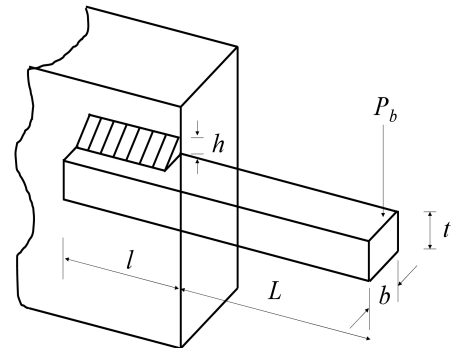

Figure 9: Welded beam design problem

Fuction:

$$min f(x) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14 + x) \quad (23)$$

Table 8
Statistical results of the I-beam structure design problem

Function	GWO	MVO	SCA	SSA	WOA	STOA	TSA	TQA
Best	**0.013074**	**0.013074**	0.013075	**0.013074**	**0.013074**	**0.013074**	**0.013074**	**0.013074**
x_1	80.000000	80.000000	80.000000	80.000000	80.000000	80.000000	80.000000	80.000000
x_2	50.000000	49.978108	50.000000	49.999996	50.000000	50.000000	50.000000	50.000000
x_3	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000	0.900000
x_4	2.321788	2.322826	2.321626	2.321792	2.321792	2.321746	2.321792	2.321792
Mean	**0.013074**	0.013075	0.013097	0.013076	0.013233	0.013076	**0.013074**	**0.013074**
Std	2.4346E-07	7.8044E-07	2.0601E-05	4.2743E-06	1.7577E-04	1.3405E-06	3.1784E-07	8.3035E-08

Subject to:

$$g_1(x) = \tau(x) - \tau_{max} \leq 0$$
$$g_2(x) = \sigma(x) - \sigma_{max} \leq 0$$
$$g_3(x) = x_1 - x_4 \leq 0$$
$$g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14 + x) - 5 \leq 0$$
$$g_5(x) = 0.125 - x_1 \leq 0$$
$$g_6(x) = \delta(x) - \delta_{max} \leq 0$$
$$g_7(x) = P - P_c(x) \leq 0$$
$$0.1 \leq x_i \leq 2, \quad i = 1, 4; \quad 0.1 \leq x_i \leq 10, \quad i = 2, 3 \quad (24)$$

Parameter:

$$P = 6000 \; lb; \; E = 30 \times 10^6 \; psi; \; G = 12 \times 10^6 \; psi;$$
$$L = 14 \; in; \; \tau_{max} = 13600 \; psi; \; \sigma_{max} = 30000 \; psi;$$

$$\delta_{max} = 0.25 \; in; \; M = P(L + \frac{x_2}{2}); \; R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2};$$

$$J = 2(\sqrt{2}x_1x_2(\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2)); \; \tau_1 = \frac{P}{\sqrt{2}x_1 + x_2};$$

$$\tau_2 = \frac{MR}{J}; \; \tau(x) = \sqrt{\tau_1^2 + 2\tau_1\tau_2 + \frac{x_2}{2R}\tau_2^2}; \; \sigma(x) = \frac{6PL}{x_4x_3^2};$$

$$\delta(x) = \frac{4PL^3}{Ex_3^3x_4}; \; P_c(x) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}) \quad (25)$$

Table.9 shows the statistical results of each algorithm at solving this problem, where the best results are shown in bold. According to the results, TQA obtained the best optimal value, and its average value was second only to GWO.

