Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2023.035911
Article

Dark Forest Algorithm: A Novel Metaheuristic Algorithm for Global Optimization Problems

Dongyang Li[1], Shiyu Du[2,*], Yiming Zhang[2] and Meiting Zhao[3]

[1]Faculty of Electrical Engineering and Computer Science, Ningbo University, Ningbo, 315000, China
[2]Engineering Laboratory of Advanced Energy Materials, Ningbo Institute of Materials Technology and Engineering, Chinese Academy of Sciences, Ningbo, 315000, China
[3]School of Material and Chemical Engineering, Ningbo University, Ningbo, 315000, China
*Corresponding Author: Shiyu Du. Email: dushiyu@nimte.ac.cn
Received: 09 September 2022; Accepted: 30 December 2022

Abstract: Metaheuristic algorithms, as effective methods for solving optimization problems, have recently attracted considerable attention in science and engineering fields. They are popular and have broad applications owing to their high efficiency and low complexity. These algorithms are generally based on the behaviors observed in nature, physical sciences, or humans. This study proposes a novel metaheuristic algorithm called dark forest algorithm (DFA), which can yield improved optimization results for global optimization problems. In DFA, the population is divided into four groups: highest civilization, advanced civilization, normal civilization, and low civilization. Each civilization has a unique way of iteration. To verify DFA's capability, the performance of DFA on 35 well-known benchmark functions is compared with that of six other metaheuristic algorithms, including artificial bee colony algorithm, firefly algorithm, grey wolf optimizer, harmony search algorithm, grasshopper optimization algorithm, and whale optimization algorithm. The results show that DFA provides solutions with improved efficiency for problems with low dimensions and outperforms most other algorithms when solving high dimensional problems. DFA is applied to five engineering projects to demonstrate its applicability. The results show that the performance of DFA is competitive to that of current well-known metaheuristic algorithms. Finally, potential upgrading routes for DFA are proposed as possible future developments.

Keywords: Metaheuristic; algorithm; global optimization

1 Introduction

With the continuing development of modern industry, optimization strategies are becoming increasingly essential; therefore, novel optimization algorithms need to be developed. To date, numerous optimization algorithms have been proposed, with nature-inspired algorithms being some of the most successful algorithms. Compared to traditional optimization methods, such as the gradient

descent and direct search methods [1,2], metaheuristic algorithms have the following advantages: 1) Simplicity: The process and the procedural implementation of metaheuristic algorithms are often relatively straightforward. 2) Flexibility: Metaheuristic algorithms can be applied to many different fields. 3) Avoiding local optima: Metaheuristic algorithms aim to search beyond the local optima for the global optima of the target solutions or to directly find global optima. 4) Free of derivatives: Derivatives are generally expensive or difficult to evaluate within complex systems, and metaheuristic algorithms usually do not employ gradients for optimization. 5) Fast Processing: Most metaheuristic algorithms perform quick searches in the sample space to obtain a suitable solution.

Metaheuristic algorithms are often inspired by observations from natural phenomena. For example, the ant colony optimization (ACO) algorithm is inspired by the foraging behavior of ants [3]; the particle swarm optimization (PSO) algorithm mimics the predatory behavior of birds [4]; the genetic algorithm (GA) models the crossover variation of DNA [5]; the simulated annealing algorithm simulates the annealing technology in materials physics [6]. Algorithm designs aim not to mimic the behavior of an individual or population but rather to derive an algorithm to solve specific problems. Thus, versatile and intelligent variants could be introduced into metaheuristic algorithms to solve various optimization problems. In fact, metaheuristic algorithms are widely used in industrial applications, including solar energy forecasting, demand and supply analysis optimization in food manufacturing industry [7–9].

Most metaheuristic algorithms are characterized by their randomness, communication, exploration, and exploitation. Randomness provides algorithms with the possibility of achieving superior optimization solutions. Communication renders information exchange between individual solutions and thereby enables their learning from each other to yield superior optimization solutions. Exploration provides algorithms with a trail of new ideas or strategies, while exploitation allows algorithms to adopt the techniques that have proven successful in the past [10].

In this study, a new algorithm called dark forest algorithm (DFA) is proposed based on the rule of superiority and inferiority among natural civilizations and the universe's dark forest law [11]. Then, the performance of DFA is compared with that of other well-known metaheuristic algorithms according to 35 benchmark functions and five engineering projects. The 35 benchmark functions comprise single-peaked, multi-peaked, high-dimensional, and fixed-dimensional functions. The results show that DFA can obtain qualified optimization results and it outperforms the compared algorithms in terms of overall performance.

The rest of this paper is organized as follows. Section 2 introduces the literature related to the development and applications of metaheuristic algorithms. Section 3 describes the mathematical model of the proposed DFA and the workflow and pseudo-code of the algorithm. Section 4 compares the performance of DFA with six other metaheuristic algorithms on 35 benchmark functions. Section 5 illustrates three engineering design problems using DFA with discussions on their outcomes. Finally, Section 6 presents the conclusions and suggestions for future research.

2 Related Works

Many intricate and fascinating phenomena can be observed in nature that provide inspiration for solving practical problems. The known metaheuristic algorithms can be classified into five categories according to the sources of their inspiration: evolutionary algorithms, swarm intelligence algorithms, physics-based algorithms, human-based algorithms, and other algorithms (Fig. 1).

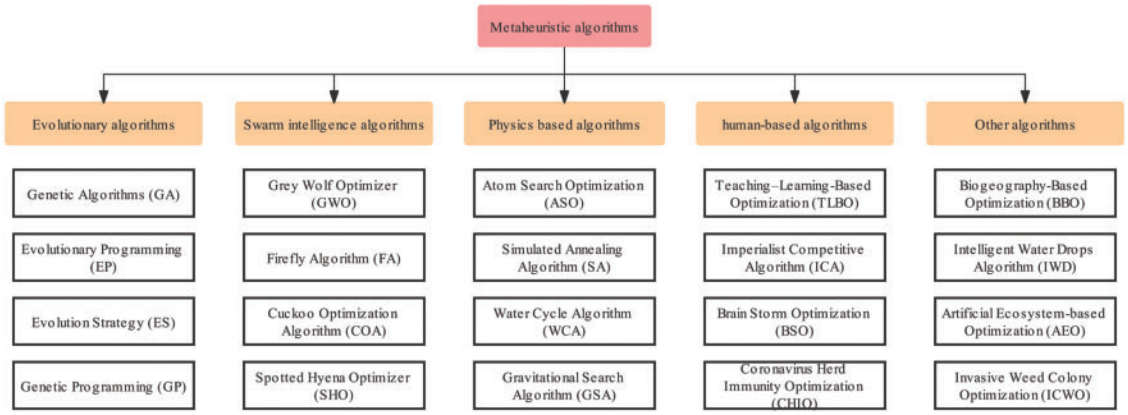

Figure 1: Classification of metaheuristic algorithms

Evolutionary algorithms are based on natural evolution and are the first metaheuristic algorithms proposed. They are the most commonly used. They are based on phenomena and developed theories within biological evolution. Typical algorithms in this category include GA, evolutionary programming [12], differential evolution algorithms [13], evolution strategy [14], flower pollination algorithm [15], and harmony search algorithm (HS) [16]. Among them, GA is the earliest, best known, and one of the most widely used evolutionary algorithms. To date, GA has solved many problems, such as carpool service problems in cloud computing [17], image enhancement and segmentation [18], and routing problems with lost sales [19].

Swarm intelligence algorithms are constructed by simulating group activities inspired by swarms in nature, such as coenosis or social animals. The so-called group intelligence includes the behaviors of simple individuals and the whole group, exhibiting a specific intelligent feature without centralized control. Typical examples include ACO algorithm, PSO algorithm, artificial bee colony algorithm (ABC) [20], artificial fish swarming algorithm [21], grey wolf optimizer (GWO) [22], firefly algorithm (FA) [23], cuckoo search algorithm [24], chimp optimization algorithm [25], grasshopper optimization algorithm (GOA) [26], slime mold algorithm [27], and whale optimization algorithm (WOA) [28].

Physics-based algorithms are inspired by the physics of matter, and they usually have a solid theoretical basis. Some popular algorithms in this category are SA, central force optimization [29], gravitational search algorithm [30], water cycle algorithm [31], atom search optimization [32], electromagnetic field optimization [33], Henry gas solubility optimization [34], and wind driven optimization [35].

Human-based algorithms are designed based on human activities in the society. For example, teaching-learning-based optimization (TLBO) is inspired by the interaction between a learner's learning and a teacher's teaching [36]. TLBO simulates the traditional classroom teaching process. The entire optimization process includes the teacher and learner sections. At the teacher level, each student learns from the best individuals. During the learning section, each student randomly learns from the other students. Examples of popular human-based algorithms are brain storm optimization [37], imperialist competitive algorithm [38], cultural algorithm [39], coronavirus herd immunity optimization [40], and team competition and cooperation optimization algorithm [41].

Some algorithms do not fall into any of the above categories but are inspired by other natural phenomena, such as water drops algorithm [42], artificial ecosystem-based optimization [43], and invasive weed colony optimization [44].

Although many metaheuristic algorithms already exist, new algorithms still need to be designed. According to the No Free Lunch (NFL) theorem [45], no single metaheuristic algorithm can accurately solve all optimization problems. In practice, some algorithms perform better than others in specific situations, and many researchers are trying to find ways to improve existing metaheuristic algorithms or develop new ones. This study proposes a novel algorithm, i.e., DFA. Rigorous experiments are conducted to demonstrate DFA as a feasible metaheuristic algorithm that can be readily used in engineering problems.

