Mathematical Biosciences and Engineering

MBE, 20 (12): 20881–20913.
DOI: 10.3934/mbe.2023924
Received: 06 September 2023
Revised: 05 November 2023
Accepted: 06 November 2023
Published: 20 November 2023

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

Research article

A hierarchical chain-based Archimedes optimization algorithm

Zijiao Zhang[1], Chong Wu[1], Shiyou Qu[1,*] and Jiaming Liu[2]

[1] School of Management, Harbin Institute of Technology, Harbin 150000, China
[2] School of Computer Science and Artificial Intelligence, Beijing Technology and Business University, Beijing 100048, China

* **Correspondence:** Email: qushiyou@hit.edu.cn.

Abstract: The Archimedes optimization algorithm (AOA) has attracted much attention for its few parameters and competitive optimization effects. However, all agents in the canonical AOA are treated in the same way, resulting in slow convergence and local optima. To solve these problems, an improved hierarchical chain-based AOA (HCAOA) is proposed in this paper. The idea of HCAOA is to deal with individuals at different levels in different ways. The optimal individual is processed by an orthogonal learning mechanism based on refraction opposition to fully learn the information on all dimensions, effectively avoiding local optima. Superior individuals are handled by an Archimedes spiral mechanism based on Levy flight, avoiding clueless random mining and improving optimization speed. For general individuals, the conventional AOA is applied to maximize its inherent exploration and exploitation abilities. Moreover, a multi-strategy boundary processing mechanism is introduced to improve population diversity. Experimental outcomes on CEC 2017 test suite show that HCAOA outperforms AOA and other advanced competitors. The competitive optimization results achieved by HCAOA on four engineering design problems also demonstrate its ability to solve practical problems.

Keywords: Archimedes optimization algorithm; hierarchical chain; orthogonal learning; Levy flight; refraction opposition-based learning

1. Introduction

Optimization problems that minimize or maximize objective functions are numerous in real life. Most of them are extraordinarily complex and challenging, which leads to the inability of gradient-based deterministic optimization methods to handle them. Because the objective functions of these

problems are hardly guaranteed to be differentiable, these traditional methods easily fall into local optima. Metaheuristic algorithms have flexibility, no gradient mechanism and local optimal avoidance, which make them more popular than deterministic methods. Therefore, metaheuristic algorithms have developed rapidly in the past decades and have made significant achievements in optimization problems in various fields, such as parameter tuning [1,2], feature selection [3,4], scheduling [5,6], system control [7,8] and engineering design [9,10].

Taking inspiration from nature, metaheuristic algorithms search for optimal solutions through iterative stochastic operations that are within a reasonable time. Scholars have divided these algorithms into four groups: evolution-based, swarm-based, physics-based and social or human-based. Representative metaheuristic algorithms in these four categories are listed in Figure 1. To summarize, most of them have the following characteristics: (1) Their update mechanisms are inspired by some phenomena in nature, such as biological behaviors, physical theorems and chemical phenomena. (2) They have two phases: exploration and exploitation. Exploration is the search for unvisited areas to ensure globally optimal solutions. Exploitation is the intensive search for the most probable region based on accumulated experience to enhance the local search. (3) They are all population-based search schemes, and attention needs to be paid to the interaction between individuals (to promote knowledge sharing and improve the quality of solutions) and the population diversity (to explore the unknown space and overcome local optima). (4) Stochastic strategies and proper parameter definitions in algorithms are essential, and appropriate parameter settings can make algorithms better fits for real problems. In addition, the different update mechanisms and stochastic strategies of these algorithms lead to their different exploration and exploitation capabilities. Hence, the main difference between various algorithms is how to strike a balance between exploration and exploitation [11].