4.5. Cantilever beam design problem

The structure of this civil engineering problem (Kamboj et al., 2020), as shown in Figure 10, consists of five hollow elements, each with a hollow section of equal thickness. The goal is to reduce or minimize the weight of the cantilever beam. The thickness of the beam is constant, and the design variable is the cross-sectional width of the five elements. The model of this problem can be found in (26) and (27).
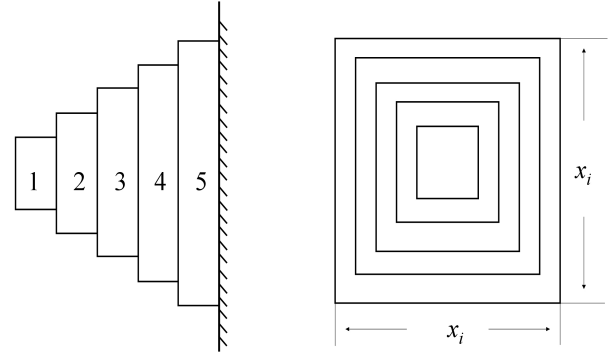

Figure 10: Cantilever beam design problem

Fuction:

$$min f(x) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5), \quad x_i > 0 \quad (26)$$

Subject to:

$$g_1(x) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} \leq 1$$
$$0.01 \leq x_i \leq 100, \quad i = 1, 2, 3, 4, 5 \quad (27)$$

Table.11 shows the statistical results of each algorithm at solving the problem, where the best results are shown in bold. According to the results, TQA obtained the best optimal values, and its average value was second only to SSA.

4.6. Speed reducer design problem

This problem is more complex (Sun and othersg, 2021), involving seven design variables: the face width $b(x_1)$, module of teeth $m(x_2)$, number of teeth in the pinion $z(x_3)$, length of the first shaft between bearings $l_1(x_4)$, length of the second shaft between bearings $l_2(x_5)$, diameter of first shaft $d_1(x_6)$, and diameter of second shaft $d_2(x_7)$. The goal is to minimize the total weight of the reducer. The structure can be found in Figure 10, and the mathematical model of the problem can be found in (28) and (29).

Table 9
Statistical results of the welded beam design problem

Function	GWO	MVO	SCA	SSA	WOA	STOA	TSA	TQA
Best	1.725534	1.727411	1.773580	1.728225	1.748147	1.727887	1.728294	**1.724853**
x_1	0.205630	0.205555	0.192200	0.204541	0.195522	0.205048	0.204966	0.205729
x_2	3.473854	3.466642	3.773926	3.488065	3.712906	3.478242	3.482987	3.470497
x_3	9.037218	9.058817	9.190639	9.057766	9.016764	9.057847	9.047889	9.036624
x_4	0.205766	0.205667	0.206080	0.205624	0.207104	0.205649	0.205861	0.205730
Mean	**1.726783**	1.739013	1.858889	1.825995	2.259938	1.737993	1.742316	1.735345
Std	9.6817E-04	1.1024E-02	3.6744E-02	9.7915E-02	6.3604E-01	8.2362E-03	4.9857E-03	2.3349E-02

Table 10
Statistical results of cantilever beam design problem

| Function	GWO	MVO	SCA	SSA	WOA	STOA	TSA	TQA
Best	1.339959	1.340007	1.353852	**1.339957**	1.343234	1.340096	1.340139	**1.339957**
x_1	6.009299	6.034015	5.867151	6.012188	5.909282	6.012432	6.044932	6.013593
x_2	5.310425	5.310577	5.162536	5.306727	5.321049	5.330040	5.305516	5.307230
x_3	4.494518	4.503953	4.700564	4.496454	4.779510	4.458157	4.438456	4.498556
x_4	3.505698	3.492451	3.532458	3.504754	3.487557	3.541110	3.524680	3.502111
x_5	2.153756	2.133478	2.433641	2.153553	2.028782	2.134160	2.162995	2.152181
Mean	1.339998	1.340418	1.379948	**1.339967**	1.430665	1.340710	1.340901	1.339975
Std	2.6519E-05	3.0554E-04	1.6282E-02	7.6428E-06	7.4887E-02	7.5025E-04	4.3682E-04	1.6258E-05
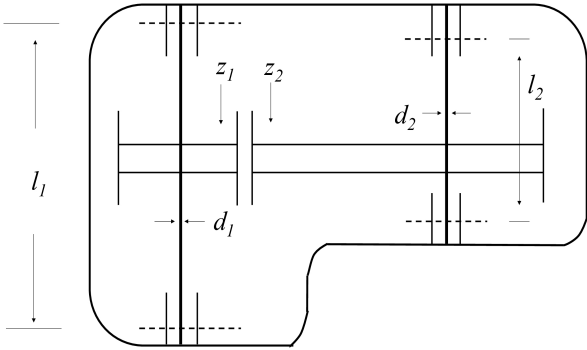
Figure 11: Speed reducer design problem