3 Dark Forest Algorithm

3.1 Inspiration

Civilizations in the universe survive under the dark forest law. According to the dark forest law, a civilization once discovered will inevitably be attacked by other civilizations in the universe. The universe's evolution is endless and is always accompanied by the extinction of existing civilizations and the birth of new ones. Civilizations plunder each other and cooperate, constantly moving toward a better direction. Civilizations can be classified according to their level of development: highest, advanced, normal, and low civilizations.

Each civilization has its own exploration strategy. The highest civilizations cannot learn from others because no known civilizations are superior to them, and they usually move around during exploration. The highest civilizations sometimes plunder other civilizations for their development. If the highest civilization does not evolve, they remain unchanged. Advanced civilizations learn from the highest civilizations and plunder the normal civilizations. Normal civilizations learn from the highest and advanced civilizations, and they choose which civilizations to learn from based on the strengths and weaknesses of the other civilizations and their distance. Finally, the low civilizations are subject to elimination, at which point newly created civilizations enter the iterations.

3.2 Algorithm

This section describes the mathematical model, algorithmic workflow, and pseudo-code of DFA. The general workflow of the algorithm is as follows: 1) randomly initializing the population coordinates in the solution space; 2) classifying civilizations according to the adaptability of each civilization coordinate; 3) iteratively updating all locations according to the corresponding civilization level; and 4) separately performing a refined search for the highest civilization in the last few iterations. Fig. 2 illustrates the flowchart.

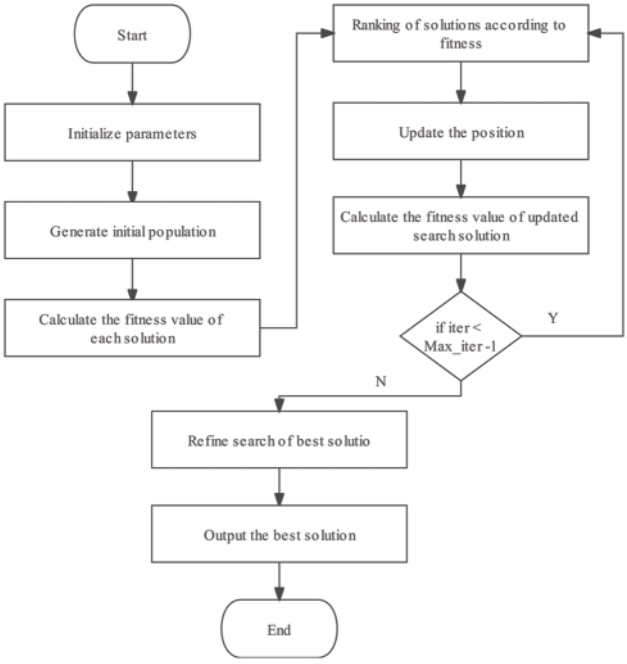

Figure 2: Flowchart of the proposed DFA

3.2.1 Population Initialization

DFA is a population-based algorithm. Similar to other population-based metaheuristic algorithms, it generates several uniformly distributed initial solutions in the search space at the beginning as follows:

$$x_i^j = x_{i,min}^j + \alpha \cdot \left(x_{i,max}^j - x_{i,min}^j\right) \tag{1}$$

$$X_{\mathrm{i}} = \left[x_{\mathrm{i}}^1, x_{\mathrm{i}}^2, \ldots, x_{\mathrm{i}}^{\mathrm{d}}\right] \tag{2}$$

where x_i^j is the initial value of the jth component of the ith solution, $x_{i,min}^j$ is the minimum value allowed for the jth component of the candidate solution, $x_{i,max}^j$ is the maximum value allowed for the j th component of the candidate solution, and α is a uniformly distributed random number in the range of [0, 1]. These components form the corresponding initial solution X_{i}, and d is the dimension of the solution. X_{i} corresponds to the degree of adaptation; $f(X_i)$ is evaluated according to the fitness function as follows:

$$f(X_i) = f\left(\left[x_{\mathrm{i}}^1, x_{\mathrm{i}}^2, \ldots, x_{\mathrm{i}}^{\mathrm{d}}\right]\right) \tag{3}$$

All solutions are constructed into a matrix X:

$$X = \begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_i \\ \vdots \\ X_n \end{bmatrix} = \begin{bmatrix} x_1^1 & x_1^2 & \ldots & x_1^j & \ldots & x_1^d \\ x_2^1 & x_2^2 & \ldots & x_2^j & \ldots & x_2^d \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_i^1 & x_i^2 & \ldots & x_i^j & \ldots & x_i^d \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \ldots & x_n^j & \ldots & x_n^d \end{bmatrix}, \begin{cases} i = 1, 2, \ldots, n \\ j = 1, 2, \ldots, d \end{cases} \tag{4}$$

where n is the number of populations, i is an integer in the range of $[1, n]$, and j is an integer in the range of $[1, d]$.

3.2.2 Civilization Location Update

After the initial population is generated, the population is sorted according to the fitness values of each civilization. The top-ranked civilization is the highest civilization; the civilizations with the top 3/10 of the population, except for the highest civilization, are the advanced civilizations. The civilizations with ranking between 3/10 and 8/10 are defined as the normal civilizations, and the remaining civilizations are defined as low civilizations. Different civilization types update their coordinates in different ways.

Locations of the highest civilizations are updated as follows. The highest civilization has globally optimal fitness, and its primary purpose of movement is to find coordinates with enhanced fitness. During the iteration process, the highest civilization only changes its coordinates when it finds improved fitness. The highest civilization moves randomly in 50% of the cases and takes reference from the advanced civilization for the other 50% of the cases. The update formulas are as follows:

$$X_{i,t+1} = \begin{cases} X_{\text{new}}, & f\left(X_{\text{new}}\right) < f\left(X_{i,t+1}\right) \\ X_{i,t}, & \text{otherwise} \end{cases} \tag{5}$$

$$X_{\text{new}} = \begin{cases} X_{\text{try}}, & p < 0.5 \\ X_{i,t} + r \cdot \text{step} \cdot U, & \text{otherwise} \end{cases} \tag{6}$$

$$X_{\text{try}}^j = \begin{cases} X_{i,\text{t}}^j, & p < 0.5 \\ X_{s,t}^j, & \text{otherwise} \end{cases} \tag{7}$$

$$\text{step} = \left(1 - \frac{\text{t}}{\text{Max\_Iter}}\right)^{\frac{2t}{\text{Max\_Iter}}} \tag{8}$$

where t is the current index of iteration, and p and r are random numbers in the range of $[0, 1]$. U is a randomly generated vector with the same dimension as X_i and elements in the range of $[-1, 1]$, and *step* is the step size that decreases with increasing number of iterations, allowing civilizations to have exploration capability at the beginning and a more vital exploitation ability at the end. $X_{s,t}$ is a randomly selected advanced civilization, and *Max_Iter* is the maximum number of iterations.

Locations of the advanced civilizations are updated as follows. Advanced civilizations obtain reference from normal civilizations for 20% of the cases. Since the highest civilization has better fitness than the advanced civilization, advanced civilizations that move to the highest civilizations are more likely to obtain superior results. Hence, the probability for advanced civilizations to move to the highest civilization is set as 80%. Advanced civilizations use spiral location updates in most cases, similar to

the spiral update in WOA, but with variations. Mathematically, the update formulas are as follows:

$$X_{i,t+1} = \begin{cases} X_{\text{try}}, & p < 0.2 \\ X_{i,t} + \mathrm{D} \cdot e^{bl} \cdot \cos(2\pi l), & \text{otherwise} \end{cases} \tag{9}$$

$$X_{\text{try}}^{j} = \begin{cases} X_{i,t}^{j}, & p < 0.5 \\ X_{c,t}^{j}, & \text{otherwise} \end{cases} \tag{10}$$

$$\mathrm{D} = \left| X^{*} - X_{i,t} \right| \tag{11}$$

where b is a logarithmic spiral shape constant, l is a random number in the range of $[-1, 1]$, $X_{c,t}$ is a randomly selected normal civilization, and X^{*} is the coordinate of a highest civilization.

Locations of normal civilizations are updated as follows. Normal civilizations also employ spiral position update. The reference is a civilization coordinate obtained via linear ranking and roulette selection, which may be either the highest civilization or an advanced civilization. In the selection process, the probability of each civilization is jointly determined by its fitness and its distance from the current normal civilization, with the weight ratio of fitness to distance length of 4:1. The update formulas are

$$X_{i,t+1} = X_{i,t} + \mathrm{D} \cdot e^{bl} \cdot \cos(2\pi l) \tag{12}$$

$$\mathrm{D} = \left| X_{m,t} - X_{i,t} \right| \tag{13}$$

where $X_{m,t}$ is a civilization coordinate obtained by linear ranking and roulette selection.

Locations of low civilizations are updated as follows. Due to the poor adaptation of the low civilizations themselves, the coordinates of the low civilizations are reset for 50% of the cases. This is reasonable as the primary responsibility of a low civilization is to maintain the population diversity to ensure that it does not fall into a local optimum. A new coordinate is updated by mapping the coordinate centroid of mass of all the highest and advanced civilizations for 50% of the cases. The update formulas are

$$X_{i,t+1} = \begin{cases} X_{\text{init}}, & p < 0.5 \\ X_{w} + \delta \cdot \left(X_{w} - X_{i,t} \right) \cdot \text{step}, & \text{otherwise} \end{cases} \tag{14}$$

$$X_{w} = \frac{1}{N} \sum\nolimits_{1}^{N} X_{i} \tag{15}$$

where X_{init} is a coordinate re-generated at random using Eq. (1), X_{w} is the generated centroid of mass of the highest and advanced civilizations, δ is the mapping factor with values in the range of [0, 1], and N is the sum of the number of highest and advanced civilizations.