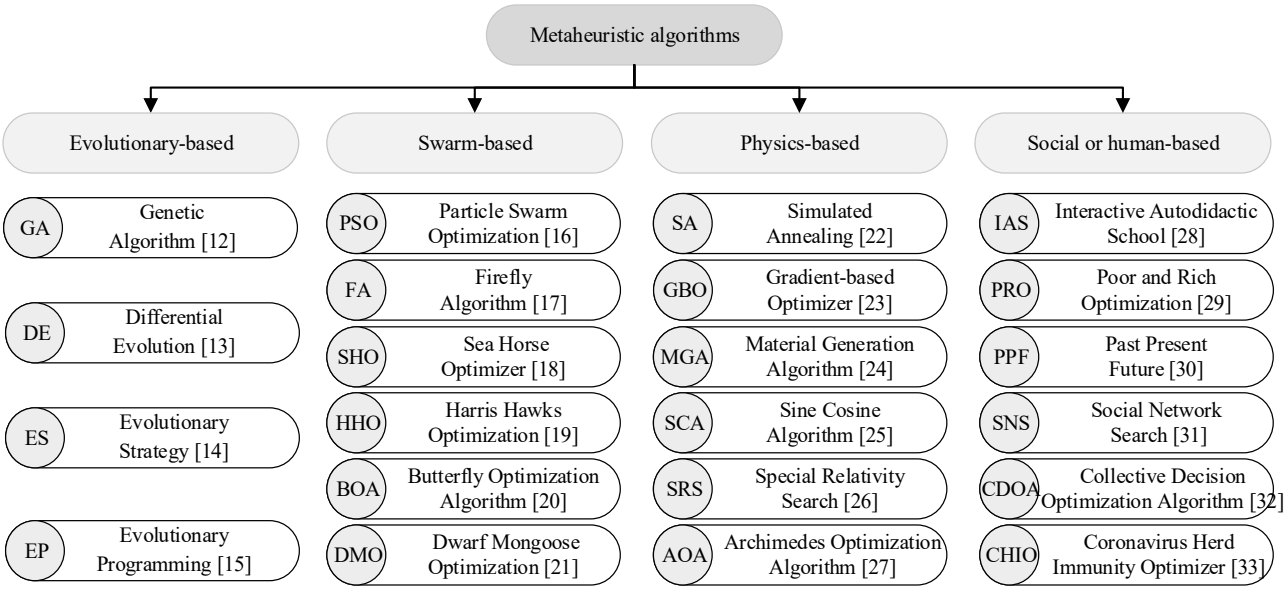

Figure 1. Categories of metaheuristic algorithms.

In summary, there are two possible problems with these algorithms: (1) One problem that all algorithms are likely to encounter is how to balance exploitation and exploration. Too much exploration may lead to slow convergence, and too much exploitation may cause falling into local optima. The update mechanisms of some algorithms, such as particle swarm optimization and grey

wolf optimizer [34], are based on globally optimal individuals, so these algorithms may converge to a local optimum. (2) The "No Free Lunch (NFL)" [35] theorem logically states that there is no metaheuristic algorithm that is most effective in every situation. For these two reasons, researchers are committed to exploring new algorithms or improving existing ones. This paper is also motivated by these two reasons. For convenience, a list of abbreviations is given in Table 1.

Table 1. List of abbreviations.

Abbreviation	Interpretation	Abbreviation	Interpretation
AGWO	Grey wolf optimizer based on Aquila exploration	IHAOAVOA	Improved hybrid Aquila optimizer and African
AOA	Archimedes optimization algorithm		vultures optimization algorithm
AVOA	African vultures optimization algorithm	MAOA	Hybrid Archimedes optimization algorithm
BOA	Butterfly optimization algorithm		enhanced with mutualism scheme
CQFFA	Chaotic quasi-oppositional farmland fertility algorithm	MCWOA	Multi-cohort whale optimization algorithm
CSOAOA	Enhanced hybrid arithmetic optimization algorithm	MFO	Moth-flame optimization algorithm
DAQUILA	Improved Aquila algorithm	MGA	Material generation algorithm
DMO	Dwarf mongoose optimization	OBL	Opposition-based learning
ECSOA	Elastic collision seeker optimization algorithm	PRO	Poor and rich optimization algorithm
FA	Firefly algorithm	PSO	Particle swarm optimization
GBO	Gradient-based optimizer:	QLGCTSA	Improved tunicate swarm algorithm
GJO	Golden jackal optimization	ROBL	Refraction opposition-based learning
GWO	Grey wolf optimizer	SCA	Sine cosine algorithm
HAO	Heterogeneous Aquila optimization algorithm	SCSO	Sand cat swarm optimization
HCAOA	Hierarchical chain-based Archimedes optimization	SHO	Sea horse optimizer
	algorithm	SNS	Social network search
HHO	Harris hawks optimization algorithm	SRS	Special relativity search
IDARSOA	Individual disturbance and attraction repulsion strategy	STD	Standard deviation
	enhanced seagull optimization algorithm	TQA	Termite queen algorithm
IGWO	Improved grey wolf optimizer	WOA	Whale optimization algorithm

This paper focuses on the Archimedes optimization algorithm (AOA) proposed by Hashim et al. based on physical laws. Inspired by Archimedes' buoyancy principle, AOA generates new solutions by imitating collisions between objects in a liquid. AOA has few parameters, simple interfaces, easy implementation and good optimization effects. As a result, AOA has been extensively applied to practical problems with satisfactory results. Abdelbadie et al. [36] used AOA to clarify the best conditions of a proportional integral regulator, which controls the charge and discharge of superconducting magnetic energy storage systems. Wang et al. [2] employed AOA to tune the main parameters in variational mode decomposition to perform better in diagnostic fault characteristics for both simulated and real situations. Neggaz and Fizazi [37] proposed a wrapper approach based on AOA for feature selection to automatically identify the best facial region features for human gender. Balakrishnan et al. [38] utilized AOA to adjust the parameters in the radial basis function neural network (RBFNN) to construct a merged algorithm, AOA-RBFNN, to achieve higher competence, and it is applied to engineering optimization problems. Annrose [39] applied AOA to improve the classification accuracy of multiple single long short-term memory (LSTM) networks and thus proposed a hybrid depth learning model for bean disease classification.