Fuction:

$$min f(x) = 0.7854 x_1 x_2^2 (3.3333 x_3^2 + 14.9334 x_3 - 43.0934)$$
$$- 1.508 x_1 (x_6^2 + x_7^2) + 7.4777(x_6^3 + x_7^3) \tag{28}$$

Subject to:

$$g_1(x) = \frac{27}{x_1 x_2^2 x_3} - 1 \le 0$$

$$g_2(x) = \frac{397.5}{x_1 x_2^2 x_3^2} - 1 \le 0$$

$$g_3(x) = \frac{1.93 x_4^3}{x_2 x_3 x_6^4} - 1 \le 0$$

$$g_4(x) = \frac{1.93 x_5^3}{x_2 x_3 x_7^4} - 1 \le 0$$

$$g_5(x) = \frac{((\frac{754 x_4}{x_2 x_3})^2 + 16.9 \times 10^6)^{\frac{1}{2}}}{110 x_6^3} - 1 \le 0$$

$$g_6(x) = \frac{((\frac{754 x_5}{x_2 x_3})^2 + 157.5 \times 10^6)^{\frac{1}{2}}}{85 x_7^3} - 1 \le 0$$

$$g_7(x) = \frac{x_2 x_3}{40} - 1 \le 0$$

$$g_8(x) = \frac{5 x_2}{x_1} - 1 \le 0$$

$$g_9(x) = \frac{x_1}{12 x_2} - 1 \le 0$$

$$g_10(x) = \frac{1.5 x_6 + 1.9}{x_4} - 1 \le 0$$

$$g_11(x) = \frac{1.1 x_7 + 1.9}{x_5} - 1 \le 0$$

$$2.6 \le x_1 \le 3.6; \ 0.7 \le x_2 \le 0.8; \ 17 \le x_3 \le 28;$$
$$7.3 \le x_4, x_5 \le 8.3; \ 2.9 \le x_6 \le 3.9; \ 5.0 \le x_7 \le 5.5 \tag{29}$$

Table.11 shows the statistical results of each algorithm solving the speed reducer design problem. The best results are shown in bold. According to the results, TQA obtained the best optimal and average values. TQA found the best optimal value in six real-world engineering design problems, and a good average can also be achieved on all issues, demonstrating its satisfactory performance and robustness.

Table 11
Statistical results of speed reducer design problem

Function	GWO	MVO	SCA	SSA	WOA	STOA	TSA	TQA
Best	2997.725189	3007.555169	3050.017911	3002.667140	3007.949579	3011.906151	3017.913973	**2994.591855**
x_1	3.500056	3.500745	3.600000	3.503237	3.501340	3.502932	3.520683	3.500000
x_2	0.700000	0.700000	0.700000	0.700000	0.700026	0.700000	0.700000	0.700000
x_3	17.000000	17.000000	17.000000	17.000000	17.000572	17.000000	17.000000	17.000000
x_4	7.406517	7.772375	7.300000	7.796101	8.068027	7.515590	7.300000	7.311747
x_5	7.782102	8.028754	8.300000	7.816460	7.815870	8.023801	7.750438	7.715845
x_6	3.352452	3.355168	3.359315	3.351416	3.351722	3.373490	3.387144	3.350237
x_7	5.287056	5.287392	5.288406	5.286689	5.291945	5.289183	5.294580	5.286655
Mean	3005.500021	3036.968535	3102.237267	3037.269055	3119.455059	3028.135763	3034.726976	**3002.910419**
Std	4.6459E+00	1.6633E+01	2.8026E+01	2.4756E+01	6.9944E+01	9.4705E+00	9.9150E+00	5.6091E+00

