Computer Modeling in Engineering & Sciences

Tech Science Press

DOI: 10.32604/cmes.2022.019198

ARTICLE

An Improved Gorilla Troops Optimizer Based on Lens Opposition-Based Learning and Adaptive β-Hill Climbing for Global Optimization

Yaning Xiao, Xue Sun*, Yanling Guo, Sanping Li, Yapeng Zhang and Yangwei Wang

College of Mechanical and Electrical Engineering, Northeast Forestry University, Harbin, 150040, China

*Corresponding Author: Xue Sun. Email: xuesun@hit.edu.cn

Received: 08 September 2021 Accepted: 28 October 2021

ABSTRACT

Gorilla troops optimizer (GTO) is a newly developed meta-heuristic algorithm, which is inspired by the collective lifestyle and social intelligence of gorillas. Similar to other metaheuristics, the convergence accuracy and stability of GTO will deteriorate when the optimization problems to be solved become more complex and flexible. To overcome these defects and achieve better performance, this paper proposes an improved gorilla troops optimizer (IGTO). First, Circle chaotic mapping is introduced to initialize the positions of gorillas, which facilitates the population diversity and establishes a good foundation for global search. Then, in order to avoid getting trapped in the local optimum, the lens opposition-based learning mechanism is adopted to expand the search ranges. Besides, a novel local search-based algorithm, namely adaptive β-hill climbing, is amalgamated with GTO to increase the final solution precision. Attributed to three improvements, the exploration and exploitation capabilities of the basic GTO are greatly enhanced. The performance of the proposed algorithm is comprehensively evaluated and analyzed on 19 classical benchmark functions. The numerical and statistical results demonstrate that IGTO can provide better solution quality, local optimum avoidance, and robustness compared with the basic GTO and five other well-known algorithms. Moreover, the applicability of IGTO is further proved through resolving four engineering design problems and training multilayer perceptron. The experimental results suggest that IGTO exhibits remarkable competitive performance and promising prospects in real-world tasks.

KEYWORDS

Gorilla troops optimizer; circle chaotic mapping; lens opposition-based learning; adaptive β-hill climbing

1 Introduction

Optimization refers to the process of searching for the optimal solution to a particular issue under certain constraints, so as to maximize benefits, performance and productivity [1–4]. With the help of optimization techniques, a large number of problems encountered in different applied disciplines could be solved in a more efficient, accurate, and real-time way [5,6]. However, with the increasing complexity of global optimization problems nowadays, conventional mathematical methods based on gradient information are challenged by high-dimensional, suboptimal regions,

and large-scale search ranges that cannot adapt to the real requirements [7,8]. The development of more effective tools to settle these complex NP-hard problems is an indivisible research hotspot. Compared to traditional approaches, meta-heuristic algorithms (MAs) are often able to obtain the global best results on such problems, which is attributed to the merits of their simple structure, ease of implementation, as well as strong capability to bypass the local optimum [9,10]. As a result, during the past few decades, MAs have entered the blowout stage and received major attention from worldwide scholars [11–13].

MAs find out the optimal solution through the simulation of stochastic phenomena in nature. Based on the different design concepts, the nature-inspired MAs may be generally classified into four categories [14–16]: evolution-based, physical-based, swarm-based, and human-based algorithms. Specifically, evolutionary algorithms emulate the laws of Darwinian natural selection theory, and some well-regarded cases of which are Genetic Algorithm (GA) [17], Differential Evolution (DE) [18], and Biogeography-Based Optimization (BBO) [19]. Physical-based algorithms simulate the physical phenomenon of the universe such as Simulated Annealing (SA) [20], Multi-Verse Optimizer (MVO) [21], Thermal Exchange Optimization (TEO) [22], Atom Search Optimization (ASO) [23], and Equilibrium Optimizer (EO) [24], etc. Swarm-based algorithms primarily originate from the collective behaviours of social creatures. A remarkable embodiment of this category of algorithms is Particle Swarm Optimization (PSO) [25], which was first proposed in 1995 based on the foraging behaviour of birds. Ant Colony Optimization (ACO) [26], Chicken Swarm Optimization (CSO) [27], Dragonfly Algorithm (DA) [28], Whale Optimization Algorithm (WOA) [29], Spotted Hyena Optimizer (SHO) [30], Emperor Penguin Optimizer (EPO) [31], Seagull Optimization Algorithm (SOA) [32], Harris Hawks Optimization (HHO) [33], Tunicate Swarm Algorithm (TSA) [34], Sooty Tern Optimization Algorithm (STOA) [35], Slime Mould Algorithm (SMA) [36], Rat Swarm Optimizer (RSO) [37], and Aquila Optimizer (AO) [38] are also essential parts in this branch. The final type is influenced by human learning habits including Search Group Algorithm (SGA) [39], Soccer League Competition Algorithm (SLC) [40], and Teaching-Learning-Based Optimization (TLBO) [41].

With their own distinctive characteristics, these metaheuristics are commonly used in a variety of computing science fields, such as fault diagnosis [42], feature selection [43], engineering optimization [44], path planning [45], and parameters identification [46]. Nevertheless, it has been shown that the most basic algorithms still have the limitations of slow convergence, poor accuracy, and ease of getting trapped into the local optimum [7,15] in several applications. The non-free lunch (NFL) theorem indicates that there is no general algorithm that could be appropriate for all optimization tasks [47]. Hence, encouraged by this theorem, many scholars begin improving existing algorithms to generate higher-quality solutions from different aspects. Fan et al. [7] proposed an enhanced Equilibrium Optimizer (m-EO) algorithm based on reverse learning and novel updating mechanisms, which considerably increase its convergence speed and precision. Jia et al. [48] introduced the dynamic control parameter and mutation strategies into the Harris Hawks Optimization, and then proposed a novel method called DHHO/M to segment satellite images. Ding et al. [49] constructed an improved Whale Optimization Algorithm (LNAWOA) for continuous optimization, in which the nonlinear convergence factor is utilized to speed up the convergence. Besides, authors in [50] employed Lévy flight and crossover operation to further promote the robust and global exploration capability of the native Salp Swarm Algorithm. Recently, there is also an emerging trend to combine two prospective MAs to overcome the performance drawbacks of one single algorithm. For instance, Abdel-Basset et al. [51] incorporated Slime Mould Optimizer and Whale Optimization Algorithm into an efficient hybrid algorithm

(HSMAWOA) for image segmentation of chest X-ray to determine whether a person is infected with the COVID-19 virus. Fan et al. [9] proposed a new hybrid algorithm named ESSAWOA, which has been successfully applied to solve structural design problems. Moreover, Liu et al. [52] developed a hybrid imperialist competitive evolutionary algorithm and used it to find out the best portfolio solutions. Dhiman [53] constructed a hybrid bio-inspired Emperor Penguin and Salp Swarm Algorithm (ESA) for numerical optimization that effectively deals with different constraint problems in engineering optimization.

In this study, we focus on a novel swarm intelligent algorithm namely Gorilla Troops Optimizer (GTO), which was proposed by Abdollahzadeh et al. in 2021 [54]. The inspiration of GTO originates from the collective lifestyle and social intelligence of gorillas. Preliminary research indicates that GTO has excellent performances on benchmark function optimization. Nevertheless, similar to other meta-heuristic algorithms, it still suffers from low optimization accuracy, premature convergence, and the propensity to fall into the local optimum when solving complex optimization problems [55]. These defects are mainly associated with the poor quality of the initial population, lack of a proper balance between the exploration and exploitation, and low likelihood of large spatial leaps in the iteration process. Therefore, NFL theorem motivates us to improve this latest swarm-inspired algorithm.

In view of the above discussion, to enhance GTO for global optimization, an improved gorilla troops optimizer known as IGTO is developed in this paper by incorporating three improvements. Firstly, Circle chaotic mapping is utilized to replace the random initialization mode of GTO for enriching population diversity. Secondly, a novel lens opposition-based learning mechanism is adopted to boost the exploration capability of the algorithm, while avoiding falling into the local optimum. Additionally, adaptive β-hill climbing, a new local search algorithm is embedded into GTO to facilitate better solution accuracy and exploitation trends. The effectiveness of the proposed IGTO is comprehensively evaluated and investigated by a series of comparisons with the basic GTO and several state-of-the-art algorithms, including GWO, WOA, SSA, HHO, and SMA on 19 classical benchmark functions. The experimental results demonstrate that IGTO performs better than the other competitors in terms of solution quality, convergence accuracy and stability. In addition, to further validate its applicability, IGTO is applied to solve four engineering design problems and train multilayer perceptron. Our results reveal that the proposed method also has strong competitiveness and superiority in real-life applications.

The remainder of this paper is arranged as follows: the basic GTO algorithm is briefly described in Section 2. In Section 3, a detailed description of three improved mechanisms and the proposed IGTO is presented. In Section 4, the experimental results of benchmark function optimization are reported and discussed. Besides, the applicability of the IGTO for resolving practical engineering problems and training multilayer perceptron is highlighted and analyzed in Sections 5 and 6. Finally, the conclusion of this work and potential future work directions are given in Section 7.

2 Gorilla Troops Optimizer

Gorilla troops optimizer is a recently proposed nature-inspired and gradient-free optimization algorithm, which emulates the gorillas' lifestyle in the group [54]. The gorilla lives in a group called troop, composed of an adult male gorilla also known as the silverback, multiple adult female gorillas and their posterity. A silverback gorilla (shown in Fig. 1) typically has an age of more than 12 years and is named for the unique hair on his back at puberty. Besides, the silverback is the head of the whole troop, taking all decisions, mediating disputes, directing others

to food resources, determining group movements, and being responsible for safety. Younger male gorillas at the age of 8 to 12 years are called blackbacks since they still lack silver-coloured back hairs. They are affiliated with the silverback and act as backup defenders for the group. In general, both female and male gorillas tend to migrate from the group where they were born to a second new group. Alternatively, mature male gorillas are also likely to separate from their original group and constitute troops for their own by attracting migrating females. However, some male gorillas sometimes choose to stay in the initial troop and continue to follow the silverback. If the silverback dies, these males might engage in a brutal battle for dominance of the group and mating with adult females. Based on the above concept of gorillas group behaviour in nature, the specific mathematical model for the GTO algorithm is developed. As with other intelligent algorithms, GTO contains three main parts: initialization, global exploration, and local exploitation, which are explained thoroughly below.

Figure 1: Silverback gorilla [54]

2.1 Initialization Phase

Suppose there are N gorillas in the D-dimensional space. The position of the i-th gorilla in the space can be defined as $X_i = (x_{i,1}, x_{i,2}, \cdots, x_{i,D}), i = 1, 2, \cdots, N$. Thus, the initialization process of gorilla populations can be described as:

$$X_{N\times D} = rand(N, D) \times (ub - lb) + lb \tag{1}$$

where ub and lb are the upper and lower boundaries of the search range, respectively, and $rand(N, D)$ denotes the matrix with N rows and D columns, where each element is a random number between 0 and 1.

2.2 Exploration Phase

Once gorillas depart from their original troop, they will move to diverse environments in nature that they might or might not have ever seen before. In the GTO algorithm, all gorillas are considered as candidate solutions, and the optimal solution in each optimization process is deemed to be the silverback. In order to accurately simulate such natural behaviour of migration, the position update equation of the gorilla for the exploration stage was designed by employing three different approaches including migrating towards unknown positions, migrating around familiar locations, and moving to other groups, as shown in Eq. (2):

$$GX(t+1)=\begin{cases}(ub-lb)\times r_2+lb, & r_1<p\\ (r_3-C)\times X_A(t)+L\times Z\times X(t), & r_1\geq 0.5\\ X(t)-L\times(L\times(X(t)-X_B(t))+r_4\times(X(t)-X_B(t))), & r_1<0.5\end{cases} \tag{2}$$

where t indicates the current iteration times, $X(t)$ denotes the current position vector of the individual gorilla, and $GX(t+1)$ refers to the candidate position of search agents in the next iteration. Besides, r_1, r_2, r_3 and r_4 are all the random values between 0 and 1. $X_A(t)$ and $X_B(t)$ are two randomly selected gorilla positions in the current population. p is a constant. Z denotes a row vector in the problem dimension with values of the element are randomly generated in $[-C, C]$. And the parameter C is calculated according to Eq. (3).

$$C=(\cos(2\times r_5)+1)\times\left(1-\frac{t}{Maxiter}\right) \tag{3}$$

where $cos(\cdot)$ represents the cosine function, r_5 is a random number in the range of 0 to 1, and $Maxiter$ indicates the maximum iterations.

As for the parameter L in Eq. (2) could be computed as follows:

$$L=C\times l \tag{4}$$

where l is a random number in between $[-1, 1]$.

Upon the completion of the exploration, the fitness values of all newly generated candidate $GX(t+1)$ solutions are evaluated. Provided that GX is better than X i.e., $F(GX)<F(X)$, where $F(\cdot)$ denotes the fitness function for a certain problem, it will be retained and replace the original solution $X(t)$. In addition, the optimal solution at this period is selected as the silverback $X_{silverback}$.

2.3 Exploitation Phase

When the troop was just established, the silverback is powerful and healthy, while the others male gorillas are still young. They obey all the decisions of silverback in search of diverse food resources and serve the silverback gorilla faithfully. Undeniably speaking, the silverback also grows old and then finally dies, with younger blackbacks in the troop might get involved into a violent conflict with the other males for mating with the adult females and the leadership. As mentioned previously, two behaviours of following the silverback and competing for adult female gorillas are modelled in the exploitation phase of GTO. At the same time, the parameter W is introduced to control the switch between them. If the value C in Eq. (3) is greater than W, the first mechanism of following the silverback is elected. The mathematical expression is as follows:

$$GX(t+1)=L\times M\times(X(t)-X_{silverback})+X(t) \tag{5}$$

where L is also evaluated using Eq. (4), $X_{silverback}$ represents the best solution obtained so far, and $X(t)$ denotes the current position vector. In addition, the parameter M could be calculated according to Eq. (6):

$$M=\left(\left|\sum_{i=1}^{N}X_i(t)/N\right|^{2^L}\right)^{\frac{1}{2^L}} \tag{6}$$

where N refers to the population size, and $X_i(t)$ denotes each position vector of the gorilla in the current iteration.