After all civilization locations are updated, one refined search in the last ten iterations is performed for the highest civilizations. The refined search is performed to update each component $X_{i,t+1}$ of the coordinates of the highest civilization. At the beginning of an initial update vector β, the fitness of $X_{i,t+1}$ plus β_i is calculated corresponding to X_{new}. If no better fitness is obtained, this update is abandoned and the minus value of β_i is taken. If a better fitness is obtained, then β_i is kept unchanged. If no better result is obtained in two consecutive updates of β_i, then β_i is inverted and reduced by half. The refined search process only updates the highest civilizations for better optimization results. The above stated process can be represented as

$$X_{i,t+1} = \begin{cases} X_{\text{new}}, & f\left(X_{\text{new}}\right) < f\left(X_{i,t+1}\right) \\ X_{i,t}, & \text{otherwise} \end{cases} \tag{16}$$

$$X_{\text{new}}^{j} = X_{i,t}^{j} + (-1)^{k} \cdot \beta_{i}^{\lceil \frac{k}{2} \rceil} \tag{17}$$

$$k = \begin{cases} k, & f\left(x_{\text{new}}\right) < f\left(x_{i,t+1}\right) \\ (k+1) \bmod 2, & \text{otherwise} \end{cases} \tag{18}$$

where β is the search radius typically in the range of [0, 1] and the value of k is initially 1.

3.3 Termination

The algorithm ends after a specified number of iterations is reached, with the coordinates recorded by the highest civilization at the end being the optimal solution and the corresponding fitness being the optimal fitness. The pseudo-code of DFA algorithm is presented in Table 1.

Table 1: DFA pseudo-code

Algorithm
Input: population X_i $(i = 1, 2, \ldots, n)$
Output: the best solution Y^{best}
1: **procedure** DFA
2: Initialize parameters b, δ, and β
3: Compute the fitness of each solution
4: **while** *iter* < Max number of iterations − 10 **do**
5: Ranking of solutions according to fitness
6: Update Y^{best} if there is a better solution than previous best solution
7: **for** highest civilization **do**
8: Update the position by Eqs. (5)–(8)
9: **end for**
10: **for** each advanced civilization **do**
11: Update the position by Eqs. (9)–(11)
13: **for** each normal civilization **do**
14: Update the position by Eqs. (12), (13)
15: **end for**
16: **for** each low civilization **do**
17: Update the position by Eq. (1) and Eqs. (14), (15)
18: **end for**
19: Check if any search solution goes beyond the given search space and then adjust it
20: Compute the fitness of each solution
21: $iter = iter + 1$
22: **end while**

(Continued)

Table 1: Continued

Algorithm
23: Refine search of best solution by Eqs. (16)–(18)
24: return Y^{best}
25: **end procedure**

4 Results and Discussion

In this section, the results of DFA on 35 benchmark functions are shown. Table 2 presents these 35 benchmark functions, and Fig. 3 displays the 3D representations of some benchmark functions. Among them, F1–F6 are low-dimensional single-peaked functions; F7–F13 are low-dimensional multi-peaked functions; F14–F20 are high-dimensional single-peaked functions; F21–F29 are high-dimensional multi-peaked functions; and F30–F35 are fixed-dimensional functions. The performance of DFA on these functions is compared with that of six well-known algorithms: ABC, FA, GWO, HS, GOA, and WOA.

For the abovementioned metaheuristic algorithms, the same initialization process as DFA is employed. To reduce the effect of randomness on the test results, all algorithms on the benchmark function are executed in 30 independent runs with 500 iterations per run. The average values and standard deviations are obtained after the evaluation of the algorithms' performance.

4.1 Evaluation of Exploitation Capability

Benchmark functions F1–F6 are single-peaked functions since they have only one global optimum. They are mainly used to assess the exploitation capability of metaheuristic algorithms. Table 3 shows the optimization results of DFA and other algorithms. The table shows that DFA yields better optimization results than the other algorithms, indicating that DFA has the best exploitation capability.

Benchmark functions F7–F13 are multi-peaked functions that have many local optima. The multimodal function can well detect the exploration ability of metaheuristic algorithms. Poor performing metaheuristic algorithms can be easily trapped in the local optimal values in the function and thus will not yield global optimization results. Table 4 shows that DFA yields optimal results for six benchmark functions, proving that it has a good exploration capability.

Table 2: The benchmark functions in the study

Function name	Equation	D	Range	f_{min}
Goldstein-Price	$F_1 = \left[1 + (x_1 + x_2 + 1)^2 \begin{pmatrix} 19 - 14x_1 + \\ 3x_1^2 - \\ 14x_2 + \\ 6x_1x_2 + 3x_2^2 \end{pmatrix}\right] * \left[30 + (2x_1 - 3x_2)^2 \begin{pmatrix} 18 - 32x_1 + 12x_1^2 \\ +48x_2 - 36x_1x_2 \\ +27x_2^2 \end{pmatrix}\right]$	2	$[-2, 2]$	3
Branin	$F_2 = \left(x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos(x_1) + 10$	2	x_1 $[-5, 10]$ x_2 $[0, 15]$	0.397887357
Bohachevsky 1	$F_3 = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$	2	$[-50, 50]$	0
Easom	$F_4 = -\cos(x_1)\cos(x_2)\exp\left[-(x_1 - \pi)^2 - (x_2 - \pi)^2\right]$	2	$[-10, 10]$	-1
Beale	$F_5 = (1.5 - x_1 + x_1x_2)^2 + \left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x_1 + x_1x_2^3\right)^2$	2	$[-4.5, 4.5]$	0
Bartels Conn	$F_6 = \lvert x_1^2 + x_2^2 + x_1x_2\rvert + \lvert\sin(x_1)\rvert + \lvert\cos(x_2)\rvert$	2	$[-500, 500]$	1
Shekel's Foxholes	$F_7 = \left(\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + \sum_{i=1}^{2}\left(x_i - a_{ij}\right)^6}\right)^{-1}$ $a_{ij} = \begin{bmatrix} -32 & -16 & 0 & 16 & -32 & -16 \\ -32' & -32' & -32' & -32' & -16' & -16' \\ 0 & 16 & 32 & -32 & -16 & \ldots\ 32 \\ -16' & -16' & -16' & 0\ ' & 0\ ' & \ldots' 32 \end{bmatrix}$	2	$[-65.536, 65.536]$	0.9980038378
Six-Hump Camel-Back	$F_8 = 4x_1^2 - 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	-1.031628453
Michalewicz	$F_9 = -\sum_{i=1}^{D}\sin(x_i)\sin^{2m}\left(\frac{ix_i^2}{\pi}\right), m = 10$	2	$[0, \pi]$	-0.801303410

(Continued)

Table 2: Continued

Function name	Equation	D	Range	f_{min}
Schaffer	$F_{10} = 0.5 + \frac{\sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{\left[1 + 0.001\left(x_1^2 + x_2^2\right)\right]^2}$	2	[−100, 100]	0
Drop Wave	$F_{11} = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5\left(x_1^2 + x_2^2\right) + 2}$	2	[−5.12, 5.12]	−1
Shubert	$F_{12} = \left[\sum_{i=1}^{5} i\cos\left(i+1\right)x_1 + i\right] * \left[\sum_{i=1}^{5} i\cos\left(i+1\right)x_2 + i\right]$	2	[−10, 10]	−186.7309
Bird	$F_{13} = (x_1 - x_2)^2 + \sin(x_1)\exp\left(\left[1 - \cos(x_2)\right]^2\right) + \cos(x_2)\exp\left(\left[1 - \sin(x_1)\right]^2\right)$	2	$[-2\pi, 2\pi]$	−106.7645367
Sphere	$F_{14} = \sum_{i=1}^{D} x_i^2$	30	[−100, 100]	0
Schwefel P2.22	$F_{15} = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	30	[−10, 10]	0
Rosenbrock	$F_{16} = \sum_{i=1}^{D-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right]$	30	[−30, 30]	0
Quartic (De-Jong)	$F_{17} = \sum_{i=1}^{D}\left(ix_i^4\right) + \text{random}[0, 1)$	30	[−1.28, 1.28]	0
Schwefel P1.2	$F_{18} = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$	30	[−100, 100]	0
Zakharov	$F_{19} = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{D} 0.5ix_i\right)^4$	30	[−5, 10]	0
Elliptic (Ellipsoid)	$F_{20} = \sum_{i=1}^{D}\left(10^6\right)^{\frac{i-1}{D-1}} x_i^2$	30	[−100, 100]	0
Rastrigin	$F_{21} = \sum_{i=1}^{D}\left[x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right]$	30	[−5.12, 5.12]	0

(Continued)