5. Conclusions and future work

The existing meta-heuristic algorithms are generally divided into two types from the perspective of search mechanism. The first category is exploration first and exploitation later, that is, in the early stage of iteration, a relatively large step length is used for rough exploration, and in the late stage of iteration, a relatively small step length is used for accurate exploitation, such as GWO. This class of algorithms, by default, can find nearby regions of the global optimal solution in the upfront phase of the iteration and exploiting it later in the iteration. However, late the algorithm does not have the ability to jump out of that local optimal solution if it becomes enmeshed in the previous period. The other category is simultaneous exploration and exploitation, where search agents are divided into two parts, one part in a larger step size for coarse exploration behavior, and the other part in a smaller step length for precise exploitation behavior, such as SSA. This class of algorithms, although improving the precision of solution, may lead to a waste of computational power because a subset of search agents is used for exploration in the early iteration. We wish to put forward, from a search mechanism perspective, an algorithm that integrates the advantages of the two classes of algorithms mentioned above.

The search mechanism of TQA combines the advantages of the above two classes of algorithms. First, a local mapping principle is proposed to divide the search space. Then, the algorithm focus on the simultaneous exploration and exploitation on the region s_1 where the current global optimal solution is located. In addition, the algorithm uses the restart idea to exploit the region s_2 obtained by Levy flight. On the one hand, simultaneous exploration and exploitation on region s_1 can help the algorithm jump out of the local optimal solution on a certain Chengdu. On the other hand, the search behavior on region s_2 is both an exploration and a exploitation, which can reduce the waste of computing power to some extent.

We proposed TQA and evaluated its performance on 23 benchmark functions. The experimental results showed its satisfactory exploration and exploitation ability compared to many advanced algorithms. TQA was applied to several engineering design problems, including true-bar truss, gear-train, I-beam structure, weld beam, cantilever beam, and speed reducer. The results were compared with many emerging algorithms, showing that TQA has satisfactory performance in solving real-world engineering design problems. We hope to develop the binary versions in the future to solve TSP problems and DNA fragment assembly problems, and develop the multi-objectives versions to solve DNA design problems.

Acknowledgement

Conflict of interest statement

On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

Amoretti, M., 2014. Evolutionary strategies for ultra-large-scale autonomic systems. Inf. Sci. 274, 1–16. URL: https://doi.org/10.1016/j.ins.2014.03.003, doi:10.1016/j.ins.2014.03.003.

Askari, Q., et al., 2020. Political optimizer: A novel socio-inspired meta-heuristic for global optimization. Knowl. Based Syst. 195, 105709. URL: https://doi.org/10.1016/j.knosys.2020.105709, doi:10.1016/j.knosys.2020.105709.

Bojan-Dragos, C.A., et al., 2021. Gwo-based optimal tuning of type-1 and type-2 fuzzy controllers for electromagnetic actuated clutch systems. IFAC-PapersOnLine 54, 189–194.

Cai, J., et al., 2020. Dynamic shuffled frog-leaping algorithm for distributed hybrid flow shop scheduling with multiprocessor tasks. Eng. Appl. Artif. Intell. 90, 103540. URL: https://doi.org/10.1016/j.engappai.2020.103540, doi:10.1016/j.engappai.2020.103540.

Cao, B., et al., 2020. Designing uncorrelated address constrain for dna storage by dmvo algorithm. IEEE/ACM Transactions on Computational Biology and Bioinformatics , 1–1doi:10.1109/TCBB.2020.3011582.

Colorni, A., et al., 1991. Distributed optimization by ant colonies, in: Proceedings of the first European conference on artificial life, Paris, France. pp. 134–142.

Das, B., et al., 2020. Student psychology based optimization algorithm: A new population based optimization algorithm for solving optimization problems. Adv. Eng. Softw. 146, 102804. URL: https://doi.org/10.1016/j.advengsoft.2020.102804, doi:10.1016/j.advengsoft.2020.102804.