If $C < W$, it implies that the latter mechanism is chosen, in this case, the location of gorillas can be updated as follows:

$$GX(t+1) = X_{silverback} - (X_{silverback} \times Q - X(t) \times Q) \times A \tag{7}$$

$$Q = 2 \times r_6 - 1 \tag{8}$$

$$A = \varphi \times E \tag{9}$$

$$E = \begin{cases} N_1, r_7 \geq 0.5 \\ N_2, r_7 < 0.5 \end{cases} \tag{10}$$

In Eq. (7), $X(t)$ denotes the current position and Q stands for the impact force, which is computed using Eq. (8). In Eq. (8), r_6 is a random value in the range of 0 to 1. Moreover, the coefficient A used to mimic the violence intensity in the competition is evaluated by Eq. (9), where φ denotes a constant and the values of E are assigned with Eq. (10). In Eq. (10), r_7 is also a random number in $[0, 1]$. If $r_7 \geq 0.5$, E would be defined as a 1-by-D array of normal distribution random numbers, and D is the spatial dimension. Instead, if $r_7 < 0.5$, E would be equal to a stochastic number that obeys the normal distribution.

Similarly, at the end of the exploitation process, the fitness values of the newly generated candidate $GX(t+1)$ solution are also calculated. If $F(GX) < F(X)$, the solution GX will be preserved and participate in the subsequent optimization, while the optimal solution within all individuals is defined as the silverback $X_{silverback}$. The pseudo-code of GTO is shown in Algorithm 1.

Algorithm 1: Gorilla troops optimizer

Initialize the population size N and the maximum number of iterations $Maxiter$
Initialize the random gorilla population $X_i (i = 1, 2, \cdots, N)$
Calculate the fitness values of all gorilla individuals
While $t < Maxiter$ **do**
  Update the parameter C according to Eq. (3)
  Update the parameter L according to Eq. (4)
  For each gorilla X_i **do** // Exploration stage
    Update the position of the current gorilla according to Eq. (2)
  End For
  Evaluate the fitness values of all gorillas
  Save the optimal solution as a silverback $X_{silverback}$
  For each gorilla X_i **do** // Exploitation stage
    If $C \geq W$ **then**
      Update the position of current gorilla according to Eq. (5)
    Else
      Update the position of current gorilla according to Eq. (7)
    End If
  End For

(Continued)

Algorithm 1 (Continued)

19: Update the fitness values of all gorillas
20: Update the global best solution $X_{silverback}$
21: $t = t + 1$
22: **End While**
23: **Output** the global best solution $X_{silverback}$ and its fitness value

3 The Proposed IGTO Algorithm

In order to further improve the performance of the basic GTO algorithm for global optimization, a novel variant named IGTO is presented in this section. First, Circle chaotic mapping is adopted to initialize the gorilla populations, which is considered from increasing the population diversity. Second, an effective lens opposition-based learning strategy is implemented to expand the search range and avoid the algorithm falling into the local optimum. Final, the modified algorithm is hybridized with the adaptive β-hill climbing for better exploitation trend and solution quality. The specific process is figured out as follows.

3.1 Circle Chaotic Mapping

It is indicated that the quality of the initial population individuals has a significant impact on the efficiency of most current metaheuristic algorithms [49,56]. When applying the GTO algorithm to tackle an optimization problem, the population is usually initialized by means of a stochastic search. Though this method is accessible to implement, yet it suffers from a lack of ergodicity and excessively depends on the probability distribution, which cannot guarantee that the initial population is uniformly distributed in the search space, thereby deteriorating the solution precision and convergence speed of the algorithm.

Chaotic mapping is a complex dynamic method found in nonlinear systems with the properties of unpredictability, randomness, and ergodicity. Compared to random distribution, chaotic mapping allows the initial population individual to explore the solution space thoroughly with a higher convergence speed and sensitivity so that it is widely adopted to improve the optimization performance of algorithms. Research results have proven that Circle chaotic mapping has superior exploration performance than the commonly used Logistic chaotic mapping and Tent chaotic mapping [57]. Consequently, in order to boost the population diversity and take full advantage of the information in the solution space, Circle chaotic mapping is introduced in this study to improve the initialization mode of the basic GTO. And the mathematical expression of Circle chaotic mapping is as follows:

$$z_{k+1} = z_k + b - \frac{a}{2\pi} \cdot \sin(2\pi z_k) \bmod(1), z_k \in (0, 1) \tag{11}$$

where $a = 0.5$ and $b = 0.2$. Under the same free independent parameters, the random search mechanism and Circle mapping are selected to be executed independently 300 times. Besides, the obtained results are shown in Fig. 2. It can be seen from the figure that the traversal of Circle chaotic mapping is wider and more homogeneously distributed in the feasible domain $[0, 1]$ than that of random search. Hence, the proposed algorithm has a more robust global exploration ability after incorporating Circle chaotic mapping.

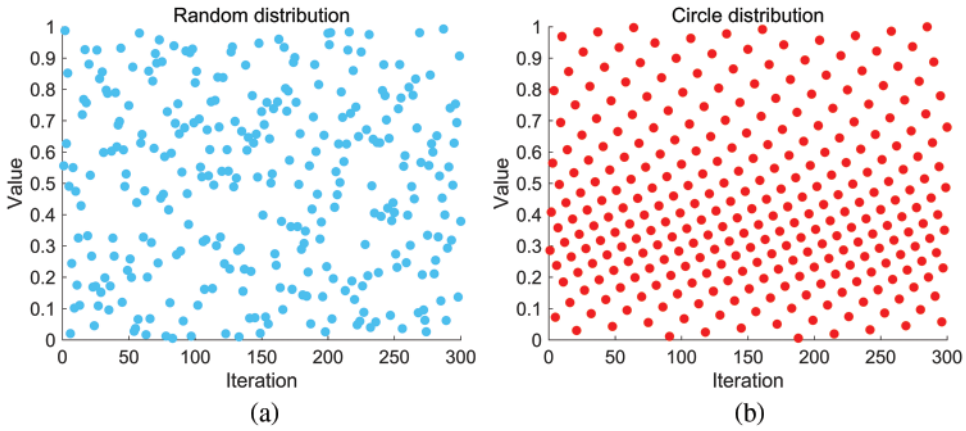

Figure 2: Distributions of random search and circle chaotic mapping. (a) Random distribution (b) Circle distribution

The pseudo-code for initializing the population using Circle chaotic mapping is outlined in Algorithm 2.

Algorithm 2: Circle chaotic mapping

Initialize the population size N and the dimension D
Randomly generate the initial value z_0 in the range $[0, 1]$
For $i = 1$ to N **do**
  For $k = 1$ to D **do**
    Generate the chaotic variable z_k according to Eq. (11)
    $S_{i,k} = z_k$
  End For
  Map the sequence S_i into the search interval of gorillas: $X_i = S_i \times (ub - lb) + lb$
End For
Return $X_i(i = 1, 2, \cdots, N)$ as the initialized population matrix

3.2 Lens Opposition-Based Learning

As a novel technique in the area of smart computing, lens opposition-based learning (LOBL), incorporating traditional opposition-based learning (OBL) [58] and convex lens imaging discipline, has been successfully employed in different intelligent algorithm optimizations [59,60]. Its basic ideology is to simultaneously calculate and compare the candidate solution and corresponding reverse solution, and then choose the superior one to proceed with the next iteration. Theoretically demonstrated by Fan et al. [9], LOBL can produce a solution close to the global optimum with higher possibility. Therefore, in this study, LOBL is utilized to update the candidate solutions during the exploration phase, in order to enlarge the search range and help the algorithm to escape from the local optimum. Several conceptions about LOBL are represented mathematically as follows.

Lens imaging is a physical optics phenomenon, which refers to the fact that while an object is located at more than two principal focal lengths away from the convex lens, a smaller and inverted image will be produced on the opposite side of the lens. Taking the one-dimensional search space

in Fig. 3 for illustration, there is a convex lens with the focal length f set at the base point O (the midpoint of search range $[lb, ub]$). Besides, an object p with the height h is placed on the coordinate axis, and its projection is GX (the candidate solution). Distance from the object to the lens u is greater than twice f. Through the lens imaging operation, an inverted imaging p' of height h^* could be attained, which is projected as GX^*(the reverse solution) on the x-axis. In accordance with the rules of lens imaging as well as similar triangle, the geometrical relationship obtained from Fig. 3 can be expressed as:

$$\frac{(lb+ub)/2 - GX}{GX^* - (lb+ub)/2} = \frac{h}{h^*} \tag{12}$$

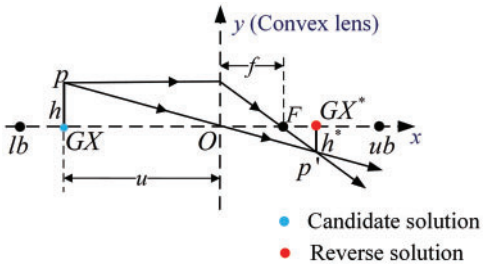

Figure 3: The principle of LOBL mechanism

Here, let the scale factor $n = h/h^*$, the reverse solution GX^* is calculated by transferring the Eq. (12):

$$GX^* = \frac{lb+ub}{2} + \frac{lb+ub}{2n} - \frac{GX}{n} \tag{13}$$

It is obvious that when $n = 1$, Eq. (13) can be simplified as the general formulation of OBL strategy:

$$GX^* = lb + ub - GX \tag{14}$$

So, we could regard the opposition-based learning strategy as a peculiar case of LOBL. In comparison to OBL, the latter allows acquiring dynamic reverse solutions and a wider search range by tuning the scale factor n.

Generally, Eq. (13) could be extended into D-dimensional space:

$$GX_j^* = (lb_j + ub_j)/2 + (lb_j + ub_j)/2n - GX_j/n \tag{15}$$

where lb_j and ub_j are the lower and upper limits of the j-th dimension, respectively, $j = 1, 2, \cdots D$, GX_j^* denotes the inverse solution of GX_j in the j-th dimension.

When a new inverse solution is generated, there is no guarantee that it is always better than the current candidate solution as in the gorilla position. Therefore, it is necessary to evaluate the

fitness values of the inverse solution and candidate solution, then the fitter one will be selected to continue participating in the subsequent exploitation phase, which is described as follows:

$$GX_{next} = \begin{cases} GX^*, & \text{if} F(GX^*) < F(GX) \\ GX, & \text{otherwise} \end{cases} \tag{16}$$

where GX^* indicates the reverse solution generated by LOBL, GX is the current candidate solution, GX_{next} is the selected gorilla to continue the subsequent position updating, and $F(\cdot)$ denotes the fitness function of the problem. The pseudo-code of lens opposition-based learning mechanism is shown in Algorithm 3.

Algorithm 3: Lens opposition-based learning

1: Input the current candidate solution of gorilla GX, the dimension D and scale factor n
2: **For** $j = 1$ to D **do**
3: Generate the reverse solution GX^* using Eq. (15)
4: **End For**
5: Calculate the fitness values of GX and GX^*
6: **If** $F(GX^*) < F(GX)$ **then**
7: $GX_{next} = GX^*$
8: **End If**
9: Output the new candidate solution GX_{next}

3.3 Adaptive β-Hill Climbing

Adaptive β-hill climbing (AβHC) [61] is a newly proposed local search-based algorithm, which is, in essence, a modified version of β-hill climbing (βHC). Research has found that AβHC provides better performance than many other famous local search algorithms, including Simulated Annealing (SA) [20], Tabu Search (TS) [62], and Variable Neighborhood Search (VNS) [63]. To boost the algorithm's exploitation ability and the quality of final solutions, AβHC is integrated into the basic GTO to help search the neighborhoods of the optimal solution in this study. And the definition of AβHC is represented mathematically as follows.

For the given current solution $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$, AβHC will iteratively generate an enhanced solution $X_i'' = (x_{i,1}'', x_{i,2}'', \ldots, x_{i,D}'')$ on the basis of two control operators: $\mathcal{N}$-operator and β-operator. The N-operator first transfers X_i to a new neighborhood solution $X_i' = (x_{i,1}', x_{i,2}', \ldots, x_{i,D}')$, which is defined in Eqs. (17) and (18) as:

$$x_{i,j}' = x_{i,j} \pm U(0,1) \times \mathcal{N}, j = 1, 2, \ldots, D \tag{17}$$

$$\mathcal{N}(t) = 1 - \frac{t^{\frac{1}{K}}}{Maxiter^{\frac{1}{K}}} \tag{18}$$

where $U(0,1)$ denotes a random number in the interval $[0,1]$, $x_{i,j}$ denotes the value of the decision variable in the j-th dimension, t denotes the current iteration, $Maxiter$ refers to the maximum number of iterations, N represents the bandwidth distance between the current solution and its neighbor, D is the spatial dimension, and the parameter K is a constant.

Immediately after, the decision variables of new solution $X_i^{''}$ are assigned either from the existing solution or randomly from the available range of β-operator, as follows:

$$x_{i,j}^{''} \leftarrow \begin{cases} x_{i,r}, & \text{if} r_8 < \beta \\ x_{i,j}^{'}, & \text{else} \end{cases} \tag{19}$$

$$\beta(t) = \beta_{\min} + (\beta_{\max} - \beta_{\min}) \times \frac{t}{Maxiter} \tag{20}$$

where r_8 is a random number in the interval [0, 1], $x_{i,r}$ denotes another random number chosen from the possible range of that particular dimension of the problem, $\beta_{\max}$ and $\beta_{\min}$ denote the maximum and minimum values of probability value $\beta \in [0, 1]$, respectively. If the generated solution $X_i^{''}$ is better than the current solution under consideration X_i, then X_i is replaced by $X_i^{''}$. The pseudo-code of adaptive β-hill climbing is given in Algorithm 4.