Table 2: Continued

Function name	Equation	D	Range	f_{min}
Griewank	$F_{22} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D}\left[\cos\left(\frac{x_i}{\sqrt{i}}\right)\right] + 1$	30	[−600, 600]	0
Alpine	$F_{23} = \sum_{i=1}^{D} \lvert x_i \sin(x_i) + 0.1x_i \rvert$	30	[−10, 10]	0
Levy and Montalvo 1	$F_{24} = \frac{\pi}{D}\left[\begin{array}{c} 10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i-1)^2 \\ *\left[1 + 10\sin^2(\pi y_{i+1})\right] \\ +(y_D-1)^2 \end{array}\right]$, $y_i = 1 + \frac{1}{4}(x_i+1)$	30	[−10, 10]	0
Levy and Montalvo 2	$F_{25} = 0.1\left\{\begin{array}{c} \sin^2(3\pi x_1) + \sum_{i=1}^{D}(x_i-1)^2 \\ *\left[1+\sin^2(3\pi x_i+1)\right] \\ +(x_D-1)^2\left[1+\sin^2(2\pi x_D)\right] \end{array}\right\}$	30	[−5, 5]	0
Xin-She Yang 6	$F_{26} = \left[\sum_{i=1}^{D}\sin^2(x_i) - \exp\left(-\sum_{i=1}^{D} x_i^2\right)\right] * \exp\left(-\sum_{i=1}^{D}\sin^2\sqrt{\lvert x_i \rvert}\right)$	30	[−10, 10]	−1
Salomon	$F_{27} = 1 - \cos\left(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}\right) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$	30	[−100, 100]	0
Sinusoidal	$F_{28} = -\left\{\begin{array}{c} 2.5\prod_{i=1}^{D}\sin\left(x_i - \frac{\pi}{6}\right) \\ +\prod_{i=1}^{D}\sin\left[5\left(x_i - \frac{\pi}{6}\right)\right] \end{array}\right\}$	30	[0, π]	−3.5
Schwefel P2.26	$F_{29} = -\sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	30	[−500, 500]	−418.98288D

(Continued)

Table 2: Continued

Function name	Equation	D	Range	f_{min}
Kowalik	$F_{30} = \sum_{i=1}^{11}\left[a_i - \frac{x_1\left(b_i^2 + b_i x_2\right)}{b_i^2 + b_i x_3 + x_4}\right]^2$ $b = \left[4, 2, 1, \frac{1}{2}, \frac{1}{4}, \frac{1}{6}, \frac{1}{8}, \frac{1}{10}, \frac{1}{12}, \frac{1}{14}, \frac{1}{16}\right]$ $a = [0.1957, 0.1947, 0.1735, 0.1600, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246]$	4	[−5, 5]	0.0003074861
Hartmann 3	$F_{31} = -\sum_{i=1}^{m} c_i \exp\left(-\sum_{j=1}^{n} a_{ij}\left(x_j - p_{ij}\right)^2\right)$, $m = 4, n = 3, a = \begin{bmatrix} 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3.0 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$, $c = \begin{bmatrix} 1.0 & 1.2 & 3.0 & 3.2 \end{bmatrix}, p = \begin{bmatrix} 0.36890, 0.1170, 0.2673 \\ 0.46990, 0.4387, 0.7470 \\ 0.10910, 0.8732, 0.5547 \\ 0.03815, 0.5743, 0.8828 \end{bmatrix}$	3	[0, 1]	−3.862782148
Shekel 7	$F_{32} = -\sum_{i=1}^{m} \frac{1}{c_i + \sum_j^D \left(x_j - a_{ij}\right)^2}, m = 7,$ $a = \begin{bmatrix} 4.0, 4.0, 4.0, 4.0 \\ 1.0, 1.0, 1.0, 1.0 \\ 8.0, 8.0, 8.0, 8.0 \\ 6.0, 6.0, 6.0, 6.0 \\ 3.0, 7.0, 3.0, 7.0 \\ 2.0, 9.0, 2.0, 9.0 \\ 5.0, 5.0, 3.0, 3.0 \end{bmatrix} c = \begin{bmatrix} 0.1, 0.2, 0.2, 0.4, 0.6 \end{bmatrix}$	4	[0, 10]	−10.1527
Paviani	$F_{33} = \sum_{i=1}^{D}\left[\ln^2\left(x_i - 2\right) + \ln^2\left(10 - x_i\right)\right] - \left(\prod_{i=1}^{D} x_i\right)^{0.2}$	10	[2.001, 9.999]	−45.7784684

(Continued)

Table 2: Continued

Function name	Equation	D	Range	f_{min}
Powell's Quartic	$F_{34} = (x_1 + 10x_2)^2 + 5(x_3 + x_4)^2 + (x_2 + 2x_3)^4 + 10(x_1 + 10x_4)^4$	4	[−10, 10]	0
Colville	$F_{35} = [100(x_1 - x_2)]^2 + (1 - x_1)^2 + \&90\left(x_4 - x_3^2\right)^2 + (1 - x_3)^2 + 10.1\begin{bmatrix}(x_2 - 1)^2 + \\ (x_4 - 1)^2\end{bmatrix} + 19.8(x_2 - 1)(x_4 - 1)$	4	[−10, 10]	0

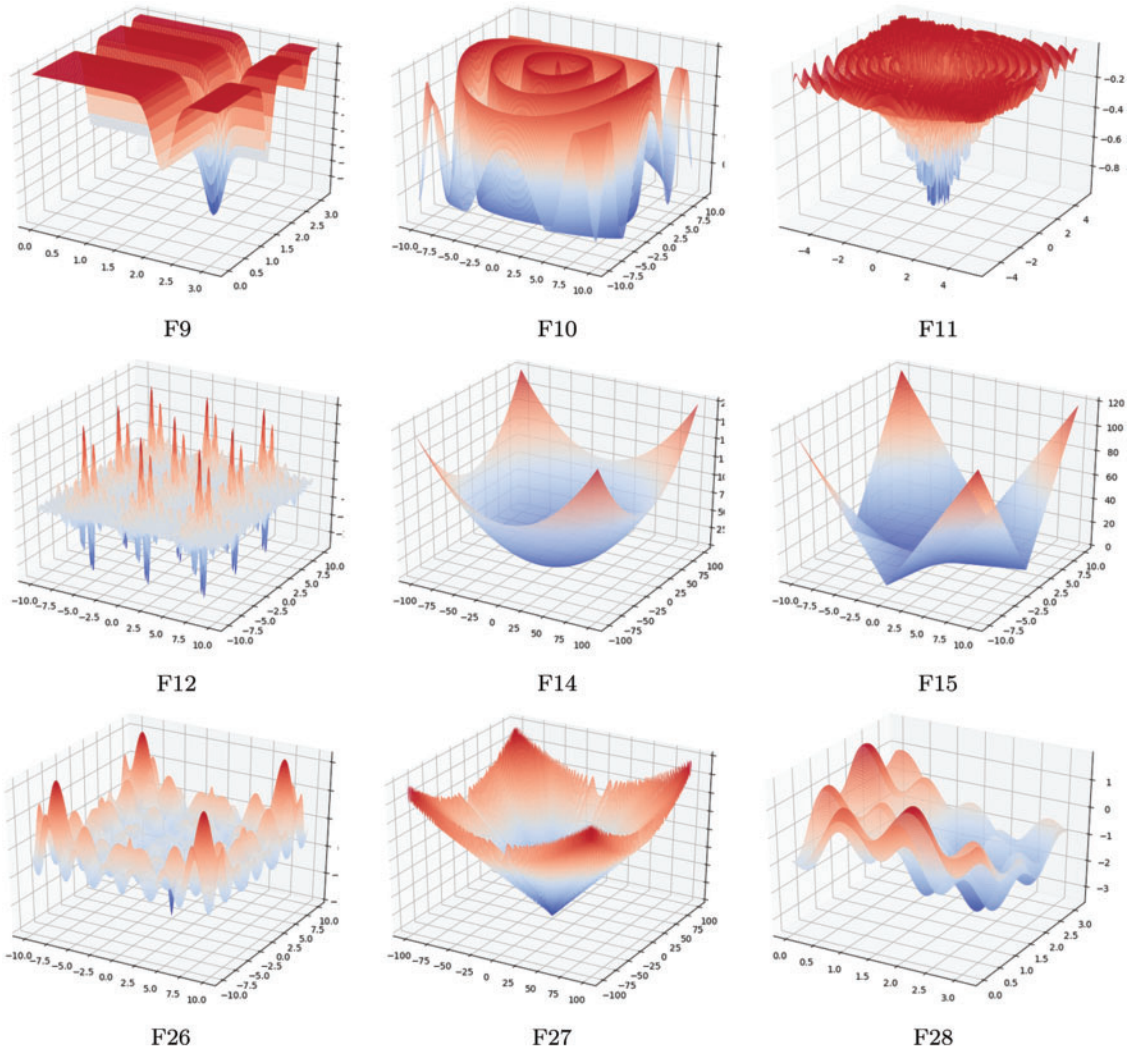

Figure 3: 3D representations of some benchmark functions

Table 3: Optimization results of low-dimensional single-peaked functions

		DFA	ABC	FA	GWO	HS	GOA	WOA
F1	Mean	3.00	3.016132	3.000004	3.000028	8.400001	3.00	3.00
	Std.	1.30E−15	0.026696	7.94E−06	5.48E−05	11.16211	1.26E−12	5.11E−15
F2	Mean	0.397887	0.398291	0.397888	0.397889	0.397887	0.397887	0.397887
	Std.	4.30E−16	5.59E−4	1.85E−06	5.42E−06	7.89E−10	2.77E−14	1.16E−15
F3	Mean	0.00	0.231584	2.37E−05	0.00	5.01E−05	2.36E−11	0.00
	Std.	0.00	0.2265062	5.45E−05	0.00	7.89E−05	2.73E−11	0.00
F4	Mean	−1.00	−0.99941	−1.00	−1.00	0.9667	−1.00	−1.00
	Std.	0.00	1.20E−03	9.68E−07	1.29E−05	2.42E−01	2.63E−13	3.93E−17

(Continued)

Table 3: Continued

		DFA	ABC	FA	GWO	HS	GOA	WOA
F5	Mean	0.00	0.00041	1.37E−07	0.101609	0.065531	0.127012	0.00
	Std.	0.00	7.54E−04	3.05E−07	2.77E−01	2.38E−01	2.97E−01	0.00
F6	Mean	1.00	3.478734	4370.4147	1.00	1.116218	1.00	1.00
	Std.	0.00	3.964094	6136.6486	0.00	1.13E−01	1.78E−06	7.85E−17