Like other metaheuristic algorithms, AOA has some drawbacks on some specific issues. All agents in the canonical AOA are handled in the same way, which leads to immature convergence and a tendency to get stuck at local optima, especially when dealing with complex optimization problems. To compensate for these shortcomings, an improved HCAOA is proposed. On the basis of the original AOA, this algorithm combines an orthogonal learning mechanism based on refraction opposition and an Archimedes spiral mechanism based on Levy flight to deal with individuals at different levels. Specifically, the main contributions of this paper are as follows:

• An ameliorative variant of AOA is presented to handle global optimization problems. The idea of hierarchical chains is introduced into AOA and different update strategies are implemented for agents at different levels to enhance optimization capability.

• Orthogonal learning mechanism based on refraction opposition is suggested in AOA, which fully learns information on all dimensions of the optimal individual and effectively avoids AOA falling into local optimum.

• Archimedes spiral integrated Levy flight is introduced into AOA to achieve an extensive range of random disturbances in searching space, thus improving the search ability of AOA.

• Computational Experimental Competition (CEC) 2017 suite and four engineering design problems are employed to evaluate the comprehensive performance of HCAOA.

The outline of the remainder of this article is listed here: Section 2 presents the conventional AOA. Section 3 details the modification and the framework of HCAOA. The feasibility of HCAOA is validated by the CEC 2017 suite in Section 4. The optimization outcomes of HCAOA on engineering design problems are presented in Section 5. Section 6 summarizes the study and presents research ideas for the future.

2. The conventional AOA

AOA treats objects in a fluid as candidate agents, and each object has its density, volume, acceleration and position. The location of an individual represents a possible solution and is updated by adjusting its density, volume and acceleration. As with other metaheuristic algorithms, the optimization process of AOA contains two parts: exploration and exploitation. In the exploration stage, collisions with random individuals are implemented to diversify populations. During the exploitation phase, there is no collision between objects, and the optimal individual is learned to facilitate local search capability. Its mathematical steps in detail are as follows:

Step 1: Initialize. The initial positions, densities, volumes and accelerations of all individuals are generated randomly by Eqs (1)–(4).

$$x_i^1 = lb + rand \times \left(ub - lb\right), \tag{1}$$

$$den_i^1 = rand, \tag{2}$$

$$vol_i^1 = rand, \tag{3}$$

$$acc_i^1 = lb + rand \times \left(ub - lb\right), \tag{4}$$

where the subscript i denotes the i -th individual, and the superscript t indicates the t -th iteration. The x_i^1, den_i^1, vol_i^1 and acc_i^1 represent the position, density, volume and acceleration of the i -th agent in

the first iteration, respectively. The ub and lb are the upper and lower limits of the solution set, respectively. The parameter $rand$ is a D -dimensional random vector generated uniformly in $[0,1]$, and D is the dimension of solutions. Then, all initial individuals are evaluated and the optimal individual is selected. The parameter of the best one is expressed as x_{best}, den_{best}, vol_{best} and acc_{best}, correspondingly.

Step 2: Update volumes and densities. The volume and density of the i -th individual in the next generation are updated according to Eqs (5) and (6):

$$vol_i^{t+1} = vol_i^t + rand \times \left(vol_{best} - vol_i^t \right), \tag{5}$$

$$den_i^{t+1} = den_i^t + rand \times \left(den_{best} - den_i^t \right). \tag{6}$$

Step 3: Update the transfer operator TF^t and density factor d^t. The transfer operator determines whether the search behavior changes from exploration to exploitation. The density factor contributes to the search scope from global to local. Their values vary with iteration t using Eqs (7) and (8). The $t_{\max}$ refers to the maximum iteration number.

$$TF^t = \exp\left(\frac{t - t_{\max}}{t_{\max}} \right), \tag{7}$$

$$d^t = \exp\left(\frac{t_{\max} - t}{t_{\max}} \right) - \left(\frac{t}{t_{\max}} \right). \tag{8}$$

Step 4: Update accelerations. Different formulas are applied to update accelerations depending on the search phase. When $TF^t < 0.5$, the search process is in the exploration stage. There is a collision between the i -th individual and a random one mr, and its acceleration is updated according to Eq (9).

$$acc_i^{t+1} = \frac{den_{mr}^{t+1} + vol_{mr}^{t+1} \times acc_{mr}^{t+1}}{den_i^{t+1} \times vol_i^{t+1}}. \tag{9}$$

When $TF^t \geq 0.5$, it is in the exploitation stage. There is no collision between agents, and the acceleration of the i -th individual is updated by Eq (10).

$$acc_i^{t+1} = \frac{den_{best} + vol_{best} \times acc_{best}}{den_i^{t+1} \times vol_i^{t+1}}. \tag{10}$$

Then, the updated accelerations are standardized according to Eq (11).

$$acc_{i-norm}^{t+1} = u \times \frac{acc_i^{t+1} - \min\left(acc\right)}{\max\left(acc\right) - \min\left(acc\right)} + l, \tag{11}$$

where u is set to 0.9 and l is set to 0.1. The max(acc) and min(acc) are the maximum and minimum of all acc values, respectively.