Algorithm 4: Adaptive β-hill climbing algorithm

Initialize the parameters $\beta_{\max}$, $\beta_{\min}$, and K
Input the current solution $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$
Calculate the fitness value $F(X_i)$
While $t \leq Maxiter$ **do**
  Generate the neighbouring solution X_i' using Eq. (17)
  For $j = 1$ to D **do**
    If $r_8 <$ **then**
      $x_{i,j}^{''} = x_{i,r}$
    Else
      $x_{i,j}^{''} = x_{i,j}^{'}$
    End If
  End For
  Calculate the fitness value $F(X_i^{''})$
  If $F(X_i^{''}) < F(X_i)$ **then**
    $X_i = X_i^{''}$
  End If
  $t = t + 1$
End while

3.4 Algorithm Flowchart

Based on the improved mechanisms mentioned in Subsections 3.1∼3.3 above, the flowchart of the proposed IGTO algorithm for global optimization problems is illustrated in Fig. 4. Moreover, Algorithm 5 outlines the pseudo-code of IGTO.

Algorithm 5: Improved gorilla troops optimizer

Initialize the population size N and the maximum number of iterations $Maxiter$
Initialize the chaotic gorilla population X_i using Circle chaotic mapping
Calculate the fitness values of all gorilla individuals
While $t < Maxiter$ **do**
  Update the parameter C according to Eq. (3)
  Update the parameter L according to Eq. (4)
  For each gorilla **do** //Exploration stage
    Update the position of the current gorilla according to Eq. (2)
    Calculate and evaluate the candidate position and its reverse position using LOBL strategy, then select the better one
  End For
  Evaluate the fitness values of all gorillas
  Set the best solution as a silverback $X_{silverback}$
  For each gorilla **do** //Exploitation stage
    If $C \geq W$ **then**
      Update the position of current gorilla according to Eq. (5)
    Else
      Update the position of current gorilla according to Eq. (7)
    End If
  End For
  Update the fitness values of all gorillas
  Update the global optimal solution $X_{silverback}$
  Perform AβHC strategy for the global optimal solution $X_{silverback}$
  $t = t + 1$
End While
Output the global best solution $X_{silverback}$ and its fitness value

4 Experimental Results and Discussion

In this section, a total of 19 benchmark functions from the literature [64] are selected for contrast experiments to comprehensively evaluate the feasibility and effectiveness of the proposed IGTO algorithm. First, the definitions of these benchmark functions, parameter settings, and measurements of algorithm performance are presented. Afterwards, the basic GTO and other five advanced meta-heuristic algorithms, such as GWO [65], WOA [29], SSA [66], HHO [33], and SMA [36], are employed as competitors to validate the improvements and superiority of the proposed algorithm based on the solution accuracy, boxplot, convergence behavior, average computation time, and statistical result. Final, the scalability of IGTO is investigated by solving high dimensional problems. All the simulation experiments are implemented in MATLAB R2014b with Microsoft Windows 7 system, and the hardware platform of the computer is configured as Intel(R) Core(TM) i5-7400 CPU @ 3.00 GHz, and 8 GB of RAM.

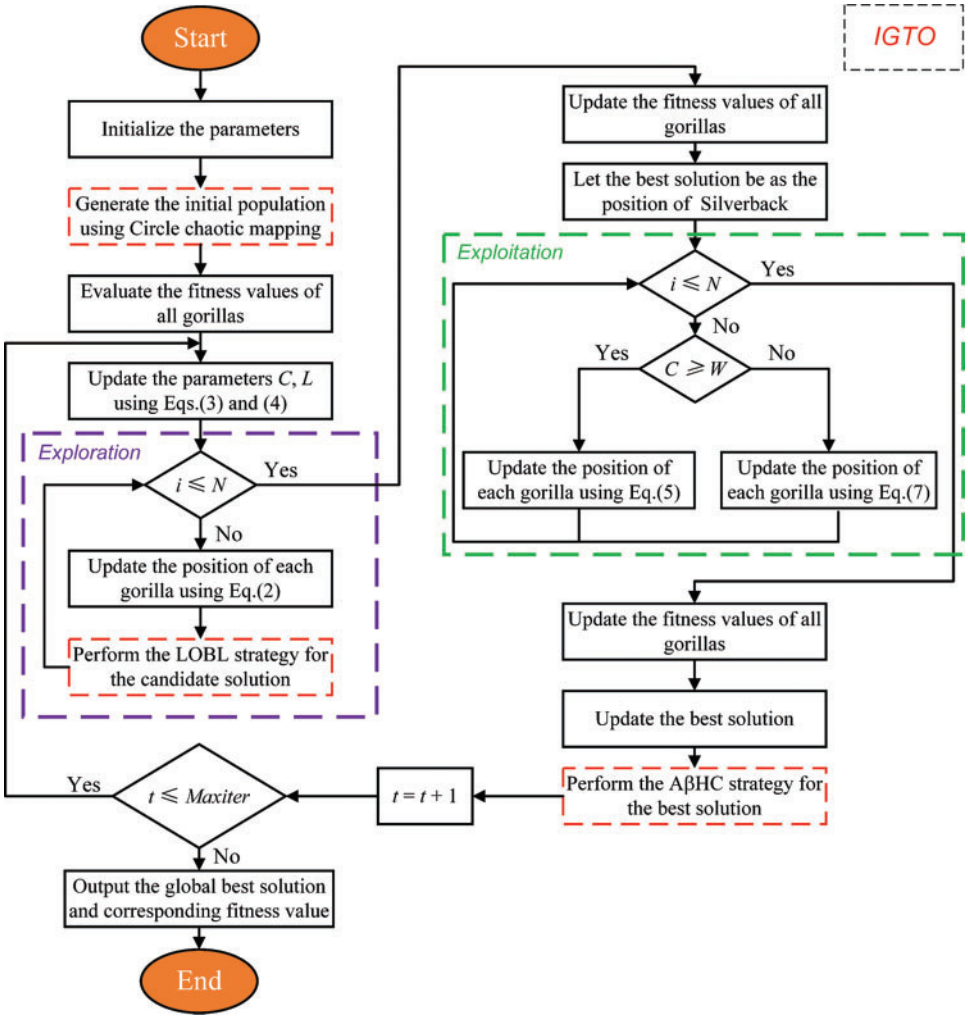

Figure 4: Flowchart of the proposed IGTO algorithm

4.1 Benchmark Function

The benchmark functions used in this paper could be divided into three various categories: unimodal (UM), multimodal (MM), and fix-dimension multimodal (FM). The unimodal functions (F_1~F_7) contain only one global minimum, which are frequently used to detect the development competence and convergence rate of algorithms. The multimodal functions (F_8~F_{13}), consisting of several local minima and a single global optimum in the search space, are well suited for assessing the algorithm's capability to explore and escape from local optima. The fix-dimension multimodal functions (F_{14}~F_{19}) are combinations of the previous two forms of functions, but with lower dimensions, and they are designed to evaluate the stability of the algorithm. Table 1 shows the formulations, spatial dimensions, search ranges, and theoretical minimum of these functions. In addition, 3D images for the search space of several typical functions are displayed in Fig. 5.

Table 1: Benchmark functions

Type	Function	Dim	Range	$F_{\min}$
UM	$F_1(x)=\sum_{i=1}^{n} x_i^2$	30	$[-100, 100]$	0
	$F_2(x)=\sum_{i=1}^{n} \lvert x_i\rvert+\prod_{i=1}^{n} \lvert x_i\rvert$	30	$[-10, 10]$	0
	$F_3(x)=\sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]$	0
	$F_4(x)=\max\{\lvert x_i\rvert, 1\le i\le n\}$	30	$[-100, 100]$	0
	$F_5(x)=\sum_{i=1}^{n-1} [100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	30	$[-30, 30]$	0
	$F_6(x)=\sum_{i=1}^{n} (\lvert x_i+0.5\rvert)^2$	30	$[-100, 100]$	0
	$F_7(x)=\sum_{i=1}^{n} ix_i^4+random[0,1]$	30	$[-1.28, 1.28]$	0
MM	$F_8(x)=\sum_{i=1}^{n} -x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)$	30	$[-500, 500]$	$-418.9829\times$ Dim
	$F_9(x)=\sum_{i=1}^{n} [x_i^2-10\cos(2\pi x_i)+10]$	30	$[-5.12, 5.12]$	0
	$F_{10}(x)=-20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})-$ $\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+\mathrm{e}$	30	$[-32, 32]$	0
	$F_{11}(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	30	$[-600, 600]$	0
	$F_{12}(x)=\frac{\pi}{n}\{10\sin(\pi y_1)+\sum_{i=1}^{n-1} (y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+$ $(y_n-1)^2\}+\sum_{i=1}^{n} u(x_i,10,100,4)$ $y_i=1+\frac{x_i+1}{4}, u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m x_i>a\\0-a<x_i<a\\k(-x_i-a)^m x_i<-a\end{cases}$	30	$[-50, 50]$	0
	$F_{13}(x)=0.1\{\sin^2(3\pi x_i)+\sum_{i=1}^{n} (x_i-1)^2[1+\sin^2(3\pi x_i+1)]+$ $\{(x_n-1)^2[1+\sin^2(2\pi x_n)]\}(x_n-1)^2[1+\sin^2(2\pi x_n)]\}+$ $\sum_{i=1}^{n} u(x_i,5,100,4)$	30	$[-50, 50]$	0

(Continued)

Table 1 (continued)

Type	Function	Dim	Range	$F_{\min}$
FM	$F_{14}(x) = (\frac{1}{500} + \sum_{j=1}^{25} (j + \sum_{i=1}^{n} (x_i - a_{ij})^6)^{-1})^{-1}$	2	$[-65, 65]$	0.998
	$F_{15}(x) = \sum_{i=1}^{11} [a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}]^2$	4	$[-5, 5]$	0.00030
	$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	-1.0316
	$F_{17}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{3} a_{ij}(x_j - p_{ij})^2)$	3	$[1, 3]$	-3.86
	$F_{18}(x) = -\sum_{i=1}^{4} c_i \exp(-\sum_{j=1}^{6} a_{ij}(x_j - p_{ij})^2)$	6	$[0, 1]$	-3.32
	$F_{19}(x) = -\sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	$[0, 10]$	-10.5363

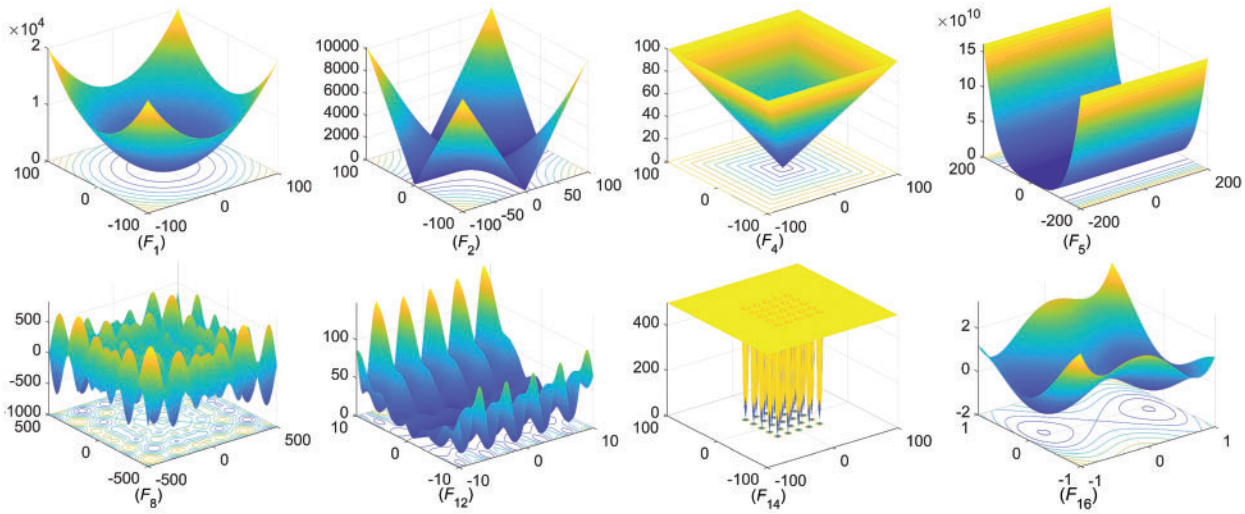

Figure 5: Search space of typical benchmark functions in 3D

4.2 Parameter Setting

In order to estimate the performance of the improved IGTO algorithm in solving global optimization problems, we select the basic GTO [54] and five state-of-the-art algorithms, namely GWO [65], WOA [29], SSA [66], HHO [33], and SMA [36]. For fair comparisons, the maximum iteration and population size of seven algorithms are set as 500 and 30, respectively. As per the references [9,61] and extensive trials, in the proposed IGTO algorithm, we set the scale factor $n = 12000, K = 30, \beta_{\max} = 1$ and $\beta_{\min} = 0.1$. Besides, all parameter values of the remaining six algorithms are set the same as those recommended in the original papers, as shown in Table 2. These parameter settings assure the fairness of the comparison experiments by allowing each algorithm to make the most of its optimization property. All algorithms are executed independently 30 times within each benchmark function to decrease accidental error.

Table 2: Parameter setting of the optimization algorithms

Reference	Algorithm	Parameter setting
[65]	GWO	Convergence constant $a = [0, 2]$
[29]	WOA	Convergence constant $a = [0, 2]$; Spiral factor $b = 1$
[66]	SSA	Coefficient $c_2, c_3 \in [0, 1]$
[33]	HHO	Random jump strength $J \in [0, 2]$
[36]	SMA	Constant $z = 0.03$
[54]	GTO	$\varphi = 3$; $W = 0.8$; $p = 0.03$
—	IGTO	$\varphi = 3$; $W = 0.8$; $p = 0.03$; $n = 12000$; $K = 30$; $\beta_{\max} = 1$; $\beta_{\min} = 0.1$

4.3 Evaluation Criteria of Performance

In this study, two metrics are used to measure the performance of the proposed algorithm including the average fitness value (Avg) and standard deviation (Std) of optimization results. The average fitness value intuitively characterizes the convergence accuracy and the search capability of the algorithm, which is calculated as follows:

$$\text{Avg} = \frac{1}{n}\sum_{i=1}^{n} S_i \tag{21}$$

where n denotes the times that an algorithm has run, and S_i indicates the obtained result of each operation.