Table 4: Optimization results of low-dimensional multi-peaked functions

		DFA	ABC	FA	GWO	HS	GOA	WOA
F7	Mean	0.998004	0.999009	0.998004	2.967961	0.998004	1.031138	1.527099
	Std.	2.78E−17	2.57E−03	7.14E−07	2.918416	3.90E−11	2.41E−01	9.55E−01
F8	Mean	−1.031628	−1.030659	−1.031628	−1.031628	−1.031628	−1.031628	−1.031628
	Std.	5.81E−16	1.56E−03	894E−07	1.87E−08	1.44E−09	2.36E−13	4.68E−16
F9	Mean	−1.801303	−1.800267	−1.801303	−1.801301	−1.801303	−1.755019	−1.715869
	Std.	8.90E−16	1.47E−03	2.46E−06	6.03E−06	2.00E−10	2.15E−01	2.48E−01
F10	Mean	0.002456	0.003090	0.002456	0.002456	0.014192	0.002456	0.009323
	Std.	0.00	4.60E−03	2.27E−08	2.69E−08	1.93E−02	4.66E−14	1.44E−02
F11	Mean	−0.997875	−0.971810	−0.999995	−0.980874	−0.905380	−1.00	−0.970248
	Std.	1.54E−02	2.80E−02	1.55E−05	2.96E−02	7.10E−02	1.35E−12	3.18E−02
F12	Mean	−186.7309	−186.6006	−186.7124	−186.7198	−186.7309	−186.7309	−186.7309
	Std.	2.13E−14	2.63E−01	4.44E−02	4.32E−02	1.48E−05	5.71E−10	3.26E−14
F13	Mean	−106.7645	−106.7416	−106.7645	−106.7645	−106.7645	−104.1707	−106.1161
	Std.	4.63E−14	2.99E−02	2.33E−04	1.73E−04	1.70E−07	7.06E+00	4.71E+00

Compared to F1–F6, benchmark functions F14–FF20 have increased dimensionality from 2 to 30 dimensions, and thus, their difficulty of exploitation is dramatically higher. Table 5 shows that DFA is highly competitive with other metaheuristic algorithms and yields the best or second best optimization results for most of the benchmark functions.

Table 5: Optimization results of high-dimensional single-peaked functions

		DFA	ABC	FA	GWO	HS	GOA	WOA
F14	Mean	0.092286	473.31073	45712.240	3.06E−84	0.541061	6204.1981	3.01E−98
	Std.	5.96E−02	6.12E+02	1.07E+04	1.69E−83	3.82E−01	3.59E+03	1.55E−97
F15	Mean	0.461559	2.681364	1.69E+12	1.54E−54	0.197023	69.192744	5.36E−61
	Std.	3.67E−01	1.97E+00	8.32E+12	4.56E−54	1.07E−01	3.66E+01	3.63E−60
F16	Mean	289.0261	613.0397	2.40E+08	28.63934	333.6808	2665196.3	27.61793
	Std.	2.51E+02	8.65E+02	4.15E+07	1.55E−01	6.60E+02	2.39E+06	5.96E−01
F17	Mean	0.098006	0.773343	113.1723	0.000606	0.113039	3.053331	0.002122

(Continued)

Table 5: Continued

		DFA	ABC	FA	GWO	HS	GOA	WOA
	Std.	4.52E−02	7.81E−01	1.65E+01	5.14E−04	4.86E−02	1.94E+00	2.49E+03
F18	Mean	2471.722	5467.711	77849.98	8727.620	3786.828	9466.5033	33888.18
	Std.	8.14E+02	2.17E+03	2.33E+04	3.27E+03	1.16E+03	5.05E+03	8.87E+03
F19	Mean	48.6756	266.4841	963.9342	133.7175	57.1907	258.12907	489.6096
	Std.	2.05E+01	4.34E+01	6.46E+01	6.75E+01	2.12E+01	1.02E+02	1.07E+02
F20	Mean	0.023375	12.942495	984.33553	1.86E−87	0.002295	32.495835	5.38E−99
	Std.	2.56E−02	2.08E+01	3.88E+02	9.79E−87	2.08E−03	2.40E+01	3.74E−98

With increasing dimensions, the number of optima in the multimodal function exponentially increases. The optimization results of F21–F29 in Table 6 shows that DFA yields superior optimization results for the multimodal functions, even in the case of high dimensions. Even if the optimal result cannot be obtained, DFA yields a result close to the optimal.

Table 6: Optimization results of high-dimensional multi-peaked functions

		DFA	ABC	FA	GWO	HS	GOA	WOA
F21	Mean	0.000411	1.550200	167.22363	0.00	0.005283	200.2107	8.29E−16
	Std.	2.82E−04	3.52E+00	2.12E+01	0.00	3.44E−03	6.42E+01	1.93E−15
F22	Mean	0.317133	175.69147	594.55982	0.000338	1.059683	62.931834	0.002985
	Std.	1.06E−01	2.28E+01	7.17E+01	2.46E−03	2.43E−02	3.06E+01	6.64E−03
F23	Mean	0.360416	0.920308	62.836655	2.20E−54	0.015389	24.326460	15.101463
	Std.	3.53E−01	5.43E−01	5.93E+00	1.03E−53	1.18E−02	6.25E+00	1.32E+01
F24	Mean	0.020931	0.064906	22.508559	0.039639	6.34E−05	4.132649	1.129130
	Std.	5.80E−02	1.25E−01	3.83E+00	1.81E−02	7.11E−05	2.14E+00	1.85E+00
F25	Mean	0.011110	0.181606	4.894935	0.232352	0.001197	2.444549	0.369730
	Std.	1.05E−02	2.99E−01	9.85E−01	1.32E−01	2.41E−03	1.25E+00	4.82E−01
F26	Mean	5.17E−16	1.76E−13	7.89E−08	−0.066666	6.01E−16	2.43E−11	5.35E−13
	Std.	4.65E−16	9.66E−14	9.06E−08	2.92E−01	5.62E−16	3.10E−11	5.25E−13
F27	Mean	2.009873	13.113274	22.730518	0.093215	1.780434	9.256540	0.239882
	Std.	3.14E−01	1.82E+00	3.04E+00	2.91E−02	2.99E−01	1.89E+00	1.39E−01
F28	Mean	−3.498860	−1.060149	−0.012483	−1.272104	−2.083275	−0.090718	−3.149778
	Std.	7.94E−04	6.29E−01	6.58E−02	4.85E−01	1.24E+00	1.58E−01	5.92E−01
F29	Mean	−12077.65	−7918.84	−2253.11	−8623.52	−12544.84	−6294.25	−8926.81
	Std.	4.45E+02	6.00E+02	5.28E+02	1.01E+03	3.47E+01	4.29E+02	1.01E+03

Fixed-dimensional functions usually comprise several low-dimensional functions, which greatly test the exploitation and exploration abilities of metaheuristic algorithms. Metaheuristic algorithms that balance the exploitation and exploration capabilities could obtain optimization results with better

chance. Table 7 shows that DFA achieves the best overall optimization results and has balanced development and exploration capabilities.

Table 7: Optimization results of fixed-dimensional functions

		DFA	ABC	FA	GWO	HS	GOA	WOA
F30	Mean	0.000426	0.001370	0.001605	0.000839	0.010827	0.005892	0.000558
	Std.	2.95E−04	5.74E−04	5.32E−04	2.06E−03	1.79E−02	8.89E−03	6.12E−04
F31	Mean	−3.862782	−3.861855	−3.862654	−3.862318	−3.862651	−3.862782	−3.862782
	Std.	2.49E−15	1.00E−03	3.42E−04	2.24E−03	9.51E−04	2.29E−13	1.90E−15
F32	Mean	−8.312070	−7.740637	−6.395885	−9.572651	−5.361369	−5.553447	−6.807390
	Std.	3.10E+00	2.30E+00	1.58E+00	1.89E+00	3.27E+00	3.38E+00	3.52E+00
F33	Mean	−45.77847	−45.05114	−19.19658	−40.97316	−45.77847	−42.02	−45.77846
	Std.	2.20E−10	1.22E+00	5.66E+00	3.06E+00	2.45E−08	4.14E+00	5.82E−05
F34	Mean	0.001695	0.002405	0.037276	0.001066	0.003754	1.32E−05	0.001757
	Std.	6.01E−03	2.76E−03	6.70E−02	3.26E−03	6.64E−03	2.60E−05	3.55E−03
F35	Mean	5.206178	23.54695	16.389712	10.087391	108.71767	0.780104	23.194003
	Std.	2.00E+01	3.84E+01	2.19E+01	3.43E+01	1.76E+02	2.17E+02	1.21E+02

Overall, DFA shows best performance compared to other algorithms in the low-dimensional single-peaked, low-dimensional multi-peaked, and fixed-dimensional functions; and also achieves better results than most algorithms in the high-dimensional functions.

4.2 Statistical Analysis

Although the optimization results show that DFA exhibits better overall performance than other metaheuristic algorithms, the superior performance of DFA needs to be demonstrated in statistical analysis. This section uses the classic statistical analysis method, i.e., the rank-sum ratio (RSR) [46], to conduct statistical analysis. The RSR values embody information on all evaluation indicators, showing the comprehensive level of these indicators. The large RSR value indicates the merit of the algorithm. The optimization mean of the algorithm on the benchmark functions is used as an evaluation indicator. The analysis results of RSR are given in Table 8, which shows that DFA exhibits better overall performance than other metaheuristic algorithms.