Step 5: Renew positions. When $TF^t < 0.5$, the position of the i -th individual is updated by Eq (12).

$$x_i^{t+1} = x_i^t + C_1 \times rand \times acc_{i-norm}^{t+1} \times d^{t+1} \times \left(x_{rand} - x_i^t \right), \tag{12}$$

where C_1 is set to 2 and x_{rand} is the location of a randomly selected individual.

Otherwise, the position of the i -th individual in the exploitation stage is updated by Eq (13).

$$x_i^{t+1} = x_{best} + F \times C_2 \times rand \times acc_{i-norm}^{t+1} \times d^{t+1} \times \left(T \times x_{best} - x_i^t\right), \tag{13}$$

where C_2 is a constant equal to 6. The parameters T and F are calculated by Eqs (14) and (15).

$$T = C_3 \times TF^{t+1}, \tag{14}$$

$$F = \begin{cases} 1 & if\ P \le 0.5 \\ -1 & if\ P > 0.5 \end{cases}, \tag{15}$$

where $P = 2 \times rand - C_4$. The parameters C_3 and C_4 are set to 2 and 0.05, respectively.

Step 6: Evaluate all individuals. Evaluate all individuals and select the optimal one. Assign x_{best}, den_{best}, vol_{best} and acc_{best}. The process of AOA in detail is shown in Figure 2.

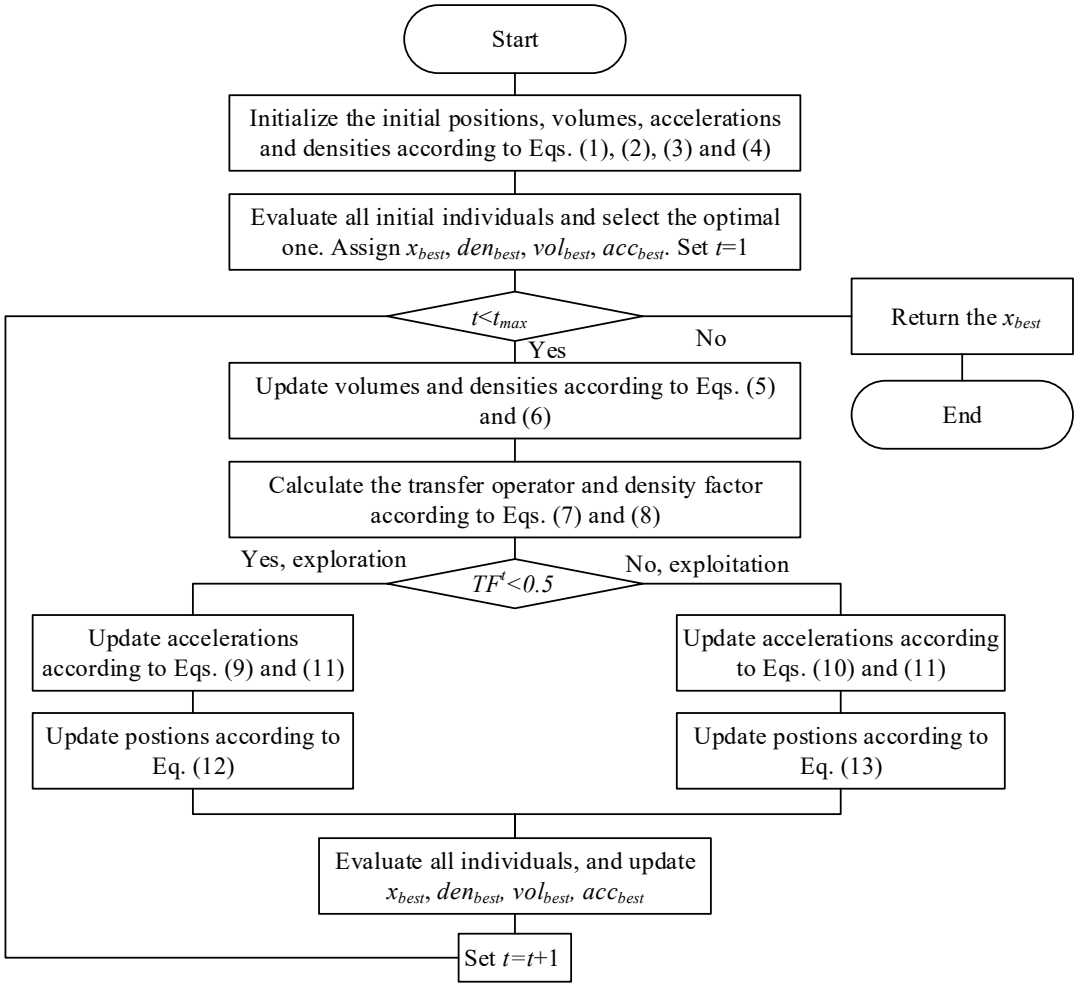

Figure 2. The flowchart of AOA.