And the standard deviation indicates the deviation degree between the experimental results and the average value. The equation of standard deviation is available as follows:

$$\text{Std} = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n} (S_i - \text{Avg})^2} \tag{22}$$

4.4 Comparison with IGTO and Other Algorithms

In this subsection, to examine the performance of the proposed algorithm, IGTO is compared with the basic GTO and five other advanced algorithms according to benchmark function optimization results. For fair comparisons, the maximum iteration and population size of seven algorithms are set as 500 and 30, respectively, and the rest parameter settings have been given in Subsection 4.2 above. Meanwhile, each algorithm runs 30 times independently on the test function F_1-F_{19} in Table 1 to decrease random error. The average fitness value (Avg) and standard deviation (Std) of each algorithm obtained from the experiment are reported in Table 3. In general, the closer the average fitness (Avg) to the theoretical minimum, the higher convergence accuracy of the algorithm. While the smaller the value of the standard deviation (Std), the better the stability and robustness of the algorithm.

As seen from Table 3, when solving the unimodal benchmark functions ($F_1 \sim F_7$), IGTO obtains the global optimal minima with regard to the average fitness on functions $F_1 \sim F_4$. For function F_5, the convergence accuracy of IGTO has a great improvement over its predecessors GTO and it is the winner among all algorithms. For test function F_6, the results of IGTO are similar to SSA and GTO, yet still marginally better them. Besides, IGTO shows superior results on function F_7 in contrast to other optimizers. In terms of standard deviation, the proposed IGTO

has excellent performance on all test problems. Given the properties of the unimodal functions, these results show that IGTO has outstanding search precision and local exploitation potential.

Table 3: Comparison results of different algorithms on 19 benchmark functions

Function	Criteria	GWO	WOA	SSA	HHO	SMA	GTO	IGTO
F_1	Avg	2.49E−27	1.32E−73	1.03E−07	2.81E−98	3.68E−275	**0.00E+00**	**0.00E+00**
	Std	6.42E−27	5.40E−73	6.64E−08	1.12E−97	0.00E+00	**0.00E+00**	**0.00E+00**
F_2	Avg	1.12E−16	1.42E−51	1.95E+00	3.62E−49	7.08E−149	3.69E−191	**0.00E+00**
	Std	7.66E−17	4.38E−51	1.23E+00	1.70E−48	3.88E−148	**0.00E+00**	**0.00E+00**
F_3	Avg	1.35E−05	4.60E+04	1.58E+03	2.55E−75	2.61E−318	**0.00E+00**	**0.00E+00**
	Std	2.66E−05	1.27E+04	8.20E+02	1.32E−74	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_4	Avg	8.38E−07	4.43E+01	1.09E+01	7.11E−50	2.19E−146	9.71E−193	**0.00E+00**
	Std	1.10E−06	2.92E+01	3.59E+00	2.50E−49	1.20E−145	**0.00E+00**	**0.00E+00**
F_5	Avg	2.75E+01	2.78E+01	1.15E+02	1.24E−02	1.17E+01	4.95E+00	**7.46E−05**
	Std	4.47E−01	6.10E−01	3.14E+01	1.81E−02	1.44E+01	1.11E+01	**8.57E−05**
F_6	Avg	8.69E−01	4.60E−01	2.44E−07	2.31E−04	5.95E−03	3.81E−07	**1.01E−07**
	Std	3.26E−01	2.15E−01	5.24E−07	3.21E−04	4.11E−03	5.37E−07	**1.15E−07**
F_7	Avg	1.97E−03	3.78E−03	1.57E−01	1.40E−04	1.84E−04	9.46E−05	**3.16E−05**
	Std	1.36E−03	4.83E−03	6.77E−02	1.11E−04	1.61E−04	6.91E−05	**2.47E−05**
F_8	Avg	−6189.550	−10334.835	−7512.488	−12410.223	−12569.055	−12568.778	**−12569.487**
	Std	7.97E+02	1.77E+03	8.87E+02	8.28E+02	3.51E−01	1.23E−04	**3.02E−05**
F_9	Avg	2.69E+00	**0.00E+00**	4.50E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	3.61E+00	**0.00E+00**	1.57E+01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{10}	Avg	1.02E−13	5.03E−15	2.45E+00	**8.88E−16**	**8.88E−16**	**8.88E−16**	**8.88E−16**
	Std	1.64E−14	2.30E−15	8.82E−01	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{11}	Avg	2.49E−03	6.44E−03	1.93E−02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	Std	5.96E−03	3.53E−02	1.54E−02	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{12}	Avg	4.54E−02	1.81E−02	7.45E+00	8.26E−06	5.18E−03	4.75E−08	**3.17E−08**
	Std	2.28E−02	7.45E−03	2.86E+00	1.06E−05	6.88E−03	1.24E−07	**5.09E−08**
F_{13}	Avg	6.21E−01	4.98E−01	1.36E+01	7.38E−05	5.71E−03	3.60E−03	**1.14E−07**
	Std	2.17E−01	2.90E−01	1.33E+01	1.32E−04	5.67E−03	6.49E−03	**3.34E−07**
F_{14}	Avg	3.35E+00	2.41E+00	1.10E+00	1.43E+00	**9.98E−01**	**9.98E−01**	**9.98E−01**
	Std	3.65E+00	2.54E+00	4.00E−01	1.261E+00	1.38E−12	6.54E−17	**4.12E−17**
F_{15}	Avg	4.42E−03	8.05E−04	1.86E−03	4.32E−04	4.58E−04	4.45E−04	**3.07E−04**
	Std	8.18E−03	5.85E−04	4.36E−03	3.04E−04	1.57E−04	3.35E−04	**4.06E−19**
F_{16}	Avg	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**	**−1.0316**
	Std	1.93E−08	6.89E−10	3.76E−14	4.27E−09	9.20E−10	6.45E−16	**1.32E−17**
F_{17}	Avg	−3.8615	−3.8594	−3.8628	**−3.8603**	−3.8628	−3.8628	−3.8610
	Std	2.46E−03	4.01E−03	3.96E−12	2.82E−03	9.39E−08	2.63E−15	**1.69E−18**
F_{18}	Avg	−3.2581	−3.2503	−3.2165	−3.1178	−3.2503	−3.2784	**−3.3101**
	Std	8.42E−02	8.41E−02	4.89E−02	1.21E−01	5.95E−02	5.83E−02	**3.63E−02**
F_{19}	Avg	−10.5341	−6.8840	−8.0801	−5.1232	−10.5360	−10.5360	**−10.5364**
	Std	1.14E−03	3.56E+00	3.56E+00	5.95E−03	4.02E−04	3.23E−15	**1.09E−15**

Note: The best result obtained is highlighted in bold.

The multimodal benchmark functions ($F_8 \sim F_{13}$) have many local minima in the search space, so these functions are usually employed to analyze the algorithm's potential to avoid the local optima. For functions F_8, F_{12} and F_{13}, the average fitness and standard deviation of IGTO are obviously better than the rest of the algorithms. For function F_9, IGTO obtains the same global

optimal minima as WOA, HHO, SMA, GTO. Moreover, HHO, SMA, GTO and IGTO obtains the same performance on functions F_{10} and F_{11}. It hopefully validates that the proposed IGTO can effectively bypass the local optimum and find high quality solutions.

The fix-dimension multimodal functions (F_{14}~F_{19}) consist of few local optima, which are designed to evaluate the stability of the algorithm in switching between exploration and exploitation processes. As far as the average fitness values are concerned, IGTO performs the same as SMA and GTO on function F_{14}, albeit better than others. For functions F_{15}, F_{18} and F_{19}, IGTO can generate superior results to all competitors. For function F_{16}, the performance of seven optimizers is identical. Although the result of the proposed IGTO is worse than HHO on function F_{17}, it still ranks second and shows significant improvements over the basic GTO to a certain extent. On the other hand, IGTO achieves the optimal standard deviation on all test cases. This proves that our proposed IGTO is able to keep a better balance between exploration and exploitation.

In view of the above, a summary can be drawn that the proposed multi-strategy combination IGTO algorithm exhibits strong global search capability and is superior to the other six intelligent algorithms in comparison. Benefiting from the hybrid AβHC with GTO operation, the solution precision of IGTO is greatly strengthened. At the same time, LOBL strategy is effective to expand the unknown search area and avoid the algorithm falling into the local optima.

In order to better illustrate the stability of the proposed algorithm, the corresponding boxplots of functions 1, 2, 3, 5 and 6 from UM benchmark functions, functions 9, 10 and 12 from MM benchmark functions, and function 15 selected from FM benchmark functions are shown in Fig. 6. From Fig. 6, it can be seen that IGTO algorithm shows remarkable consistency in most issues with respect to the median, maximum and minimum values compared with the others. In addition, IGTO generates no outliers during the iterations with the more concentrated distribution of convergence values, thereby verifying the strong robustness and superiority of the improved IGTO.

Fig. 7 visualizes the convergence curves of different algorithms on nine representative benchmark functions. Likewise, where functions 1, 2, 3, 5 and 6 are unimodal, functions 9, 10 and 12 are multimodal, and function 15 belongs to the fix-dimension multimodal category. From Fig. 7, it is clear that the convergence speed of IGTO is the fastest among all algorithms on functions F_1~F_3, and the proposed algorithm can rapidly reach the global optimal solution at the beginning of the search process. For functions F_5 and F_6, IGTO has a similar trend to HHO and GTO in the initial stage, but its efficiency is fully demonstrated in the late iterations, and eventually the proposed IGTO obtains the best result. For functions F_9, IGTO remains a superior convergence rate and obtains the global optimum within 20 iterations. Although the convergence accuracy of IGTO is the same as that of HHO, SMA and the basic GTO on functions F_{10}, yet it converges more quickly. For function F_{12}, the proposed algorithm is still the champion compared with the remaining six optimizers in terms of final accuracy and speed. Besides, the convergence curves of seven algorithms are pretty close on the fix-dimension multimodal function F_{15}. However, the performance of IGTO is slightly better than the others.

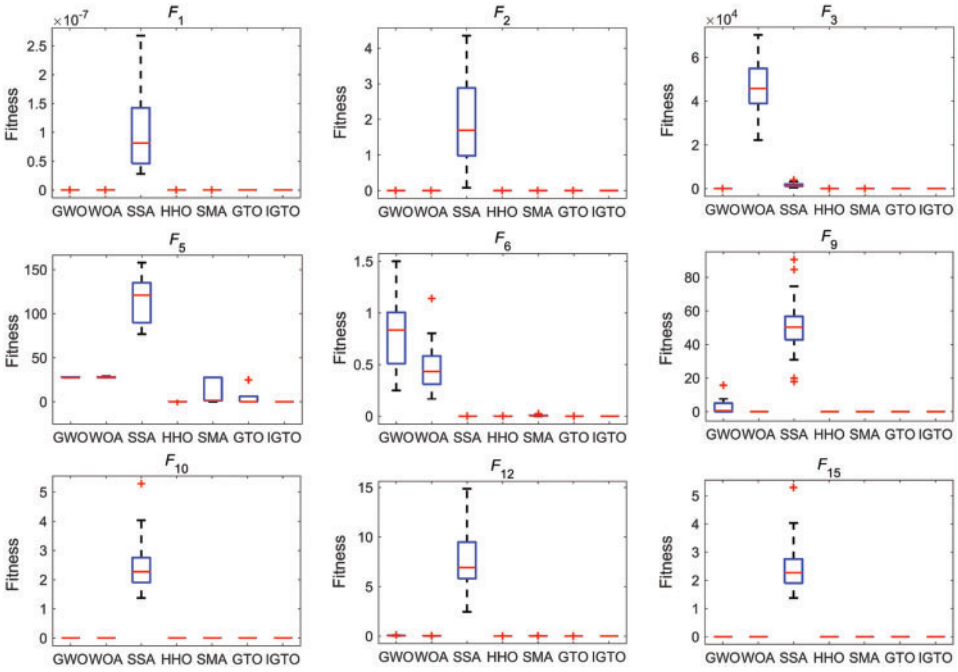

Figure 6: Boxplot analysis of different algorithms on partial benchmark functions

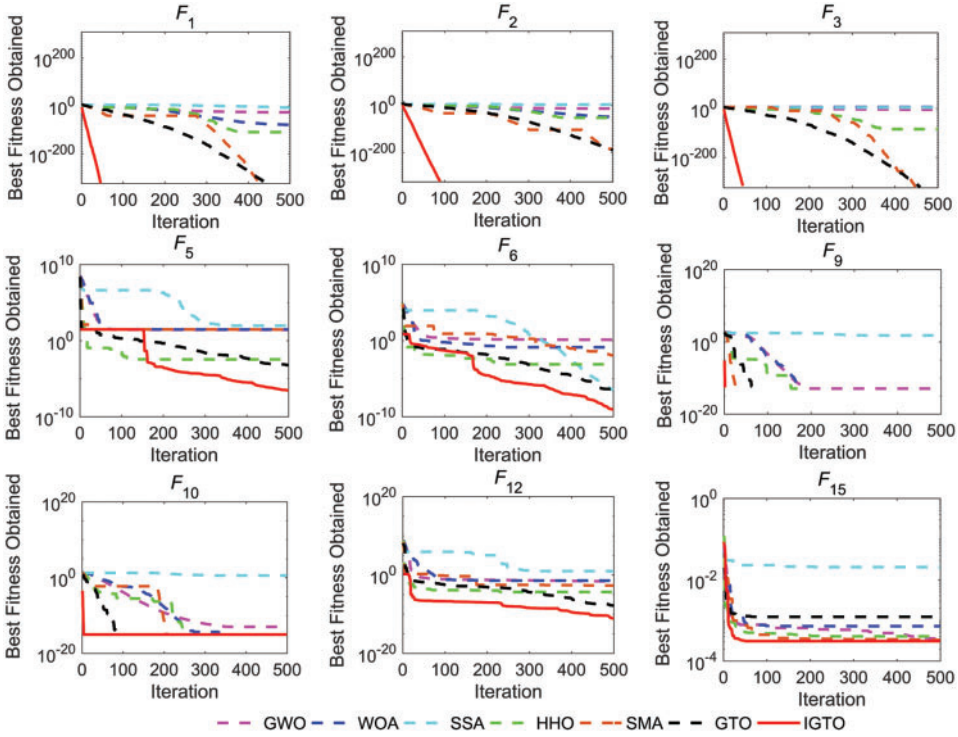

Figure 7: Convergence curves of different algorithms on nine benchmark functions

On the basis of experimental results of boxplot analysis and convergence curves, IGTO has a considerable enhancement in convergence speed and stability compared with the basic GTO, which is owed to the good foundation of global search laid by Circle chaotic mapping and LOBL strategy.