Table 8: Rank-sum ratio result

Algorithm	RSR ranking	Probit	RSR fitted values	Grading level
DFA	1	6.802743090739191	0.9912724183765839	6
GWO	2	6.067570523878141	0.8917274765228165	5
WOA	3	5.565948821932863	0.8238061402222359	5
ABC	4	5.1800123697927045	0.7715489921466842	4

(Continued)

Table 8: Continued

Algorithm	RSR ranking	Probit	RSR fitted values	Grading level
HS	5	4.819987630207295	0.722800380754713	4
FA	7	3.9324294761218583	0.6026218963785807	3
GOA	6	4.434051178067137	0.6705432326791614	3

4.3 Convergence Analysis

Fig. 4 shows the convergence curves of DFA along with those of other algorithms on some of the benchmark functions. The figure shows that DFA usually converges quickly at the beginning and the convergence speed starts slowing down shortly after the beginning; thus, the overall convergence rate of DFA is not fast, which is attributed to DFA's conservative exploitation strategy. However, the continuous update of DFA during subsequent iterations enables a preferable exploration capability that prevents DFA from falling into the local optimum. In most cases, DFA converges to the optimum at 1/3 of the iterations.

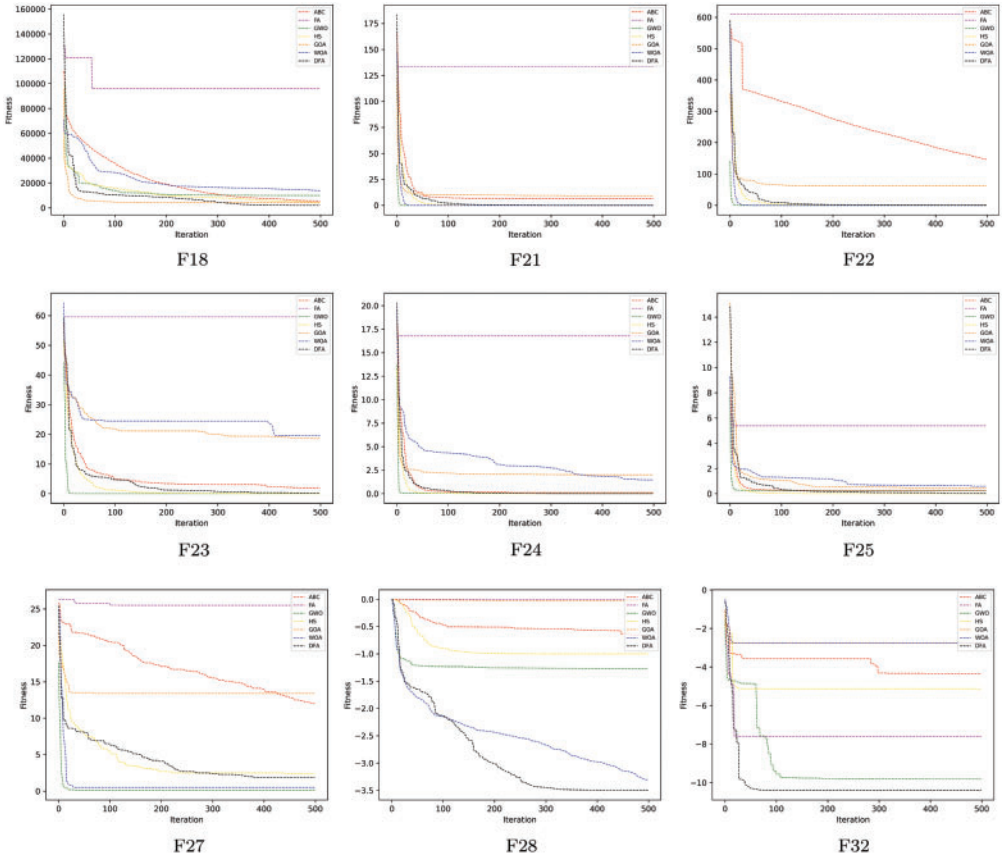

Figure 4: Convergence curves of DFA and comparison algorithms on some benchmark functions

5 DFA for Engineering Projects

In this section, DFA is applied to five constrained engineering design problems: welded beam design, pressure vessel design, three-bar truss design, compression spring design, and cantilever beam design. All the algorithms are tested for each engineering project in 30 independent runs with 500 iterations per run and the best value is taken as the final optimization result.

5.1 Welded Beam Design

Welded beam design is a common engineering optimization problem where the objective is to find the optimum length of the clamped bar l, height of the bar t, thickness of the bar b, and weld thickness h of a beam bar to minimize the manufacturing cost of a welded beam (Fig. 5). The cost of a welded beam is formulated as

$$\min f\left(\vec{x}\right) = 1.10471x_1^2x_2 + 0.04811x_3x_4\left(14 + x_2\right)$$

$$\vec{x} = \left[x_1 \quad x_2 \quad x_3 \quad x_4\right] = \left[h \quad l \quad t \quad b\right]$$

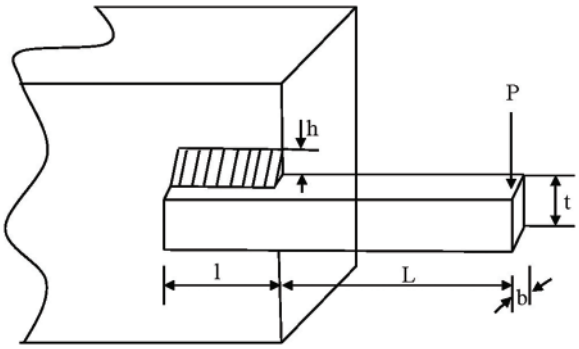

Figure 5: Diagram of welded beam design

Subject to

$$g_1\left(\vec{x}\right) = \tau\left(\vec{x}\right) - \tau_{\max} \le 0,$$

$$g_2\left(\vec{x}\right) = \sigma\left(\vec{x}\right) - \sigma_{\max} \le 0,$$

$$g_3\left(\vec{x}\right) = x_1 - x_4 \le 0,$$

$$g_4\left(\vec{x}\right) = 0.10471x_1^2 + 0.04811x_3x_4\left(14 + x_2\right) - 5 \le 0,$$

$$g_5\left(\vec{x}\right) = 0.125 - x_1 \le 0,$$

$$g_6\left(\vec{x}\right) = \delta\left(\vec{x}\right) - \delta_{\max} \le 0,$$

$$g_7\left(\vec{x}\right) = P - P_c\left(\vec{x}\right) \le 0$$

Variable range

$$0.1 \le x_1 \le 2,\ 0.1 \le x_2 \le 10,\ 0.1 \le x_3 \le 10,\ 0.1 \le x_4 \le 2$$

The constraints and their associated constants are expressed as follows:

$$\tau\left(\vec{x}\right) = \sqrt{\left(\tau'\right)^2 + 2\tau'\tau''\frac{x_2}{2R} + \left(\tau''\right)^2}$$

$$\tau' = \frac{P}{\sqrt{2}x_1x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + x_2/2\right)$$

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

$$J = 2\left[\sqrt{2}x_1x_2\left(\frac{x_2^2}{12} + \frac{\left(x_1 + x_3\right)^2}{4}\right)\right]$$

$$\sigma\left(\vec{x}\right) = \frac{6PL}{x_4x_3^2},$$

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

$P = 6000\text{lb}, L = 14\,\text{in}, \ E = 30 \times 10^6\text{psi}$

$G = 12 \times 10^6\text{psi}, \tau_{\max} = 13,600\text{psi},$

$\sigma_{\max} = 30{,}000\text{psi}, \delta_{\max} = 0.25\,\text{in}$

The results of DFA are compared with those of the other algorithms, and the data in Table 9 show that DFA yields the lowest manufacturing cost, indicating that DFA has the potential to solve the welded beam design problem.

Table 9: Comparison results for welded beam design

Algorithm	Optimum variables				Optimum cost
	h	l	t	b	
DFA	0.2053263	3.47659665	9.04334824	0.20569611	1.72595568
FA	0.26841149	3.15279017	7.65984043	0.29365304	2.10712421
ABC	0.19403412	3.66950515	9.22029126	0.2075474	1.77937345
GWO	0.19870343	3.64395848	9.00965781	0.20696315	1.74176404
HS	0.25534331	3.12713781	7.63445357	0.28824088	2.03847263
GOA	0.18912920	3.85154748	9.03662298	0.20572969	1.74982934
WOA	0.18719115	7.44065593	8.89750281	0.2122135	2.23569138

5.2 Pressure Vessel Design

DFA is employed for the pressure vessel design problem. This design aims to find the appropriate shell ($T_s = x_1$), thickness of the head ($T_h = x_2$), inner radius ($R = x_3$), and length of the shell ($L = x_4$) to minimize the total material cost, incorporating four constraints: T_s and T_h are integer multiples of 0.625 and R and L are continuous variables. Fig. 6 shows the dimensions of the pressure vessel structure. The cost of the pressure vessel design is formulated as

$$\min f\left(\vec{x}\right) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3$$

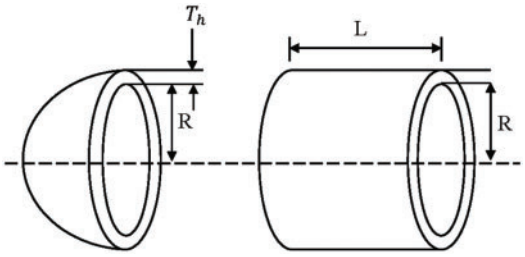

Figure 6: Diagram of pressure vessel design

Subject to

$$\begin{aligned} g_1 &= -x_1 + 0.0193x_3 && \leq 0, \\ g_2 &= -x_2 + 0.00954x_3 && \leq 0, \\ g_3 &= -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 && \leq 0, \\ g_4 &= x_4 - 240 && \leq 0, \end{aligned}$$

Variable range

$$0 \leq x_1 \leq 99, 0 \leq x_2 \leq 99,$$

$$10 \leq x_3 \leq 200, 10 \leq x_4 \leq 200,$$

Table 10 presents the optimization results of DFA and other algorithms and shows that DFA yields the best optimization results.