3. The proposed HCAOA

In the canonical AOA, all individuals are treated in the same way. During the exploration period, the positions of individuals are updated by collisions with random individuals. As a result, the knowledge of better individuals is not well learned, slowing down optimization speed. In the exploitation phase, all individuals move towards the optimal one. This causes the aggregation of all individuals to the best one and makes the algorithm get stuck at local optima. Moreover, the whole process of the standard AOA does not fully use the optimal individual, leading to slow convergence speed. Finally, the same boundary treatment for all individuals somewhat reduces population diversity. To address these problems, an improved HCAOA is put forward. On the basis of the original AOA, the algorithm combines an orthogonal learning mechanism based on refraction opposition and an Archimedes spiral mechanism based on Levy flight to deal with individuals at different levels. In addition, a multi-strategy boundary processing mechanism is introduced.

3.1. Orthogonal learning mechanism based on refraction opposition

3.1.1. Refraction opposition-based learning

In metaheuristic algorithms, all individuals move towards the optimal individual, causing the disappearance of population variety and the fall into a local optimum. To overcome this shortcoming, opposition-based learning (OBL) has been proposed [40] and widely used [7]. OBL is a greedy policy that selects a point with better fitness between the initial and the opposite point.

Definition 1 Opposite number. Suppose there exists a number x in $[lb, ub]$, then its opposite number $\bar{x}$ is obtained by Eq (16).

$$\bar{x} = lb + ub - x. \tag{16}$$

Definition 2 Opposite spot. Assume $X = (x_1, \ x_2, \ \cdots \ x_D)$ is a D-dimensional spatial point, $x_1, x_2, \cdots, x_D \in R$ and $x_j \in [lb_j, ub_j]$ $(j = 1, 2, \cdots D)$. Then, its opposite spot is calculated by Eq (17).

$$\overline{x_j} = lb_j + ub_j - x_j. \tag{17}$$

In addition, several variants have been derived, such as elite opposition-based learning [41], refraction opposition-based learning (ROBL) [10], quasi opposition-based learning [5] and random opposition-based learning [3]. ROBL is a dynamic oppositional learning strategy based on OBL and the lens imaging principle to help algorithms find better candidate solutions.

The process of obtaining refraction opposition points is shown in Figure 3. There is an object N with height h directly on the coordinate x, $x \in [lb, ub]$. Place a lens at the midpoint o of lb and ub, and the height h^* of mirror point N^* can be obtained based on the lens imaging principle.

The refraction-opposition point is x^*, which is calculated through Eq (18).

$$\frac{(lb + ub)/2 - x}{x^* - (lb + ub)/2} = \frac{h}{h^*}. \tag{18}$$

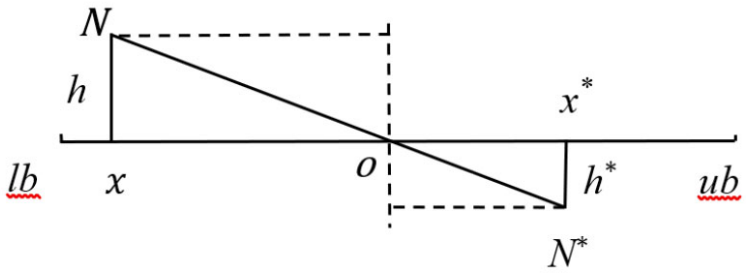

Figure 3. The refraction opposition-based learning.

Let $h / h^* = k$, where k is the scaling factor. The refraction-opposition spot x^* is obtained by Eq (19).

$$x^* = \frac{ub + lb}{2} + \frac{ub + lb}{2k} - x / k. \tag{19}$$

By generalizing to D dimensional space, the refraction-opposition spot is attained by Eq (20).

$$x_j^* = \frac{ub_j + lb_j}{2} + \frac{ub_j + lb_j}{2k} - x_j / k. \tag{20}$$

When $k = 1$, the refraction opposition solution x^* in Eq (20) is the opposite point in OBL. The opposite point obtained by the OBL strategy is fixed, while the refraction-position point obtained by ROBL dynamically changes when k is set to different values. The reason for using ROBL in this paper is that ROBL provides a variety of solutions due to the randomness of k.

3.1.2. Orthogonal learning

There is a better one when comparing two individuals, but there is no guarantee that this better individual outperforms the other in all dimensions. Each individual has several superior dimensions. Mining better dimensions from each individual is expected to yield a better individual. This requires permutation and combination experiments. However, the number of experiments in traditional permutations grows exponentially as the dimensionality increases, which is unsuitable for optimization algorithms with multiple iterations and individuals in high dimensions. Therefore, orthogonal learning is introduced to find the optimal combination through a few experiments. The orthogonal learning experiment consists of two steps: orthogonal design and factor analysis [42].