The average computation time spent by each algorithm on test functions $F_1 \sim F_{19}$ is reported in Table 4. For a more intuitive conclusion, the total runtime of each method is calculated and ranked as follows: SMA(14.118 s)>IGTO(8.073 s)>GTO(6.690 s)>HHO(6.568 s)>GWO(4.912 s)>SSA(4.897 s)>WOA(4.065 s). It can be found that IGTO uses more computation time than GTO, which is the second to last. Compared with the basic GTO algorithm, the introduction of three improved strategies increases the steps of the algorithm and extra time. Of course, the high computation cost of GTO algorithm itself is also a primary cause of this. However, the IGTO takes less time than SMA on most test functions. To improve the solution accuracy, a little more runtime is sacrificed. On the whole, the proposed algorithm is acceptable in view of the optimal search performance, and its limitation is still the need to decrease the computational time.

Table 4: Comparison results of the average computation time of different algorithms (unit: s)

Function	GWO	WOA	SSA	HHO	SMA	GTO	IGTO
F_1	1.66E−01	**1.53E−01**	1.79E−01	1.82E−01	8.34E−01	2.05E−01	3.09E−01
F_2	1.73E−01	**1.50E−01**	1.27E−01	1.60E−01	8.08E−01	2.15E−01	2.90E−01
F_3	4.75E−01	**4.28E−01**	5.89E−01	8.03E−01	1.09E+00	6.14E−01	7.06E−01
F_4	1.78E−01	**1.45E−01**	2.10E−01	1.85E−01	8.39E−01	2.02E−01	2.77E−01
F_5	1.88E−01	**1.50E−01**	1.34E−01	2.60E−01	8.51E−01	2.38E−01	3.04E−01
F_6	2.17E−01	**1.70E−01**	1.53E−01	2.66E−01	8.18E−01	2.40E−01	3.18E−01
F_7	1.99E−01	**1.23E−01**	1.77E−01	3.03E−01	8.71E−01	2.97E−01	3.69E−01
F_8	1.58E−01	**1.48E−01**	1.94E−01	2.69E−01	8.73E−01	2.90E−01	3.52E−01
F_9	1.53E−01	**1.30E−01**	1.95E−01	2.38E−01	8.29E−01	2.56E−01	2.99E−01
F_{10}	1.90E−01	**1.57E−01**	1.31E−01	2.35E−01	8.57E−01	2.96E−01	3.60E−01
F_{11}	1.45E−01	**1.38E−01**	1.70E−01	2.83E−01	8.93E−01	3.08E−01	3.51E−01
F_{12}	4.96E−01	**3.80E−01**	4.01E−01	6.30E−01	1.01E+00	6.37E−01	7.05E−01
F_{13}	4.94E−01	**3.52E−01**	4.34E−01	6.48E−01	1.00E+00	6.39E−01	7.10E−01
F_{14}	6.92E−01	**6.34E−01**	6.77E−01	7.85E−01	7.57E−01	8.27E−01	9.22E−01
F_{15}	1.72E−01	**1.24E−01**	2.07E−01	1.91E−01	3.18E−01	2.23E−01	3.03E−01
F_{16}	1.54E−01	**1.31E−01**	1.90E−01	2.03E−01	2.90E−01	2.02E−01	2.71E−01
F_{17}	2.00E−01	**1.73E−01**	2.31E−01	2.72E−01	3.25E−01	2.79E−01	3.47E−01
F_{18}	2.24E−01	**1.87E−01**	2.34E−01	2.48E−01	3.80E−01	2.76E−01	3.61E−01
F_{19}	2.38E−01	**1.92E−01**	2.64E−01	4.07E−01	4.75E−01	4.46E−01	5.19E−01

Note: The best result obtained is highlighted in bold.

Moreover, since the average fitness (Avg) and standard deviation (Std) of the algorithm after 30 runs are not compared with the results of each run, it is often not accurate to evaluate the performance of an algorithm based only on the mean and standard deviation. To represent the robustness and fairness of the improved algorithm, the Wilcoxon rank-sum test [67], a nonparametric statistical test approach is used to estimate the significant differences between IGTO and other algorithms. For Wilcoxon rank-sum test, the significance level is set to 0.05 and acquired

p-values are listed in Table 5. In this table, the sign "+" denotes that IGTO performs significantly better than the corresponding algorithm, "=" denotes that the performance of IGTO is analogous to that of the compared algorithm, "-" denotes that IGTO is poorer than the compared one, and the last line counts the total number of all signs. It can be seen from the table that for the 19 benchmark test functions, the proposed IGTO algorithm outperforms GWO on 19 functions, WOA and SSA on 18 functions, HHO on 16 functions, SMA on 14 functions, and the basic GTO on 13 functions, respectively. Therefore, according to the statistical theory analysis, our proposed IGTO has a significant enhancement over the other algorithms and it is the optimal optimizer among them.

Table 5: Statistical results of IGTO on Wilcoxon rank-sum test

Function	IGTO *vs.* GWO		IGTO *vs.* WOA		IGTO *vs.* SSA		IGTO *vs.* HHO		IGTO *vs.* SMA		IGTO *vs.* GTO	
	p-value	win	*p*-value	win	*p*-value	win	*p*-value	win	*p*-value	win	*p*-value	win
F_1	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.61E−01	-	NaN	=
F_2	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.21E−12	+
F_3	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.61E−01	-	NaN	=
F_4	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.21E−12	+	1.21E−12	+
F_5	3.02E−11	+	3.02E−11	+	6.70E−11	+	4.21E−02	+	1.04E−04	+	2.56E−02	+
F_6	3.02E−11	+	3.02E−11	+	1.00E+00	−	3.02E−11	+	3.02E−11	+	1.05E−02	+
F_7	3.02E−11	+	3.34E−11	+	3.02E−11	+	5.53E−08	+	4.44E−07	+	4.35E−05	+
F_8	3.02E−11	+	3.02E−11	+	3.02E−11	+	3.02E−11	+	2.15E−10	+	9.35E−04	+
F_9	1.19E−12	+	NaN	=	1.21E−12	+	NaN	=	NaN	=	NaN	=
F_{10}	1.12E−12	+	9.79E−11	+	1.21E−12	+	NaN	=	NaN	=	NaN	=
F_{11}	2.16E−02	+	3.34E−04	+	1.21E−12	+	NaN	=	NaN	=	NaN	=
F_{12}	3.02E−11	+	3.02E−11	+	3.02E−11	+	1.46E−10	+	3.02E−11	+	3.79E−08	+
F_{13}	3.02E−11	+	3.02E−11	+	3.02E−11	+	6.07E−11	+	3.02E−11	+	1.61E−06	+
F_{14}	1.72E−12	+	1.72E−12	+	1.45E−12	+	1.72E−12	+	1.72E−12	+	5.57E−09	+
F_{15}	6.67E−08	+	6.67E−08	+	6.67E−08	+	6.67E−08	+	6.67E−08	+	1.54E−01	-
F_{16}	7.57E−12	+	7.57E−12	+	7.56E−12	+	7.57E−12	+	7.57E−12	+	5.70E−04	+
F_{17}	5.20E−12	+	5.20E−12	+	5.20E−12	+	5.20E−12	+	5.20E−12	+	1.96E−08	+
F_{18}	2.00E−09	+	2.00E−09	+	4.43E−11	+	8.14E−11	+	3.60E−10	+	6.25E−08	+
F_{19}	1.25E−11	+	1.25E−11	+	1.25E−11	+	1.25E−11	+	1.25E−11	+	5.35E−03	+
+/=/−	19/0/0		18/1/0		18/0/1		16/3/0		14/3/2		13/5/1	

Lastly, the mean absolute error (MAE) of all algorithms on 19 benchmark problems is evaluated and ranked. MAE is also a useful statistical tool to reveal the gap between the experimental results and the theoretical values [1], and its mathematical expression is as follows:

$$\mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|a_i - o_i| \tag{23}$$

In Eq. (23), N is the number of benchmark functions used, o_i represents the desired value of each test function, and a_i is the actual value obtained. The MAE and relative rankings of each algorithm are reported in Table 6. From this table, it is obvious that IGTO outperforms all competitors and the MAE of IGTO is reduced by 2 orders of magnitude compared to GTO, which once again demonstrates the superiority of the proposed algorithm statistically.

Table 6: Ranking of different algorithms by MAE on 19 benchmark functions

Algorithm	MAE	Rank
GWO	3.3758E+02	5
WOA	2.5428E+03	7
SSA	3.5980E+02	6
HHO	8.7013E+00	4
SMA	6.4326E−01	3
GTO	3.0040E−01	2
IGTO	**1.4797E−03**	**1**

Note: The best result obtained is highlighted in bold.

4.5 Scalability Test

Scalability reflects the execution efficiency of an algorithm in different dimensional spaces. As the dimensions of the optimization problem increase, most current intelligent algorithms are highly prone to be ineffective and subject to "dimensional disaster". To investigate the scalability of IGTO, the proposed algorithm is utilized to optimize 13 benchmark functions $F_1 \sim F_{13}$ in Table 1 with higher dimensions (i.e., 50, 100 and 200 dimensions). The average fitness values (Avg) obtained by the basic GTO and IGTO on each function are reported in Table 7. From the data in the table, it is clear that the convergence accuracy of both algorithms gradually decreases with the increase in dimensions, which is due to the fact that the larger the dimensions, the more elements an algorithm needs to optimize. However, the experimental results of IGTO are consistently superior to GTO on functions $F_1 \sim F_8$, F_{12} and F_{13}, and the disparity in optimization performance between them is increasingly obvious as the dimension increases. Besides, it is notable that the proposed IGTO can always obtain the theoretical optimal solution on functions $F_1 \sim F_4$. For functions $F_9 \sim F_{11}$, two algorithms obtain the same performance.

Table 7: Fitness values of GTO and IGTO in 50, 100, 200 dimensions on 13 test functions

Function	50		100		200	
	GTO	IGTO	GTO	IGTO	GTO	IGTO
F_1	5.39E−322	**0.00E+00**	1.77E−319	**0.00E+00**	2.88E−304	**0.00E+00**
F_2	2.48E−154	**0.00E+00**	4.39E−152	**0.00E+00**	6.60E−136	**0.00E+00**
F_3	1.05E−296	**0.00E+00**	8.78E−239	**0.00E+00**	6.21E−221	**0.00E+00**
F_4	4.80E−156	**0.00E+00**	3.34E−128	**0.00E+00**	1.92E−110	**0.00E+00**
F_5	4.52E+00	**5.96E−03**	1.58E+01	**6.39E−02**	1.95E+01	**5.09E−02**
F_6	3.35E−04	**2.21E−04**	6.19E−02	**8.23E−04**	1.08E−01	**6.53E−04**
F_7	3.37E−02	**1.44E−05**	6.30E−01	**3.20E−05**	5.36E−01	**4.34E−04**
F_8	−20948	**−20949**	−41897	**−41898**	−83794	**−83796**

(Continued)

Table 7 (continued)

Function	50		100		200	
	GTO	IGTO	GTO	IGTO	GTO	IGTO
F_9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{10}	**8.88E−16**	**8.88E−16**	**8.88E−16**	**8.88E−16**	**8.88E−16**	**8.88E−16**
F_{11}	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
F_{12}	3.18E−06	**8.82E−07**	2.42E−04	**4.91E−06**	1.99E−03	**4.91E−05**
F_{13}	1.18E−03	**1.98E−06**	5.43E−03	**2.45E−05**	9.43E−02	**3.61E−05**

Note: The best result obtained is highlighted in bold.

The overall results fully prove that IGTO is not only able to solve low-dimensional functions at ease, but also maintain good scalability in high-dimensional functions, that is to say, the performance of IGTO does not deteriorate significantly when tackling high-dimensional problems, and it can still provide high-quality solutions effectively with well exploitation and exploration capabilities.

5 IGTO for Solving Engineering Design Problems

In this section, the applicability of the proposed IGTO is tested by solving four practical engineering design problems including pressure vessel design problem, gear train design problem, welded beam design problem and rolling element bearing design problem. For the sake of convenience, the death penalty [68] function is used here to handle the infeasible solutions subjected to constraints. IGTO runs independently 30 times for each issue, with the maximum iterations and population size are set to 500 and 30, respectively. At last, the obtained results are compared against those of different advanced meta-heuristic algorithms in the literature, as well as the corresponding analysis are presented.