Derrac, J., et al., 2011. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. 1, 3–18. URL: https://doi.org/10.1016/j.swevo.2011.02.002, doi:10.1016/j.swevo.2011.02.002.

Dhiman, G., Kaur, A., 2019. STOA: A bio-inspired based optimization algorithm for industrial engineering problems. Eng. Appl. Artif. Intell. 82, 148–174. URL: https://doi.org/10.1016/j.engappai.2019.03.021, doi:10.1016/j.engappai.2019.03.021.

Eberhart, R., Kennedy, J., 1995. A new optimizer using particle swarm theory, in: MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science, pp. 39–43. doi:10.1109/MHS.1995.494215.

Elaziz, M.E.A., et al., 2021. A grunwald-letnikov based manta ray foraging optimizer for global optimization and image segmentation. Eng. Appl. Artif. Intell. 98, 104105. URL: https://doi.org/10.1016/j.engappai.2020.104105, doi:10.1016/j.engappai.2020.104105.

Etminaniesfahani, A., et al., 2018. Fibonacci indicator algorithm: A novel tool for complex optimization problems. Eng. Appl. Artif. Intell. 74, 1–9. URL: https://doi.org/10.1016/j.engappai.2018.04.012, doi:10.1016/j.engappai.2018.04.012.

Faramarzi, A., et al., 2020a. Equilibrium optimizer: A novel optimization algorithm. Knowl. Based Syst. 191, 105190. URL: https://doi.org/10.1016/j.knosys.2019.105190, doi:10.1016/j.knosys.2019.105190.

Faramarzi, A., et al., 2020b. Marine predators algorithm: A nature-inspired metaheuristic. Expert Syst. Appl. 152, 113377. URL: https://doi.org/10.1016/j.eswa.2020.113377, doi:10.1016/j.eswa.2020.113377.

Ghoumari, A., et al., 2018. Evolutionary algorithm with ensemble strategies based on maximum a posteriori for continuous optimization. Information Sciences 460, 1–22.

Hancer, E., 2020. A new multi-objective differential evolution approach for simultaneous clustering and feature selection. Eng. Appl. Artif. Intell. 87. URL: https://doi.org/10.1016/j.engappai.2019.103307, doi:10.1016/j.engappai.2019.103307.

Hayyolalam, V., Kazem, A.A.P., 2020. Black widow optimization algorithm: A novel meta-heuristic approach for solving engineering optimization problems. Eng. Appl. Artif. Intell. 87. URL: https://doi.org/10.1016/j.engappai.2019.103249, doi:10.1016/j.engappai.2019.103249.

Heidari, A.A., et al., 2019. Harris hawks optimization: Algorithm and applications. Future Gener. Comput. Syst. 97, 849–872. URL: https://doi.org/10.1016/j.future.2019.02.028, doi:10.1016/j.future.2019.02.028.

Holland, J.H., 1992. Genetic algorithms. Scientific American 267, 66–73. URL: http://www.jstor.org/stable/24939139.

Kamboj, V.K., et al., 2020. An intensify harris hawks optimizer for numerical and engineering optimization problems. Appl. Soft Comput. 89, 106018. URL: https://doi.org/10.1016/j.asoc.2019.106018, doi:10.1016/j.asoc.2019.106018.

Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. 39, 459–471. URL: https://doi.org/10.1007/s10898-007-9149-x, doi:10.1007/s10898-007-9149-x.

Karami, H., et al., 2021. Flow direction algorithm (FDA): A novel optimization approach for solving optimization problems. Comput. Ind. Eng. 156, 107224. URL: https://doi.org/10.1016/j.cie.2021.107224, doi:10.1016/j.cie.2021.107224.

Kaur, S., et al., 2020. Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Eng. Appl. Artif. Intell. 90, 103541. URL: https://doi.org/10.1016/j.engappai.2020.103541, doi:10.1016/j.engappai.2020.103541.