Table 10: Comparison results for pressure vessel design

Algorithm	Optimum variables				Optimum cost
	T_s	T_h	R	L	
DFA	12.68830574	6.27191504	41.0895908	189.55214995	5911.15146639
ABC	12.70506601	6.27994045	41.13437792	188.9686432	5914.386945161
FA	15.06349385	7.4636159	48.13021892	114.0638514	7810.28453922
GWO	13.32095422	6.82361653	43.10752296	164.5239233	6038.11905684
HS	13.19592313	6.52452479	42.72899153	169.00003781	5971.21817933
GOA	12.88471723	6.36890596	41.72542605	181.31441634	5933.29373914
WOA	13.4869777	6.66662006	43.67544592	158.02818082	6005.51170785

5.3 Three-Bar Truss Design

The three-bar truss design is the third case study employed. It aims to evaluate the optimum cross-sectional areas $A_1 = A_3 = x_1$ and $A_2 = x_2$ so that the volume of the static load truss structure is minimized and constrained considering the stress (σ). Fig. 7 displays the dimensions of the three-bar truss design. The volume equation of the truss structure is

$$\min f\left(\vec{x}\right) = \left(2\sqrt{2}x_1 + x_2\right) \times H,$$

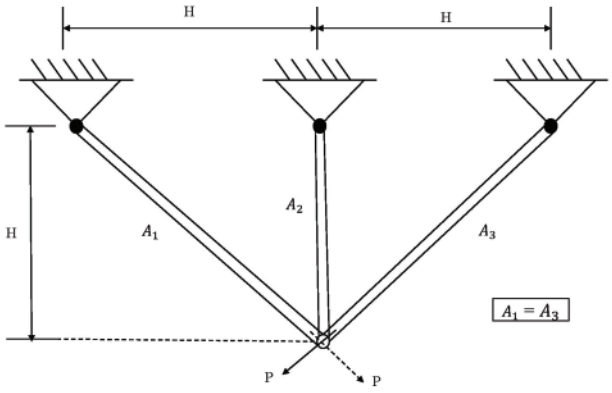

Figure 7: Diagram of pressure vessel design

Subject to

$$g_1 = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0,$$
$$g_2 = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0,$$
$$g_3 = \frac{1}{x_1 + \sqrt{2}x_2}P - \sigma \leq 0,$$

Variable range

$$0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1,$$

$$H = 100\,\text{cm}, P = 2KN/\text{cm}^2, \sigma = 2KN/\text{cm}^2$$

The statistical results of DFA and the other algorithms for the three-bar truss design problem are shown in Table 11, indicating that DFA yields the lowest volume compared to the other optimization algorithms.

Table 11: Comparison results for three-bar truss design

Algorithm	Optimum variables		Optimum volume
	$A_1 = A_3$	A_2	
DFA	0.78875500	0.40802178	263.89578211
ABC	0.78873793	0.40811035	263.89980965
FA	0.78868058	0.40823232	263.89578701
GWO	0.78856879	0.40855575	263.89651005
HS	0.78764908	0.41115752	263.89655385
GOA	0.78881723	0.40784588	263.89579234
WOA	0.7883832	0.40907398	263.89584001

5.4 Compression Spring Design

Compression spring design aims to minimize the mass f(x) under certain constraints, including four inequality constraints of minimum deflection, shear stress, surge frequency, and deflection. Three design variables are present: the mean coil diameter (D), wire diameter (d), and number of active coils (N). Fig. 8 shows the dimensions of the compression spring design. The mass equation of the compression spring is

$$min f(x) = (N + 2)\, Dd^2$$

Subject to

$$g_1(x) = 1 - \frac{D^3 N}{71785 d^4} \leq 0$$

$$g_2(x) = \frac{4D^2 - dD}{12566\left(Dd^3 - d^4\right)} + \frac{1}{5108 d^2} - 1 \leq 0$$

$$g_3(x) = 1 - \frac{140.45 d}{D^2 N} \leq 0$$

$$g_4(x) = \frac{D + d}{1.5} - 1 \leq 0$$

Variable range

$$0.05 \leq d \leq 2,\ 0.25 \leq D \leq 1.3,\ 2 \leq N \leq 15$$

The statistical results of DFA and the other algorithms for the three-bar truss design problem are shown in Table 12. DFA yields the lowest mass, denoting that DFA can well solve the compression spring design problem.

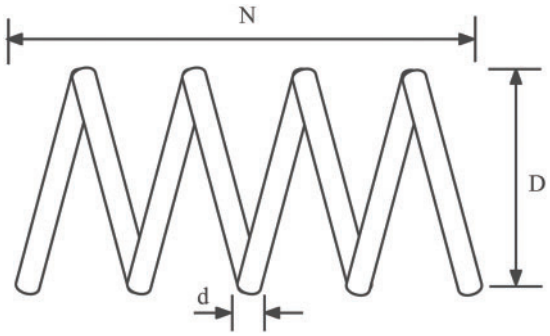

Figure 8: Diagram of compression spring design

Table 12: Comparison results for compression spring design

Algorithm	Optimum variables			Optimum mass
	d	*D*	N	
DFA	0.05180635	0.35954649	11.12500622	0.012665467
ABC	0.05227057	0.35384607	12.16672679	0.013696147
FA	0.05	0.31667258	14.24697868	0.01286243
GWO	0.05298374	0.38866847	9.65846781	0.01272055
HS	0.05581944	0.46445987	6.95555321	0.0129601920
GOA	0.05362217	0.40503082	8.93195867	0.012731378
WOA	0.05126438	0.34658734	11.90853585	0.012668523

5.5 Cantilever Beam Design

Cantilever beam design is a structural engineering design problem related to the weight optimization of the square section cantilever. The cantilever beam is rigidly supported at one end, and a vertical force acts on the free node of the cantilever. The beam comprises five hollow square blocks of constant thickness, the height (or width) of which is the decision variable and the thickness is fixed. Fig. 9 depicts the dimensions of the cantilever beam design. The weight equation of the cantilever beam is

$$f(X) = 0.0624\,(x_1 + x_2 + x_3 + x_4 + x_5)$$

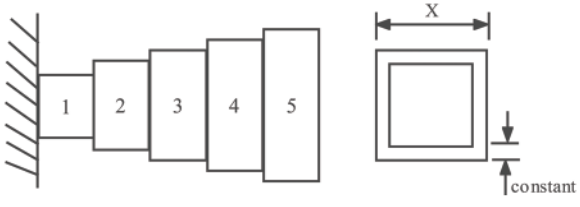

Figure 9: Diagram of cantilever beam design

Subject to

$$g(X) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \leq 0$$

Variable range

$$0.01 \leq x_i \leq 100, i = 1, 2 \cdots, 5$$

The statistical results of DFA and the other algorithms for the cantilever beam design problem are shown in Table 13. The table shows that DFA outperforms all other algorithms except GOA. Although DFA achieves suboptimal optimization results, this result is acceptable according to NFL.

Table 13: Comparison results for cantilever beam design

Algorithm	Optimum variables					Optimum weight
	x_1	x_2	x_3	x_4	x_5	
DFA	5.96871891	5.35265306	4.53393307	3.475844	2.14508805	1.34011719
ABC	6.34433249	5.25405738	4.83394417	3.12836834	2.0932221	1.35120489
FA	5.98319073	5.38116037	4.44248883	3.44473814	2.26317699	1.34302424
GWO	6.18084273	5.11890478	4.66205867	3.36104363	2.19997452	1.34302424
HS	6.11954396	5.39091533	4.42759653	3.46660407	2.08273114	1.34081320
GOA	6.01598408	5.30917278	4.4945209	3.50137541	2.15260648	1.33995636
WOA	9.53170142	4.20385935	3.62127756	12.87413556	3.28738588	2.091545650

6 Conclusion

This study presented a novel metaheuristic algorithm called DFA. The effectiveness of DFA was validated on 35 well-known benchmark functions and compared with that of six well-known metaheuristic algorithms. The optimization capabilities of DFA were examined in terms of exploitation capability, statistical analyses, and convergence analyses. The results indicate that DFA is a competitive metaheuristic algorithm with outstanding performance in terms of global optimization problems.

DFA was also applied to five engineering design problems for verification in practical applications (i.e., welded beam, pressure vessel, three-bar truss, compression spring, and cantilever beam design problems). DFA outperforms the other six metaheuristic algorithms in the chosen evaluation criteria. In subsequent studies, DFA will be used to improve the efficiency of machine learning potential development for Fe–Cr–Al ternary alloys.

Funding Statement: This work is performed under collaboration with College of Materials Science and Chemical Engineering, Harbin Engineering University by the support of National Key Research and Development Program of China (2019YFB1901003). The authors also acknowledge the financial support of National Natural Science Foundation of China (Grants No. 52250005, 21875271, 21707147, 11604346, 21671195, and 51872302), the Key R & D Projects of Zhejiang Province No. 2022C01236, the Zhejiang Province Key Research and Development Program (No. 2019C01060), and the project of the key technology for virtue reactors from NPIC. Entrepreneurship Program of Foshan National Hi-tech Industrial Development Zone.

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

References

[1] S. H. Haji and A. M. Abdulazeez, "Comparison of optimization techniques based on gradient descent algorithm: A review," *PalArch's Journal of Archaeology of Egypt/Egyptology*, vol. 18, no. 4, pp. 2715–2743, 2021.