5.1 Pressure Vessel Design

The pressure vessel design problem was first purposed by Kannan et al. [69], the purpose of which is to minimize the overall fabrication cost of a pressure vessel. There are four decision variables involved in this optimum design: T_s (z_1, thickness of the shell), T_h (z_2, thickness of the head), R (z_3, inner radius), and L (z_4, length of the cylindrical portion). Fig. 8 illustrates the structure of pressure vessel used in this study and its related mathematical model can be defined as follows:

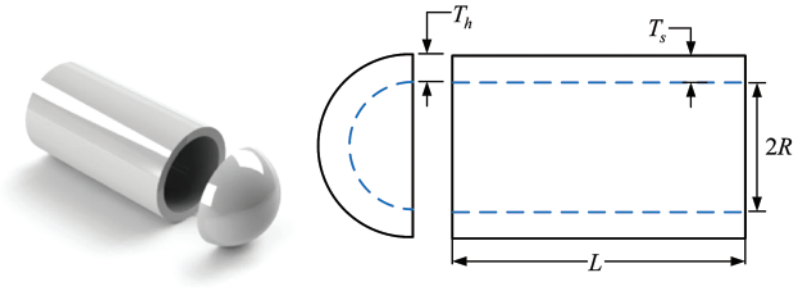

Figure 8: Pressure vessel design problem

consider

$\overrightarrow{z} = [z_1 z_2 z_3 z_4] = [T_s T_h R L]$

minimize

$f(\overrightarrow{z}) = 0.6224 z_1 z_3 z_4 + 1.7781 z_2 z_3^2 + 3.1661 z_1^2 z_4 + 19.84 z_1^2 z_3$

subject to

$g_1(\overrightarrow{z}) = -z_1 + 0.0193 z_3 \le 0$
$g_2(\overrightarrow{z}) = -z_3 + 0.00954 z_3 \le 0$
$g_3(\overrightarrow{z}) = -\pi z_3^2 z_4 - \frac{4}{3}\pi z_3^3 + 1296000 \le 0$
$g_4(\overrightarrow{z}) = z_4 - 240 \le 0$

variable range: $0 \le z_1 \le 99, 0 \le z_2 \le 99, 10 \le z_3 \le 200, 10 \le z_4 \le 200$

The experimental results of IGTO for this problem are compared against those resolved by GTO, SMA [36], HHO [33], AOA [4], SSA, WOA [29], and GWO [65], as shown in Table 8. It is shown that IGTO can provide the best design among all algorithms, and the minimum cost obtained is $f(\vec{z})_{\min} = 5904.2189$, which corresponds to the optimum solution $\vec{z} = [0.7889\ 0.3900\ 40.8764\ 192.4031]$. Thus, the proposed IGTO algorithm is regarded as more suitable for solving such problem.

Table 8: Comparison results of pressure vessel design

Algorithms	Optimum variables				Minimum cost
	$T_s(z_1)$	$T_h(z_2)$	$R(z_3)$	$L(z_4)$	
IGTO	0.7889	0.3900	40.8764	192.4031	**5904.2189**
GTO	0.7785	0.3849	40.8908	197.8512	5935.7284
SMA [36]	0.7931	0.3932	40.6711	196.2178	5994.1857
HHO [33]	0.8176	0.4073	42.0917	176.7196	6000.4626
AOA [4]	0.8304	0.4162	42.7513	169.3454	6048.7844
SSA	0.7907	0.3908	40.9677	195.9182	6012.1885
WOA [29]	0.8125	0.4375	42.0987	176.6390	6059.7410
GWO [65]	0.8125	0.4345	42.0892	176.7587	6051.5639

Note: The best solution obtained is highlighted in bold.

5.2 Gear Train Design

This is a classical mechanical engineering problem developed by Sandgren [70]. Fig. 9 shows the schematic view of the gear train. As its name suggests, the ultimate aim of this problem is to find four optimal parameters that minimize the gear ratio ($\frac{z_2 z_3}{z_1 z_4}$) as much as possible. The test case can be also represented mathematically as follows:

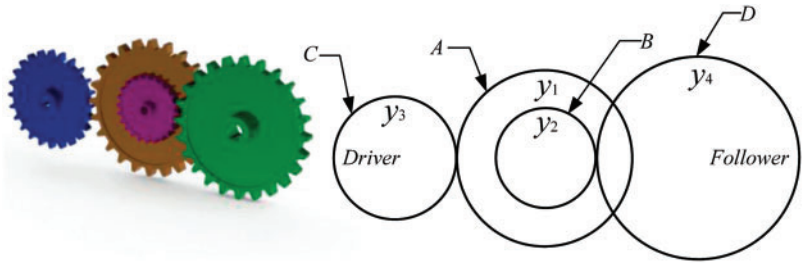

Figure 9: Gear train design problem

consider

$$\vec{z} = [z_1\ z_2\ z_3\ z_4] = [n_A\ n_B\ n_C\ n_D]$$

minimize

$$f(\vec{z}) = \left(\frac{1}{6.931} - \frac{z_2 z_3}{z_1 z_4}\right)^2$$

variable range: $12 \le z_1, z_2, z_3, z_4 \le 60$

Table 9 reports the detailed results of comparative experiments for the gear train design problem. From the data in Table 9, it is apparent that the proposed IGTO is better than other optimizers in handling this case and effectively finds a brilliant solution.

Table 9: Comparison results of gear train design

Algorithms	Optimum variables				Minimum gear ratio
	$n_A(z_1)$	$n_B(z_2)$	$n_c(z_3)$	$n_D(z_4)$	
IGTO	48.6044	16.3242	19.3289	43.2518	**2.7009E−12**
GTO	53.0248	12.9137	19.8583	33.5390	2.3078E−11
SMA	39.0400	12.0187	15.2520	31.6773	2.3576E−09
HHO	57.4901	17.0467	14.6957	31.2659	1.0936E−09
SCA	33.1692	17.1429	13.9093	49.8355	1.3616E−09
WOA	23.3375	12.0000	12.5454	46.8061	9.9216E−10
GWO	55.9250	21.5029	17.8696	48.5747	1.2634E−09

Note: The best solution obtained is highlighted in bold.

5.3 Welded Beam Design

As its name implies, the purpose of this welded beam design problem is to reduce the total manufacturing cost as much as possible. This optimum design contains four decision parameters: the width of weld (h), the length of the clamped bar (l), the height of the bar (t), and the bar thickness (b). Besides, in the optimization process, several constraints should not be contravened such as bending stress in the beam, buckling load, shear stress and end deflection. The schematic view of this issue is shown in Fig. 10, and the related mathematical formulation is illustrated as follows: consider

$$\vec{z} = [z_1\ z_2\ \ z_3\ z_4] = [h\ l\ t\ b]$$

minimize

$$f(\vec{z}) = 1.10471 z_1^2 z_2 + 0.04811 z_3 z_4 (14 + z_2)$$

subject to

$$g_1(\vec{z}) = \tau(\vec{z}) - \tau_{\max} \le 0$$
$$g_2(\vec{z}) = \sigma - \sigma_{\max} \le 0$$
$$g_3(\vec{z}) = \delta - \delta_{\max} \le 0$$
$$g_4(\vec{z}) = z_1 - z_4 \le 0$$
$$g_5(\vec{z}) = P - P_C(\vec{z}) \le 0$$
$$g_6(\vec{z}) = 0.125 - z_1 \le 0$$
$$g_7(\vec{z}) = 1.10471 z_1^2 + 0.04811 z_3 z_4 (14 + z_2) - 5 \le 0$$

variable range: $0.1 \le z_1, z_4 \le 2, 0.1 \le z_2, z_3 \le 10$
where

$$\tau(\vec{z}) = \sqrt{(\tau')^2 + 2\tau'\tau'' \frac{z_2}{2R} + (\tau'')^2}$$
$$\tau' = \frac{P}{\sqrt{2} z_1 z_2}, \tau' = \frac{MR}{J}, M = P\left(L + \frac{z_2}{2}\right)$$
$$R = \sqrt{\frac{z_2^2}{4} + \left(\frac{z_1 + z_3}{2}\right)^2}$$
$$J = 2\left\{\sqrt{2} z_1 z_2 \left[\frac{z_2^2}{4} + \left(\frac{z_1 + z_3}{2}\right)^2\right]\right\}$$
$$\sigma(\vec{z}) = \frac{6PL}{E z_3^2 z_4}, \delta(\vec{z}) = \frac{6PL^3}{E z_3^2 z_4}$$
$$P_{\mathrm{C}}(\vec{z}) = \frac{4.013 E \sqrt{\frac{z_3^2 z_4^6}{36}}}{L^2}\left(1 - \frac{z_3}{2L}\sqrt{\frac{E}{4G}}\right)$$
$P = 6000\mathrm{lb}, L = 14\mathrm{in}, E = 30 \times 10^6\mathrm{psi}, G = 12 \times 10^6\mathrm{psi},$
$\delta_{\max} = 0.25\mathrm{in}, \tau_{\max} = 13{,}600\mathrm{psi}, \sigma_{\max} = 30{,}000\mathrm{psi}$

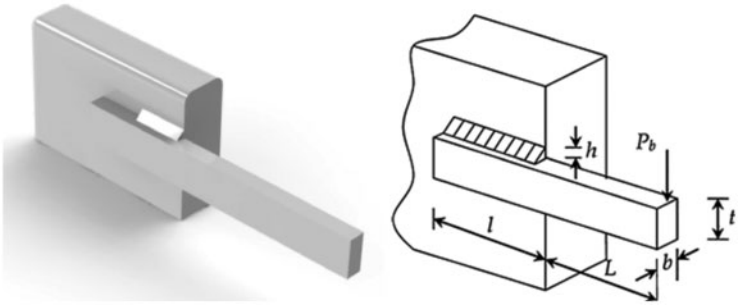

Figure 10: Welded beam design problem

The optimal results of IGTO *vs.* those achieved by GTO, MVO [21], SSA [66], HHO [33], WOA [29], MTDE [71], ESSWOA [9] are reported in Table 10. As can be seen from Table 10, it is obvious that the proposed IGTO provides better design than majority of other algorithms. The minimum cost $f(\vec{z})_{\min} = 1.72485$ is obtained with the related optimal solution $\vec{z} = [0.2057\ 3.4705\ 9.0366\ 0.2057]$. Therefore, it is justifiable to believe that the proposed IGTO has the superior capability to deal with such problem.

Table 10: Comparison results of welded beam design

Algorithms	Optimum variables				Minimum cost
	$h(z_1)$	$l(z_1)$	$t(z_3)$	$b(z_4)$	
IGTO	0.2057	3.4705	9.0366	0.2057	**1.72485**
GTO	0.2068	3.4570	9.0140	0.2068	1.72890
MVO [21]	0.2055	3.4732	9.0445	0.2057	1.72645
SSA [66]	0.2057	3.4714	9.0366	0.2057	1.72491
HHO [33]	0.2040	3.5311	9.0275	0.2061	1.73199
WOA [29]	0.2054	3.4843	9.0374	0.2063	1.73050
MTDE [71]	0.2057	3.4705	9.0366	0.2057	**1.72485**
ESSAWOA [9]	0.2055	3.4753	9.0367	0.2057	1.72516

Note: The best solution obtained is highlighted in bold.

5.4 Rolling Element Bearing Design

Unlike the previous problems, the final objective of this issue is to maximize the dynamic load capacity of rolling element bearings as possible. The structure of a rolling element bearing is illustrated in Fig. 11. There is a total of ten structural variables involved in the solution of this optimization problem, namely: pitch diameter (D_m), ball diameter (D_b), the number of balls (Z), the inner and outer raceway curvature radius coefficient (f_i and f_o), K_{dmin}, K_{dmax}, δ, e as well as ζ. Mathematically, the description of this problem is given as follows:

maximize

$$C_d = \begin{cases} f_c Z^{2/3} D_b^{1.8}, & if\ D_b \le 25.4\ \text{mm} \\ 3.647 f_c Z^{2/3} D_b^{1.4}, & if\ D_b > 25.4\ \text{mm} \end{cases}$$

subject to

$$g_1(\vec{z}) = \frac{\phi_0}{2\sin^{-1}(D_b/D_m)} - Z + 1 \le 0$$
$$g_2(\vec{z}) = 2D_b - K_{\text{dmin}}(D-d) > 0$$
$$g_3(\vec{z}) = K_{\text{dmax}}(D-d) - 2D_b \ge 0$$
$$g_4(\vec{z}) = \zeta B_w - D_b \le 0$$
$$g_5(\vec{z}) = D_m - 0.5(D+d) \ge 0$$
$$g_6(\vec{z}) = (0.5+e)(D+d) - D_m \ge 0$$
$$g_7(\vec{z}) = 0.5(D - D_m - D_b) - \delta D_b \ge 0$$
$$g_8(\vec{z}) = f_i \ge 0.515$$
$$g_9(\vec{z}) = f_o \ge 0.515$$

where

$$f_c = 37.91\left[1+\left\{1.04\left(\frac{1-\gamma}{1+\gamma}\right)^{1.72}\left(\frac{f_i(2f_o-1)}{f_o(2f_i-1)}\right)^{0.41}\right\}^{10/3}\right]^{-0.3} \times \left[\frac{\gamma^{0.3}(1-\gamma)^{1.39}}{(1+\gamma)^{1/3}}\right]\left[\frac{2f_i}{2f_i-1}\right]^{0.41}$$

$$x = [\{(D-d)/2-3(T/4)\}^2+\{D/2-T/4-D_b\}^2-\{d/2+T/4\}^2]$$

$$y = 2\{(D-d)/2-3(T/4)\}\{D/2-T/4-D_b\}$$

$$\phi_0 = 2\prod -\cos^{-1}\left(\frac{x}{y}\right), \gamma = \frac{D_b}{D_m}, f_i = \frac{r_i}{D_b}, f_o = \frac{r_o}{D_b}, T = D-d-2D_b$$

$$D = 160, d = 90, B_w = 30, r_i = r_o = 11.033, 0.5(D+d) \le D_m \le 0.6(D+d)$$

$$0.15(D-d) \le D_b \le 0.45(D-d), 4 \le Z \le 50, 0.515 \le f_i and f_o \le 0.6$$

$$0.4 \le K_{\text{dmin}} \le 0.5, 0.6 \le K_{\text{dmax}} \le 0.7$$

$$0.3 \le \delta \le 0.4, 0.02 \le e \le 0.1, 0.6 \le \zeta \le 0.85$$

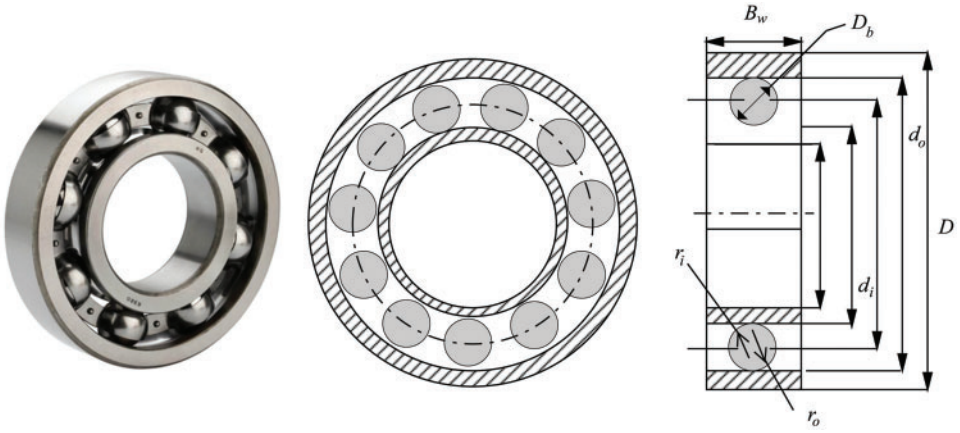

Figure 11: Rolling element bearing design problem

The results of optimum variables and fitness fetched applying different intelligent algorithms are listed in Table 11. Compared with other well-known optimizers, the proposed IGTO reveals the superior quality solution at $\vec{z} = [125\ 21.41885\ 10.94110\ 0.515\ 0.515\ 0.40.7\ 0.3\ 0.02\ 0.6]$ corresponding to the best fitness $C_d = 85067.962$ with a significant improvement. This case once again highlights the applicability of IGTO algorithm.