Orthogonal design defines the content and number of experiments through predefined orthogonal arrays. Orthogonal arrays provide a series of different combinations that are represented as $L_M(S^Q)$, where S is layers number, Q is factors number and M is test times. For example, for the experiment with 7 factors and 2 levels, $128(2^7)$ trials are required to get an optimal combination when adopting permutation. However, if orthogonal experimental design is employed, the best combination can be found through $8(L_8(2^7))$ trials in Eq (21).

$$L_8\left(2^7\right)=\begin{bmatrix}1&1&1&1&1&1&1\\1&1&1&2&2&2&2\\1&2&2&1&1&2&2\\1&2&2&2&2&1&1\\2&1&2&1&2&1&2\\2&1&2&2&1&2&1\\2&2&1&1&2&2&1\\2&2&1&2&1&1&2\end{bmatrix}. \tag{21}$$

Factor analysis assesses the influence of every level of different factors on the outcomes of M trials. For example, the objective function is the spherical function $f(x)=\sum_i x_i^2$. Two individuals are $A=[1,3,5,2,3,1,2]$ and $B=[2,1,3,1,2,2,4]$. Orthogonal learning is carried out by Eq (21). Table 2 shows the process of obtaining a new individual x_{new} through factor analysis. The A_i and B_i denote the cumulative fitness value of the i -th factor in the individual A and B, respectively. A smaller cumulative value of an individual on a dimension indicates that dimension value of the individual is the dominant dimension value. All the predominant dimension values are combined to generate a new individual.

Table 2. The process of obtaining a new individual by orthogonal learning.

Times	Factor							Fitness value
	1	2	3	4	5	6	7	
1	1(1)	3(1)	5(1)	2(1)	3(1)	1(1)	2(1)	53
2	1(1)	3(1)	5(1)	1(2)	2(2)	2(2)	4(2)	60
3	1(1)	1(2)	3(2)	2(1)	3(1)	2(2)	4(2)	44
4	1(1)	1(2)	3(2)	1(2)	2(2)	1(1)	2(1)	21
5	2(2)	3(1)	3(2)	2(1)	2(2)	1(1)	4(2)	47
6	2(2)	3(1)	3(2)	1(2)	3(1)	2(2)	2(1)	40
7	2(2)	1(2)	5(1)	2(1)	2(2)	2(2)	2(1)	46
8	2(2)	1(2)	5(1)	1(2)	3(1)	1(1)	4(2)	57
A_i	178	200	216	190	194	178	160	----
B_i	190	168	152	178	174	190	208	----
x_{new}	1	1	3	1	2	1	2	21

3.1.3. The proposed orthogonal learning based on ROBL

In metaheuristic algorithms, the optimal individual plays an essential role. The best one determines the convergence speed. In the exploration phase of AOA, all individuals update their positions through random collisions, which reduces the optimization speed. In the exploitation phase of AOA, all individuals converge to the optimal individual and may fall into a local optimum. Therefore, this paper applies an orthogonal learning mechanism based on refraction opposition to improve convergence speed and the capacity to escape from a local extremum.

This approach employs the ROBL mechanism to generate refraction opposition solutions for the optimal individual. The randomness of k ensures the diversity of refraction opposition solutions, thus reducing the chances of the algorithm immersing a local extremum. In traditional ROBL strategy, the

optimal individual is compared with its refraction opposition solution in a greedy way, and the superior individual is selected for the next generation. This does not take full advantage of the information available on all dimensions of the two individuals because one of the two is inevitably superior to the other in some dimensions and inferior to the other in the remaining dimensions. To resolve this issue, orthogonal learning is utilized to fully learn the information of the optimal individual and its refraction opposition individual, combining the dominant dimensions of both to produce a better solution.

3.2. Archimedes spiral mechanism based on Levy flight

An Archimedes spiral describes the trajectory of a point moving away from a stationary point at an invariant velocity and revolving around the fixed point at an unchanging angular velocity. Its polar coordinate r is defined by Eq (22).

$$r = a + b\theta\,, \tag{22}$$

where a is the distance from the point of departure to the coordinate origin in polar coordinates, b is the increment of the unit angle of a helix and θ is the rotation angle. Altering a means revolving the spiral and b determines the distance between two adjacent curves. In this paper, we combine Levy flight and Archimedes spiral to discover better solutions in the neighborhood of superior individuals.