Koza, J., 1990. Genetically breeding populations of computer programs to solve problems in artificial intelligence, in: [1990] Proceedings of the 2nd International IEEE Conference on Tools for Artificial Intelligence, pp. 819–827. doi:10.1109/TAI.1990.130444.

Leboucher, C., et al., 2018. An enhanced particle swarm optimization method integrated with evolutionary game theory. IEEE Transactions on Games 10, 221–230.

Lee, K.S., Geem, Z.W., 2005. A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Computer Methods in Applied Mechanics and Engineering 194, 3902–3933. URL: https://www.sciencedirect.com/science/article/pii/S0045782504004682, doi:https://doi.org/10.1016/j.cma.2004.09.007.

Li, S., et al., 2020. Slime mould algorithm: A new method for stochastic optimization. Future Gener. Comput. Syst. 111, 300–323. URL: https://doi.org/10.1016/j.future.2020.03.055, doi:10.1016/j.future.2020.03.055.

Magdziarz, M., Szczotka, W., 2016. Quenched trap model for lévy flights. Commun. Nonlinear Sci. Numer. Simul. 30, 5–14. URL: https://doi.org/10.1016/j.cnsns.2015.05.027, doi:10.1016/j.cnsns.2015.05.027.

Mirjalili, S., 2015. Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. Knowl. Based Syst. 89, 228–249. URL: https://doi.org/10.1016/j.knosys.2015.07.006, doi:10.1016/j.knosys.2015.07.006.

Mirjalili, S., 2016a. Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput. Appl. 27, 1053–1073. URL: https://doi.org/10.1007/s00521-015-1920-1, doi:10.1007/s00521-015-1920-1.

Mirjalili, S., 2016b. SCA: A sine cosine algorithm for solving optimization problems. Knowl. Based Syst. 96, 120–133. URL: https://doi.org/10.1016/j.knosys.2015.12.022, doi:10.1016/j.knosys.2015.12.022.

Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67. URL: https://doi.org/10.1016/j.advengsoft.2016.01.008, doi:10.1016/j.advengsoft.2016.01.008.

Mirjalili, S., et al., 2014. Grey wolf optimizer. Adv. Eng. Softw. 69, 46–61. URL: https://doi.org/10.1016/j.advengsoft.2013.12.007, doi:10.1016/j.advengsoft.2013.12.007.

Mirjalili, S., et al., 2016. Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput. Appl. 27, 495–513. URL: https://doi.org/10.1007/s00521-015-1870-7, doi:10.1007/s00521-015-1870-7.

Mirjalili, S., et al., 2017. Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. Adv. Eng. Softw. 114, 163–191. URL: https://doi.org/10.1016/j.advengsoft.2017.07.002, doi:10.1016/j.advengsoft.2017.07.002.

Moghdani, R., Salimifard, K., 2018. Volleyball premier league algorithm. Appl. Soft Comput. 64, 161–185. URL: https://doi.org/10.1016/j.asoc.2017.11.043, doi:10.1016/j.asoc.2017.11.043.

Moosavi, S.H.S., Bardsiri, V.K., 2017. Satin bowerbird optimizer: A new optimization algorithm to optimize ANFIS for software development effort estimation. Eng. Appl. Artif. Intell. 60, 1–15. URL: https://doi.org/10.1016/j.engappai.2017.01.006, doi:10.1016/j.engappai.2017.01.006.

Pelusi, D., et al., 2020. Improving exploration and exploitation via a hyperbolic gravitational search algorithm. Knowl. Based Syst. 193, 105404. URL: https://doi.org/10.1016/j.knosys.2019.105404, doi:10.1016/j.knosys.2019.105404.

Precup, R.E., et al., 2021. Optimal tuning of interval type-2 fuzzy controllers for nonlinear servo systems using slime mould algorithm. International Journal of Systems Science , 1–16.

Punnathanam, V., Kotecha, P., 2016. Yin-yang-pair optimization: A novel lightweight optimization algorithm. Eng. Appl. Artif. Intell. 54, 62–79. URL: https://doi.org/10.1016/j.engappai.2016.04.004, doi:10.1016/j.engappai.2016.04.004.