Table 11: Comparison results of rolling element bearing design

Algorithms	IGTO	TLBO [41]	SHO [30]	HHO [33]	PVS [72]	COOT [73]
D_m	125	125.71910	125	125	125.71906	125
D_b	21.41885	21.42559	21.40732	21.00000	21.42559	21.87500
Z	10.94110	11.00000	10.93268	11.09207	11.00000	10.77700

(Continued)

Table 11 (continued)

Algorithms	IGTO	TLBO [41]	SHO [30]	HHO [33]	PVS [72]	COOT [73]
f_i	0.51500	0.51500	0.51500	0.51500	0.51500	0.51500
f_o	0.51500	0.51500	0.51500	0.51500	0.51500	0.51500
K_{dmin}	0.40000	0.42427	0.40000	0.40000	0.40043	0.43190
K_{dmax}	0.70000	0.63395	0.70000	0.60000	0.68016	0.65290
δ	0.30000	0.30000	0.30000	0.30000	0.30000	0.30000
e	0.02000	0.06886	0.02000	0.05047	0.07999	0.02000
ζ	0.60000	0.79950	0.60000	0.60000	0.70000	0.60000
Optimum cost	**85067.962**	81859.740	85054.532	83011.883	81859.741	83918.492

Note: The best solution obtained is highlighted in bold.

As a summary, it is reasonable to believe that the proposed IGTO is equally feasible and competitive in practical engineering design problems from the observed results. In addition, the excellent performance in resolving engineering design problems indicates that IGTO is able to be widely used in real-world optimization problems as well.

6 IGTO for Training Multilayer Perceptron

Multilayer perceptron (MLP), as one of the most extensively used artificial neural network models [74], has been successfully implemented for solving various real-world issues such as pattern classification [75] and regression analysis [76]. The MLP is characterized by multiple perceptron, in which there is at least one hidden layer in addition to one input layer and one output layer. The information is received as input on one side of the MLP and the output is supplied from the other side via one-way transmission between nodes in different layers. For the MLP, since the sample data space is mostly high-dimensional and multimodal, at the same time there is also a potential for data interference by noise, data redundancy and data loss. Thus, the main purpose of training the MLP is to update two crucial parameters that dominate the final output: the weights W and biases θ, which is a very challenging optimization problem [15,77]. In this section, the Balloon and Breast cancer datasets from the University of California at Irvine (UCI) repository [78] are utilized for examining the applicability of the proposed IGTO algorithm for training MLP. Table 12 presents the specification of these datasets.

Table 12: Specification of the datasets

Datasets	Number of attributes	Number of training samples	Number of classes
Balloon	4	16	2
Breast cancer	9	599	2

In order to measure the algorithm performance of training the MLP, the average mean square error criteria ($\overline{\text{MSE}}$) for all training samples are defined as follows:

$$\overline{\text{MSE}} = \sum_{k=1}^{q} \frac{\sum_{i=1}^{m} (o_i^k - d_i^k)^2}{q} \tag{24}$$

In Eq. (24), q represents the number of training samples, m is the number of outputs, and d_i^k and o_i^k denote the desired and actual output for i-th input with k-th training sample is used, respectively. If the actual output data is closer to the desired one, the value of $\overline{\text{MSE}}$ is smaller, which means that the trained model gains a better performance.

Besides the optimization algorithms shown in Table 2, Tunicate Swarm Algorithm (TSA) [34], Sooty Tern Optimization Algorithm (STOA) [35], and Seagull Optimization Algorithm (SOA) [32] are also taken into account in this experiment. The variables are assumed to be in the range of $[-10, 10]$. Each optimizer executes independently 10 times, with the maximum iterations and population size are set to 250 and 30, respectively. Meanwhile, the parameters of all algorithms are consistent with the original literature. With regard to the structure of the MLP, the number of nodes in the hidden layer is equal to $2n+1$ as recommended in [74], where n denotes the number of attributes in the dataset. Fig. 12 illustrates an example for the process of training the MLP by IGTO.

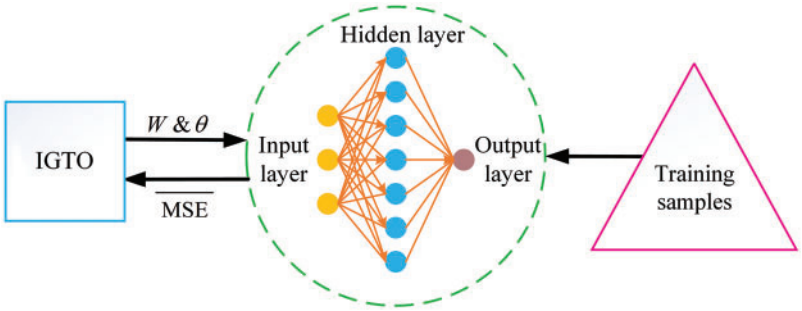

Figure 12: Training MLP by the proposed IGTO

The $\overline{\text{MSE}}$ and classification accuracy attained by each method on the datasets are listed in Table 13. In consideration of the simplicity of the Ballon dataset, all optimisers have achieved 100% classification accuracy except WOA and SMA, yet the proposed IGTO provides a better value of $\overline{\text{MSE}}$ than the others. In relation to the Breast cancer dataset, it is obvious that IGTO still obtains the best result with the $\overline{\text{MSE}}$ of 7.35E–04% and 100% classification accuracy.

All these results demonstrate that the proposed algorithm has a stable and consistent ability to get rid of the local optimum and eventually find the global minima in the complex search space. Besides, this case also highlights the applicability of IGTO algorithm. IGTO is capable of finding more suitable crucial parameters for MLP, thus making it perform better.

Table 13: Comparison results of two datasets

Datasets	Algorithm	$\overline{\text{MSE}}$	Classification accuracy (%)
Balloon	GWO	6.13E−09	**100**
	WOA	2.99E−02	58
	SSA	3.68E−08	**100**
	HHO	2.15E−08	**100**
	SMA	9.37E−03	70
	TSA	1.43E−06	**100**
	STOA	2.50E−06	**100**
	SOA	3.88E−07	**100**
	GTO	4.95E−10	**100**
	IGTO	**1.48E−14**	**100**
Breast cancer	GWO	1.51E−03	96
	WOA	4.43E−03	81
	SSA	4.16E−03	87
	HHO	1.84E−03	98
	SMA	3.20E−03	96
	TSA	3.01E−03	91
	STOA	1.86E−02	67
	SOA	2.57E−02	84
	GTO	1.48E−03	98
	IGTO	**7.35E−04**	**100**

Note: The best result obtained is highlighted in bold.

7 Conclusion and Future Work

In this paper, a novel improved version of the basic gorilla troops algorithm named IGTO was put forward to solve complex global optimization problems. First, Circle chaotic mapping was introduced to enhance the diversity of the initial gorilla population. Second, the lens opposition-based learning strategy was adopted to expand the search domain, thus avoiding the algorithm falling into the local optima. Moreover, the adaptive β-hill climbing algorithm was hybridized with GTO to boost the quality of final solutions. In order to evaluate the effectiveness of the proposed algorithm, IGTO was compared with the basic GTO and five other state-of-the-art algorithms based on 19 classical benchmark functions, including unimodal, multimodal, and fix-dimension multimodal functions. Besides, the non-parametric Wilcoxon's rank-sum test and average absolute error (MAE) were used to analyze the experimental results. The statistical results demonstrate that the proposed IGTO algorithm provides better local optimum avoidance, solution quality, and robustness than the other competitors. Three improvements can significantly boost the performance of IGTO. In order to further test the applicability of IGTO in real-world applications, IGTO was applied to solve four engineering design problems and train multilayer perceptron. The experimental results show that IGTO has strong competitive performance in terms of optimization accuracy.

Nevertheless, as mentioned in the experiment section above, IGTO still has the main limitation of high computation time, which needs to be improved. It is believed that this situation could be

mitigated via the introduction of several parallel mechanisms, e.g., master-slave model, cell model and coordination strategy.

In the future work, we will aim to further enhance the solution accuracy of IGTO while reducing the total process consumption. Also, we plan to further investigate the impact of the lens opposition-based learning and adaptive β-climbing strategies on the performance of other meta-heuristic algorithms. In addition, we hope to apply the proposed technique to solve more practical problems, such as the parameter self-tuning of speed proportional integral differential (PID) controller for brushless direct current motors, the global path planning for autonomous underwater vehicles in a complex environment, and the maximum power point tracking of solar photovoltaic systems.

Acknowledgement: The authors are grateful to the editor and reviewers for their constructive comments and suggestions, which have improved the presentation.

Funding Statement: This work is financially supported by the Fundamental Research Funds for the Central Universities under Grant 2572014BB06.

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

References

1. Zhang, X., Zhao, K., Niu, Y. (2020). Improved harris hawks optimization based on adaptive cooperative foraging and dispersed doraging strategies. *IEEE Access, 8,* 160297–160314. DOI 10.1109/access.2020.3013332.
2. Birogul, S. (2019). Hybrid harris hawk optimization based on differential evolution (HHODE) algorithm for optimal power flow problem. *IEEE Access, 7,* 184468–184488. DOI 10.1109/access.2019.2958279.
3. Hussain, K., Salleh, M. N. M., Cheng, S., Shi, Y. H. (2019). Metaheuristic research: A comprehensive survey. *Artificial Intelligence Review, 52(4),* 2191–2233. DOI 10.1007/s10462-017-9605-z.
4. Abualigah, L., Diabat, A., Mirjalili, S., Abd Elaziz, M., Gandomi, A. H. (2021). The arithmetic optimization algorithm. *Computer Methods in Applied Mechanics and Engineering, 376,* 113609. DOI 10.1016/j.cma.2020.113609.
5. Liang, J., Xu, W., Yue, C., Yu, K., Song, H. et al. (2019). Multimodal multiobjective optimization with differential evolution. *Swarm and Evolutionary Computation, 44,* 1028–1059. DOI 10.1016/j.swevo.2018.10.016.
6. Nadimi-Shahraki, M. H., Taghian, S., Mirjalili, S. (2021). An improved grey wolf optimizer for solving engineering problems. *Expert Systems with Applications, 166,* 113917. DOI 10.1016/j.eswa.2020.113917.
7. Fan, Q., Huang, H., Yang, K., Zhang, S., Yao, L. et al. (2021). A modified equilibrium optimizer using opposition-based learning and novel update rules. *Expert Systems with Applications, 170,* 114575. DOI 10.1016/j.eswa.2021.114575.
8. Boussaid, I., Lepagnot, J., Siarry, P. (2013). A survey on optimization metaheuristics. *Information Sciences, 237,* 82–117. DOI 10.1016/j.ins.2013.02.041.
9. Fan, Q., Chen, Z., Zhang, W., Fang, X. (2020). ESSAWOA: Enhanced whale optimization algorithm integrated with salp swarm algorithm for global optimization. *Engineering with Computers,* DOI 10.1007/s00366-020-01189-3.
10. Dokeroglu, T., Sevinc, E., Kucukyilmaz, T., Cosar, A. (2019). A survey on new generation metaheuristic algorithms. *Computers & Industrial Engineering, 137,* 106040. DOI 10.1016/j.cie.2019.106040.
11. Slowik, A., Kwasnicka, H. (2018). Nature inspired methods and their industry applications—swarm intelligence algorithms. *IEEE Transactions on Industrial Informatics, 14(3),* 1004–1015. DOI 10.1109/tii.2017.2786782.