The target regions at different stages are different. In the exploration stage, some individuals explore the area around dominant individuals to learn more about them and avoid clueless random search, thus improving optimization speed. Therefore, the update formula in this stage is Eq (23).

$$x_i^{t+1} = x_i^t + \left|x_i^t - x_{levy}^t\right| \times l \times \cos\left(2\pi l\right). \tag{23}$$

During the exploitation phase, the areas near the best individual are focused on searching to enhance optimization ability and avoid local optima. Currently, the locations are updated as in Eq (24).

$$x_i^{t+1} = x_{best} + \left|x_{best} - x_{levy}^t\right| \times l \times \cos\left(2\pi l\right), \tag{24}$$

where l stands for a uniform random number in [–1,1]. Then a in Archimedes spiral corresponds to x_i^t in Eq (23) and x_{best} in Eq (24), and b in Archimedes spiral corresponds to $\left|x_i^t - x_{levy}^t\right| \times l$ in Eq (23) and $\left|x_{best} - x_{levy}^t\right| \times l$ in Eq (24). The $\cos\left(2\pi l\right)$ in Eqs (23) and (24) corresponds to θ in Archimedes spiral. The generated Levy flight solution x_{levy}^t is calculated by Eq (25).

$$x_{levy}^t = \frac{\mu}{\left|\nu\right|^{\frac{1}{\beta}}} \times \left(x_i^t - x_{best}^t\right), \tag{25}$$

where μ and ν conform to a normal distribution, correspondingly, $\mu \sim N(0, \sigma_\mu^2)$, $\nu \sim N(0, \sigma_\nu^2)$. σ_ν is 1 and σ_μ is obtained by Eq (26).

$$\sigma_{\mu}=\left\{\frac{\Gamma(1+\beta)\times\sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left[(1+\beta)/2\right]\times\beta\times2^{\frac{\beta-1}{2}}}\right\}^{\frac{1}{\beta}}, \tag{26}$$

where the value range of β is typically $(0,2]$, and β is taken as 1 in this paper.

Levy flight adopts a random search method combining small steps and long jumps, which can effectively expand the search area. Archimedes spiral mechanism helps to search the neighborhood of excellent individuals, avoiding missing part of the solution space and ensuring mining meticulousness to the maximum extent. In the exploration stage, the integration of the two means allows the proposed algorithm to not only fully learn from the excellent individuals, but also to avoid clueless random mining. In addition, this combination is applied in the exploitation stage to fully explore the neighborhood of the best individual. This can ensure the rigor and accuracy of the search process to enhance local search ability, and improve population diversity to avoid premature convergence phenomena.

3.3. Multi-strategy boundary processing mechanism

In the standard AOA, all individuals are treated with the same boundary processing method at all stages. If the values of new individuals are greater than the upper limits, they are set to the upper limits. Similarly, the values less than the lower bounds are set to the lower bounds. This is the most common boundary processing mechanism in previous algorithms. However, this may affect the diversity of populations to some extent, especially in the exploration phase. There may be multiple individuals who exceed the boundary values in the same dimension and are set to the boundary values. Therefore, this study proposes a multi-strategy boundary processing mechanism that treats individuals differently at different stages. In the exploration phase, the dimension values of individuals outside their range are set to random numbers to increase population diversity. When it comes to the exploitation stage, the traditional boundary treatment is continued. The pseudo-code of the multi-strategy boundary processing mechanism is algorithm 1.

Algorithm 1: The pseudo code of multi-strategy boundary processing mechanism.

Update the positions of all individuals $xnew_{ij}$

```
if TF^t < 0.5      # exploration phase
    for i = 1 to popsize
        for j = 1 to dim
            if xnew_ij > ub_j or xnew_ij < lb_j
                xnew_ij = lb_j + (ub_j − lb_j) × rand
            end if
        end for
    end for
end if
if TF^t ≥ 0.5      # exploitation phase
    for i = 1 to popsize
        for j = 1 to dim
```

13: if $xnew_{ij} > ub_j$
14: $xnew_{ij} = ub_j$
15: end if
16: if $xnew_{ij} < lb_j$
17: $xnew_{ij} = lb_j$
18: end if
19: end for
20: end for
21: end if
22: return $xnew_{ij}$

3.4. The frame of HCAOA

All individuals are treated in the same manner in the standard AOA, which does not efficiently learn the information of the optimal and better individuals, thus reducing the convergence speed and possibly leading to a local optimum. Therefore, the HCAOA is proposed in this paper.

All individuals in a population are classified into three classes according to their fitness: the optimal individual, superior individuals and general individuals. The optimal individual is treated through the orthogonal learning mechanism based on refraction opposition, which effectively avoids local optima and improves convergence speed to a certain extent. Superior individuals are processed by Archimedes spiral mechanism based on Levy flight to conduct information mining from better individuals. This avoids clueless random search and improves optimization speed during the exploration period, while ensuring population diversity and reducing the probability of local optimality during the exploitation period. For general individuals, the conventional AOA is applied to effectively utilize its exploration and exploitation capabilities. Among them, the optimal individual is the one with the best fitness in a population, and there is only one. The general individuals make up $a\%$ of a population. The rest are superior individuals, and the number is $[(1-a\%)\times popsize-1]$. Finally, the multi-strategy boundary processing mechanism is employed to increase population diversity. In accordance with the above analysis, its pseudocode is listed in algorithm 2 in detail. Moreover, the flowchart of HCAOA is depicted in Figure 4.

Algorithm 2: The pseudo code of HCAOA algorithm.