[2] A. L. Custódio and J. F. A. Madeira, "GLODS: Global and local optimization using direct search," *Journal of Global Optimization*, vol. 62, no. 1, pp. 1–28, 2015.

[3] M. Dorigo and T. Stützle, "Ant colony optimization: Overview and recent advances," in *Handbook of Metaheuristics*, 1st edition, vol. 272. Cham, CH: Springer, Cham, pp. 311–351, 2019.

[4] D. Wang, D. Tan and L. Liu, "Particle swarm optimization algorithm: An overview," *Soft Computing*, vol. 22, no. 2, pp. 387–408, 2018.

[5] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992. https://www.jstor.org/stable/24939139

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

[7] O. Abedinia, N. Amjady and N. Ghadimi, "Solar energy forecasting based on hybrid neural network and improved metaheuristic algorithm," *Computational Intelligence*, vol. 34, no. 1, pp. 241–260, 2018.

[8] O. Kramer, "Genetic algorithms," *Genetic Algorithm Essentials*, vol. 679, pp. 11–19, 2017. https://doi.org/10.1007/978-3-319-52156-5_2

[9] W. Ezra and W. Zhu, "A survey on metaheuristics for optimization in food manufacturing industry," *Applied Soft Computing*, vol. 46, pp. 328–343, 2016.

[10] S. Dan, "Optimization," in *Evolutionary Optimization Algorithms*, 1st edition, vol. 1. Hoboken, New Jersey, Canada: John Wiley & Sons, pp. 11–35, 2013.

[11] C. Liu, "The dark forest," in *The Three-Body Problem*, 1st edition, vol. 1. Chongqing, China: Chongqing Publishing Group, pp. 441–449, 2008.

[12] X. Yao, Y. Liu and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[13] Bilal, M. Pant, H. Zaheer, L. Garcia-Hernandez and A. Abraham, "Differential evolution: A review of more than two decades of research," *Engineering Applications of Artificial Intelligence*, vol. 90, pp. 103479, 2020. https://doi.org/10.1016/j.engappai.2020.103479

[14] N. Hansen, D. V. Arnold and A. Auger, "Evolution strategies," in *Springer Handbook of Computational Intelligence*, 1st edition. Heidelberg, BER, DE: Springer, Berlin, Heidelberg, pp. 871–898, 2015.

[15] M. Abdel-Basset and L. A. Shawky, "Flower pollination algorithm: A comprehensive review," *Artificial Intelligence Review*, vol. 52, no. 4, pp. 2533–2557, 2019.

[16] T. Zhang and Z. W. Geem, "Review of harmony search with respect to algorithm structure," *Swarm and Evolutionary Computation*, vol. 48, pp. 31–43, 2019.

[17] A. Lambora, K. Gupta and K. Chopra, "Genetic algorithm-a literature review," in *2019 Int. Conf. on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, Faridabad, India, pp. 380–384, 2019.

[18] M. Paulinas and A. Ušinskas, "A survey of genetic algorithms applications for image enhancement and segmentation," *Information Technology and Control*, vol. 36, no. 3, pp. 278–284, 2007.

[19] Y. Park, J. Yoo and H. Park, "A genetic algorithm for the vendor-managed inventory routing problem with lost sales," *Expert Systems with Applications*, vol. 53, pp. 149–159, 2016.

[20] E. Hancer, "Artificial bee colony: Theory, literature review, and application in image segmentation," *Recent Advances on Memetic Algorithms and its Applications in Image Processing*, vol. 873, pp. 47–67, 2020.

[21] F. Pourpanah, R. Wang, C. P. Lim, X. Wang and D. Yazdani, "A review of artificial fish swarm algorithms: Recent advances and applications," *Artificial Intelligence Review*, pp. 1–37, 2022. https://doi.org/10.1007/s10462-022-10214-4

[22] S. Mirjalili, L. Aljarah, M. Mafarja, A. A. Heidari and H. Faris, "Grey wolf optimizer: Theory, literature review, and application in computational fluid dynamics problems," *Nature-Inspired Optimizers*, vol. 811, pp. 87–105, 2020.

[23] V. Kumar and D. Kumar, "A systematic review on firefly algorithm: Past, present, and future," *Archives of Computational Methods in Engineering*, vol. 28, no. 4, pp. 3269–3291, 2021.
[24] A. S. Joshi, O. Kulkarni, G. M. Kakandikar and V. M. Nandedkar, "Cuckoo search optimization-a review," *Materials Today: Proceedings*, vol. 4, no. 8, pp. 7262–7269, 2017.
[25] Q. Zhang, S. Du, Y. Zhang, H. Wu, K. Duan *et al.,* "A novel chimp optimization algorithm with refraction learning and its engineering applications," *Algorithms*, vol. 15, no. 6, pp. 189, 2022.
[26] L. Abualigah and A. Diabat, "A comprehensive survey of the grasshopper optimization algorithm: Results, variants, and applications," *Neural Computing and Applications*, vol. 32, no. 19, pp. 15533–15556, 2022.
[27] Y. Zhang, S. Du and Q. Zhang, "Improved slime mold algorithm with dynamic quantum rotation gate and opposition-based learning for global optimization and engineering design problems," *Algorithms*, vol. 15, no. 9, pp. 317–341, 2022.
[28] F. S. Gharehchopogh and H. Gholizadeh, "A comprehensive survey: Whale optimization algorithm and its applications," *Swarm and Evolutionary Computation*, vol. 48, pp. 1–24, 2019.
[29] Y. Liu, and P. Tian, "A Multi-start central force optimization for global optimization," *Applied Soft Computing*, vol. 27, pp. 92–98, 2015.
[30] E. Rashedi, E. Rashedi and H. Nezamabadi-Pour, "A comprehensive survey on gravitational search algorithm," *Swarm and Evolutionary Computation*, vol. 41, pp. 141–158, 2018.
[31] H. Eskandar, A. Sadollah, A. Bahreininejad and M. Hamdi, "Water cycle algorithm–a novel metaheuristic optimization method for solving constrained engineering optimization problems," *Computers & Structures*, vol. 110, pp. 151–166, 2012.
[32] W. Zhao, L. Wang and Z. Zhang, "Atom search optimization and its application to solve a hydrogeologic parameter estimation problem," *Knowledge-Based Systems*, vol. 163, pp. 283–304, 2019.
[33] H. Abedinpourshotorban, S. M. Shamsuddin, Z. Beheshti and D. N. A. JawawI, "Electromagnetic field optimization: A physics-inspired metaheuristic optimization algorithm," *Swarm and Evolutionary Computation*, vol. 26, pp. 8–22, 2016.
[34] F. A. Hashim, E. H. Houssein, M. S. Mabrouk, W. Al-Atabany and S. Mirjalili, "Henry gas solubility optimization: A novel physics-based algorithm," *Future Generation Computer Systems*, vol. 101, pp. 646–667, 2019.
[35] O. Abdalla, H. Rezk and E. M. Ahmed, "Wind driven optimization algorithm based global MPPT for PV system under non-uniform solar irradiance," *Solar Energy*, vol. 180, pp. 429–444, 2019.
[36] F. Zou, D. Chen and Q. Xu, "A survey of teaching–learning-based optimization," *Neurocomputing*, vol. 335, pp. 366–383, 2019.
[37] S. Cheng, Q. Qin, J. Chen and Y. Shi, "Brain storm optimization algorithm: A review," *Artificial Intelligence Review*, vol. 46, no. 4, pp. 445–458, 2016.
[38] A. Fathy and H. Rezk, "Parameter estimation of photovoltaic system using imperialist competitive algorithm," *Renewable Energy*, vol. 111, pp. 307–320, 2017.
[39] A. Maheri, S. Jalili, Y. Hosseinzadeh, R. Khani and M. Miryahyavi, "A comprehensive survey on cultural algorithms," *Swarm and Evolutionary Computation*, vol. 62, pp. 100846, 2021.
[40] M. A. Al-Betar, Z. A. A. Alyasseri, M. A. Awadallah and L. A. Doush, "Coronavirus herd immunity optimizer (CHIO)," *Neural Computing and Applications*, vol. 33, no. 10, pp. 5011–5042, 2021.
[41] T. Wu, X. Wu, J. Chen, X. Chen and A. H. Ashrafzadeh, "A novel metaheuristic algorithm: The team competition and cooperation optimization algorithm," *CMC-Computers Materials & Continua*, vol. 73, no. 2, pp. 2879–2896, 2022.
[42] H. Ma, D. Simon, P. Siarry, Z. Yang and M. Fei, "Biogeography-based optimization: A 10-year review," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 391–407, 2017.
[43] W. Zhao, L. Wang and Z. Zhang, "Artificial ecosystem-based optimization: A novel nature-inspired meta-heuristic algorithm," *Neural Computing and Applications*, vol. 32, no. 13, pp. 9383–9425, 2020.
[44] A. R. Pouya, M. Solimanpur and M. J. Rezaee, "Solving multi-objective portfolio optimization problem using invasive weed optimization," *Swarm and Evolutionary Computation*, vol. 28, pp. 42–57, 2016.

[45] T. Joyce and J. M. Herrmann, "A review of no free lunch theorems, and their implications for metaheuristic optimisation," *Nature-Inspired Algorithms and Applied Optimization*, vol. 744, pp. 27–51, 2018.
[46] Z. Wang, S. Dang, Y. Xing, Q. Li and H. Yan, "Applying rank sum ratio (RSR) to the evaluation of feeding practices behaviors, and its associations with infant health risk in Rural Lhasa, Tibet," *International Journal of Environmental Research and Public Health*, vol. 12, no. 12, pp. 15173–15181, 2015.