Rao, R.V., et al., 2011. Teaching-learning-based optimization: A novel method for constrained mechanical design optimization problems. Comput. Aided Des. 43, 303–315. URL: https://doi.org/10.1016/j.cad.2010.12.015, doi:10.1016/j.cad.2010.12.015.

Rashedi, E., et al., 2009. GSA: A gravitational search algorithm. Inf. Sci. 179, 2232–2248. URL: https://doi.org/10.1016/j.ins.2009.03.004, doi:10.1016/j.ins.2009.03.004.

Saremi, S., et al., 2017. Grasshopper optimisation algorithm: Theory and application. Adv. Eng. Softw. 105, 30–47. URL: https://doi.org/10.1016/j.advengsoft.2017.01.004, doi:10.1016/j.advengsoft.2017.01.004.

Shadravan, S., et al., 2019. The sailfish optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. Eng. Appl. Artif. Intell. 80, 20–34. URL: https://doi.org/10.1016/j.engappai.2019.01.001, doi:10.1016/j.engappai.2019.01.001.

Shareef, H., et al., 2015. Lightning search algorithm. Appl. Soft Comput. 36, 315–333. URL: https://doi.org/10.1016/j.asoc.2015.07.028, doi:10.1016/j.asoc.2015.07.028.

Storn, R., Price, K.V., 1997. Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. 11, 341–359. URL: https://doi.org/10.1023/A:1008202821328, doi:10.1023/A:1008202821328.

Sulaiman, M.H., et al., 2020. Barnacles mating optimizer: A new bio-inspired algorithm for solving engineering optimization problems. Eng. Appl. Artif. Intell. 87. URL: https://doi.org/10.1016/j.engappai.2019.103330, doi:10.1016/j.engappai.2019.103330.

Sun, P., othersg, 2021. An intensify atom search optimization for engineering design problems. Applied Mathematical Modelling 89, 837–859. URL: https://www.sciencedirect.com/science/article/pii/S0307904X20304170, doi:https://doi.org/10.1016/j.apm.2020.07.052.

Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1, 67–82. URL: https://doi.org/10.1109/4235.585893, doi:10.1109/4235.585893.

Wu, J., et al., 2021. Enhancing physical and thermodynamic properties of dna storage sets with end-constraint. IEEE Transactions on NanoBioscience , 1–1doi:10.1109/TNB.2021.3121278.

Yang, X.S., 2010. Nature-inspired metaheuristic algorithms. Luniver press.

Yao, X., et al., 1999. Evolutionary programming made faster. IEEE Trans. Evol. Comput. 3, 82–102. URL: https://doi.org/10.1109/4235.771163, doi:10.1109/4235.771163.

Yin, Q., et al., 2021. Design of constraint coding sets for archive dna storage. IEEE/ACM Transactions on Computational Biology and Bioinformatics , 1–1doi:10.1109/TCBB.2021.3127271.

Zamfirache, I.A., et al., 2022. Reinforcement learning-based control using q-learning and gravitational search algorithm with experimental validation on a nonlinear servo system. Information Sciences 583, 99–120.

Zhang, Z., et al., 2019. A hybrid optimization algorithm based on cuckoo search and differential evolution for solving constrained engineering problems. Eng. Appl. Artif. Intell. 85, 254–268. URL: https://doi.org/10.1016/j.engappai.2019.06.017, doi:10.1016/j.engappai.2019.06.017.

Zhao, W., et al., 2020. Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. Eng. Appl. Artif. Intell. 87. URL: https://doi.org/10.1016/j.engappai.2019.103300, doi:10.1016/j.engappai.2019.103300.

Zhong, K., et al., 2021. Mompa: Multi-objective marine predator algorithm. Computer Methods in Applied Mechanics and Engineering 385, 114029. URL: https://www.sciencedirect.com/science/article/pii/S0045782521003601, doi:https://doi.org/10.1016/j.cma.2021.114029.