12. Abualigah, L., Alsalibi, B., Shehab, M., Alshinwan, M., Khasawneh, A. M. et al. (2020). A parallel hybrid krill herd algorithm for feature selection. *International Journal of Machine Learning and Cybernetics, 12(3),* 783–806. DOI 10.1007/s13042-020-01202-7.
13. Debnath, S., Baishya, S., Sen, D., Arif, W. (2020). A hybrid memory-based dragonfly algorithm with differential evolution for engineering application. *Engineering with Computers, 37(4),* 2775–2802. DOI 10.1007/s00366-020-00958-4.
14. Nguyen, T. T., Wang, H. J., Dao, T. K., Pan, J. S., Liu, J. H. et al. (2020). An improved slime mold algorithm and its application for optimal operation of cascade hydropower stations. *IEEE Access, 8,* 226754–226772. DOI 10.1109/access.2020.3045975.
15. Jia, H., Sun, K., Zhang, W., Leng, X. (2021). An enhanced chimp optimization algorithm for continuous optimization domains. *Complex & Intelligent Systems,* DOI 10.1007/s40747-021-00346-5.
16. Dehghani, M., Montazeri, Z., Givi, H., Guerrero, J., Dhiman, G. (2020). Darts game optimizer: A new optimization technique based on darts game. *International Journal of Intelligent Engineering and Systems, 13(5),* 286–294. DOI 10.22266/ijies2020.1031.26.
17. Hamed, A. Y., Alkinani, M. H., Hassan, M. R. (2020). A genetic algorithm optimization for multi-objective multicast routing. *Intelligent Automation & Soft Computing, 26(6),* 1201–1216. DOI 10.32604/iasc.2020.012663.
18. Jiang, A., Guo, X., Zheng, S., Xu, M. (2021). Parameters identification of tunnel jointed surrounding rock based on Gaussian process regression optimized by difference evolution algorithm. *Computer Modeling in Engineering & Sciences, 127(3),* 1177–1199. DOI 10.32604/cmes.2021.014199.
19. Simon, D. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12(6),* 702–713. DOI 10.1109/TEVC.2008.919004.
20. Kirkpatrick, S., Gelatt, C. D., Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220(4598),* 671–680. DOI 10.1126/science.220.4598.671.
21. Mirjalili, S., Mirjalili, S. M., Hatamlou, A. (2015). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27(2),* 495–513. DOI 10.1007/s00521-015-1870-7.
22. Kaveh, A., Dadras, A. (2017). A novel meta-heuristic optimization algorithm: Thermal exchange optimization. *Advances in Engineering Software, 110,* 69–84. DOI 10.1016/j.advengsoft.2017.03.014.
23. Zhao, W., Wang, L., Zhang, Z. (2019). Atom search optimization and its application to solve a hydrogeologic parameter estimation problem. *Knowledge-Based Systems, 163,* 283–304. DOI 10.1016/j.knosys.2018.08.030.
24. Faramarzi, A., Heidarinejad, M., Stephens, B., Mirjalili, S. (2020). Equilibrium optimizer: A novel optimization algorithm. *Knowledge-Based Systems, 191,* 105190. DOI 10.1016/j.knosys.2019.105190.
25. Shi, Y., Eberhart, R. C. (1999). Empirical study of particle swarm optimization. *Proceedings of the 1999 Congress on Evolutionary Computation*, pp. 1945–1950. Washington, DC, USA.
26. Dorigo, M., Birattari, M., Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine, 1(4),* 28–39. DOI 10.1109/MCI.2006.329691.
27. Meng, X., Liu, Y., Gao, X., Zhang, H. (2014). A new bio-inspired algorithm: Chicken swarm optimization. *International Conference in Swarm Intelligence*, pp. 86–94. Cham, Switzerland: Springer. DOI 10.1007/978-3-319-11857-4_10.
28. Mirjalili, S. (2016). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications, 27(4),* 1053–1073. DOI 10.1007/s00521-015-1920-1.
29. Mirjalili, S., Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95,* 51–67. DOI 10.1016/j.advengsoft.2016.01.008.
30. Dhiman, G., Kumar, V. (2017). Spotted hyena optimizer: A novel bio-inspired based metaheuristic technique for engineering applications. *Advances in Engineering Software, 114,* 48–70. DOI 10.1016/j.advengsoft.2017.05.014.
31. Dhiman, G., Kumar, V. (2018). Emperor penguin optimizer: A bio-inspired algorithm for engineering problems. *Knowledge-Based Systems, 159,* 20–50. DOI 10.1016/j.knosys.2018.06.001.

32. Dhiman, G., Kumar, V. (2019). Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. *Knowledge-Based Systems, 165,* 169–196. DOI 10.1016/j.knosys.2018.11.024.
33. Heidari, A. A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M. et al. (2019). Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems, 97,* 849–872. DOI 10.1016/j.future.2019.02.028.
34. Kaur, S., Awasthi, L. K., Sangal, A. L., Dhiman, G. (2020). Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. *Engineering Applications of Artificial Intelligence, 90,* 103541. DOI 10.1016/j.engappai.2020.103541.
35. Dhiman, G., Kaur, A. (2019). STOA: A bio-inspired based optimization algorithm for industrial engineering problems. *Engineering Applications of Artificial Intelligence, 82,* 148–174. DOI 10.1016/j.engappai.2019.03.021.
36. Li, S., Chen, H., Wang, M., Heidari, A. A., Mirjalili, S. (2020). Slime mould algorithm: A new method for stochastic optimization. *Future Generation Computer Systems, 111,* 300–323. DOI 10.1016/j.future.2020.03.055.
37. Dhiman, G., Garg, M., Nagar, A., Kumar, V., Dehghani, M. (2020). A novel algorithm for global optimization: Rat swarm optimizer. *Journal of Ambient Intelligence and Humanized Computing, 12(8),* 8457–8482. DOI 10.1007/s12652-020-02580-0.
38. Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A. A., Al-qaness, M. A. A. et al. (2021). Aquila optimizer: A novel meta-heuristic optimization algorithm. *Computers & Industrial Engineering, 157,* 107250. DOI 10.1016/j.cie.2021.107250.
39. Gonçalves, M. S., Lopez, R. H., Miguel, L. F. F. (2015). Search group algorithm: A new meta-heuristic method for the optimization of truss structures. *Computers & Structures, 153,* 165–184. DOI 10.1016/j.compstruc.2015.03.003.
40. Moosavian, N., Roodsari, B. K. (2014). Soccer league competition algorithm: A novel meta-heuristic algorithm for optimal design of water distribution networks. *Swarm and Evolutionary Computation, 17,* 14–24. DOI 10.1016/j.swevo.2014.02.002.
41. Rao, R. V., Savsani, V. J., Vakharia, D. (2011). Teaching-learning-based optimization: A novel method for con-strained mechanical design optimization problems. *Computer-Aided Design, 43(3),* 303–315. DOI 10.1016/j.cad.2010.12.015.
42. Wu, T., Liu, C. C., He, C. (2019). Fault diagnosis of bearings based on KJADE and VNWOA-lSSVM algorithm. *Mathematical Problems in Engineering, 2019,* 1–19. DOI 10.1155/2019/8784154.
43. Ghosh, K. K., Ahmed, S., Singh, P. K., Geem, Z. W., Sarkar, R. (2020). Improved binary sailfish optimizer based on adaptive β-hill climbing for feature selection. *IEEE Access, 8,* 83548–83560. DOI 10.1109/access.2020.2991543.
44. Tang, A., Zhou, H., Han, T., Xie, L. (2021). A chaos sparrow search algorithm with logarithmic spiral and adaptive step for engineering problems. *Computer Modeling in Engineering & Sciences, 129(1),* 1–34. DOI 10.32604/cmes.2021.017310.
45. Yan, Z., Zhang, J., Tang, J. (2021). Path planning for autonomous underwater vehicle based on an enhanced water wave optimization algorithm. *Mathematics and Computers in Simulation, 181,* 192–241. DOI 10.1016/j.matcom.2020.09.019.
46. El-Fergany, A. A. (2021). Parameters identification of PV model using improved slime mould optimizer and Lambert W-function. *Energy Reports, 7,* 875–887. DOI 10.1016/j.egyr.2021.01.093.
47. Wolpert, D. H., Macready, W. G. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation, 1(1),* 67–82. DOI 10.1109/4235.585893.
48. Jia, H., Lang, C., Oliva, D., Song, W., Peng, X. (2019). Dynamic harris hawks optimization with mutation mechanism for satellite image segmentation. *Remote Sensing, 11(12),* 1421. DOI 10.3390/rs11121421.
49. Ding, H., Wu, Z., Zhao, L. (2020). Whale optimization algorithm based on nonlinear convergence factor and chaotic inertial weight. *Concurrency and Computation: Practice and Experience, 32(24),* e5949. DOI 10.1002/cpe.5949.

50. Jia, H., Lang, C. (2021). Salp swarm algorithm with crossover scheme and Lévy flight for global optimization. *Journal of Intelligent & Fuzzy Systems, 40(5),* 9277–9288. DOI 10.3233/jifs-201737.
51. Abdel-Basset, M., Chang, V., Mohamed, R. (2020). HSMA_WOA: A hybrid novel slime mould algorithm with whale optimization algorithm for tackling the image segmentation problem of chest X-ray images. *Applied Soft Computing, 95,* 106642. DOI 10.1016/j.asoc.2020.106642.
52. Liu, C. A., Lei, Q., Jia, H. (2020). Hybrid imperialist competitive evolutionary algorithm for solving biobjective portfolio problem. *Intelligent Automation & Soft Computing, 26(6),* 1477–1492. DOI 10.32604/iasc.2020.011853.
53. Dhiman, G. (2019). ESA: A hybrid bio-inspired metaheuristic optimization approach for engineering problems. *Engineering with Computers, 37(1),* 323–353. DOI 10.1007/s00366-019-00826-w.
54. Abdollahzadeh, B., Soleimanian Gharehchopogh, F., Mirjalili, S. (2021). Artificial gorilla troops optimizer: A new nature-inspired metaheuristic algorithm for global optimization problems. *International Journal of Intelligent Systems, 36(10),* pp. 5887–5958. DOI 10.1002/int.22535.
55. Ginidi, A., Ghoneim, S. M., Elsayed, A., El-Sehiemy, R., Shaheen, A. et al. (2021). Gorilla troops optimizer for electrically based single and double-diode models of solar photovoltaic systems. *Sustainability, 13(16),* 9459. DOI 10.3390/su13169459.
56. Duan, Y., Liu, C., Li, S. (2021). Battlefield target grouping by a hybridization of an improved whale optimization algorithm and affinity propagation. *IEEE Access, 9,* 46448–46461. DOI 10.1109/access.2021.3067729.
57. Kaur, G., Arora, S. (2018). Chaotic whale optimization algorithm. *Journal of Computational Design and Engineering, 5(3),* 275–284. DOI 10.1016/j.jcde.2017.12.006.
58. Tizhoosh, H. R. (2005). Opposition-based learning: A new scheme for machine intelligence. *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, web Technologies and Internet Commerce*, pp. 695–701. Vienna, Austria.
59. Ouyang, C., Zhu, D., Qiu, Y., Zhang, H. (2021). Lens learning sparrow search algorithm. *Mathematical Problems in Engineering, 2021,* 1–17. DOI 10.1155/2021/9935090.
60. Long, W., Wu, T., Tang, M., Xu, M., Cai, S. (2020). Grey wolf optimizer algorithm based on lens imaging learning strategy. *Acta Automatica Sinica, 46(10),* 2148–2164. DOI 10.16383/j.aas.c180695.
61. Al-Betar, M. A., Aljarah, I., Awadallah, M. A., Faris, H., Mirjalili, S. (2019). Adaptive β-hill climbing for optimization. *Soft Computing, 23(24),* 13489–13512. DOI 10.1007/s00500-019-03887-7.
62. Glover, F. (1986). Future paths for integer programming and links to artificial intelligence. *Computers & Operations Research, 13(5),* 533–549. DOI 10.1016/0305-0548(86)90048-1.
63. Mladenović, N., Hansen, P. (1997). Variable neighborhood search. *Computers & Operations Research, 24(11),* 1097–1100. DOI 10.1016/S0305-0548(97)00031-2.
64. Long, W., Jiao, J., Liang, X., Wu, T., Xu, M. et al. (2021). Pinhole-imaging-based learning butterfly optimization algorithm for global optimization and feature selection. *Applied Soft Computing, 103,* 107146. DOI 10.1016/j.asoc.2021.107146.
65. Mirjalili, S., Mirjalili, S. M., Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software, 69,* 46–61. DOI 10.1016/j.advengsoft.2013.12.007.
66. Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H. et al. (2017). Salp swarm algorithm: A bio-inspired optimizer for engineering design problems. *Advances in Engineering Software, 114,* 163–191. DOI 10.1016/j.advengsoft.2017.07.002.
67. García, S., Fernández, A., Luengo, J., Herrera, F. (2010). Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. *Information Sciences, 180(10),* 2044–2064. DOI 10.1016/j.ins.2009.12.010.
68. Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89,* 228–249. DOI 10.1016/j.knosys.2015.07.006.
69. Kannan, B., Kramer, S. N. (1994). An augmented lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. *Journal of Mechanical Design, 116(2),* 405–411. DOI 10.1115/1.2919393.

70. Sandgren, E. (1990). Nonlinear integer and discrete programming in mechanical design optimization. *Journal of Mechanical Design, 112(2),* 223–229. DOI 10.1115/1.2912596.
71. Nadimi-Shahraki, M. H., Taghian, S., Mirjalili, S., Faris, H. (2020). MTDE: An effective multi-trial vector-based differential evolution algorithm and its applications for engineering design problems. *Applied Soft Computing, 97,* 106761. DOI 10.1016/j.asoc.2020.106761.
72. Savsani, P., Savsani, V. (2016). Passing vehicle search (PVS): A novel metaheuristic algorithm. *Applied Mathematical Modelling, 40(5),* 3951–3978. DOI 10.1016/j.apm.2015.10.040.
73. Naruei, I., Keynia, F. (2021). A new optimization method based on COOT bird natural life model. *Expert Systems with Applications, 183,* 115352. DOI 10.1016/j.eswa.2021.115352.
74. Mirjalili, S., Mirjalili, S. M. Lewis, A. (2014). Let a biogeography-based optimizer train your multi-layer perceptron. *Information Sciences, 269,* 188–209. DOI 10.1016/j.ins.2014.01.038.
75. Melin, P., Sánchez, D., Castillo, O. (2012). Genetic optimization of modular neural networks with fuzzy response integration for human recognition. *Information Sciences, 197,* 1–19. DOI 10.1016/j.ins.2012.02.027.
76. Guo, Z. X., Wong, W. K., Li, M. (2012). Sparsely connected neural network-based time series forecasting. *Information Sciences, 193,* 54–71. DOI 10.1016/j.ins.2012.01.011.
77. Wang, L., Zhang, D., Fan, Y., Xu, H., Wang, Y. (2021). Multilayer perceptron training based on a Cauchy variant grey wolf optimizer algorithm. *Computer Engineering and Science, 43(6),* 1131–1140. DOI 10.3969/j.issn.1007-130X.2021.06.024.
78. Dua, D., Graff, C. (2019). *UCI machine learning repository*. Irvine, CA: University of California, School of Information and Computer Science http://archive.ics.uci.edu/ml.