1: Input: the proportion of general individuals $a\%$, the population size N, maximum iterations $t_{\max}$ and the parameters C_1, C_2, C_3, C_4
2: Initialize the positions, densities, volumes and accelerations of all individuals according to Eqs (1)–(4)
3: Set iteration $t = 1$
4: While $t < t_{\max}$ do
5: Evaluate the fitness of all individuals and rank them. Reassign serial numbers to all individuals in sorted order
6: Record the optimal individual of the whole population
7: Update volumes and densities of all individuals though Eqs (5) and (6)
8: Update transfer operator and density factor by Eqs (7) and (8)
9: For $k = 1$ do # The optimal individual
10: Calculate the refraction opposition solution of the optimal individual by formula (20). Perform orthogonal learning mechanism to update the position
11: End for

12: For $k = 2$ to $[(1-a\%)\times N]$ do # The superior individuals
13: If $TF^t < 0.5$
14: Update positions by Eq (23)
15: Else
16: Update positions by Eq (24)
17: End if
18: End for
19: For $k = [(1-a\%)\times N+1]$ to N do # The general individuals
20: If $TF^t < 0.5$
21: Update accelerations by Eqs (9) and (11)
22: Update positions by Eq (12)
23: Else
24: Update accelerations by Eqs (10) and (11)
25: Update positions by Eq (13)
26: End if
27: End for
28: Boundary processing according to Algorithm 1
29: Evaluate the fitness values of each individual before and after the update, and select the better one to enter the next generation
30: Set $t = t+1$
31: End while

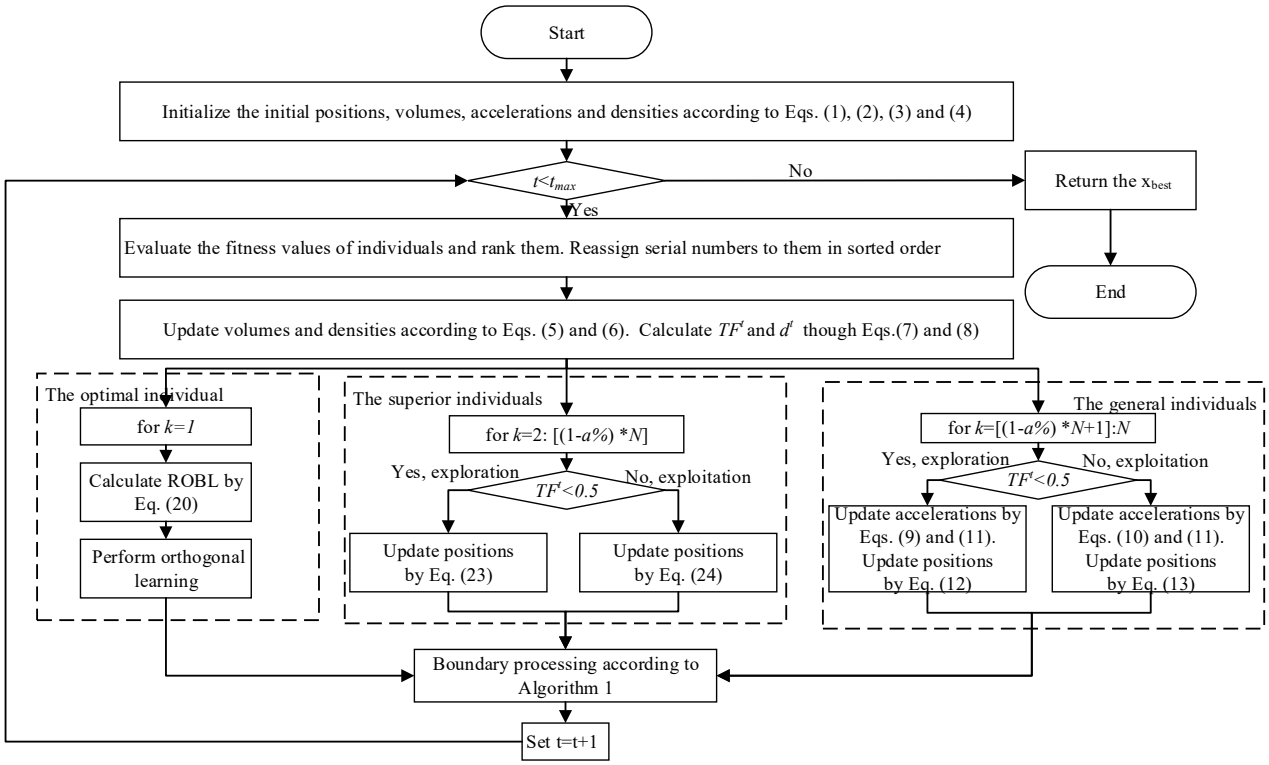

Figure 4. The flowchart of HCAOA.

3.5. Computational complexity

In the worst scenario, the time complexity of HCAOA is calculated based on its pseudocode. Here, N is population size, D is problem dimension and $t_{\max}$ is max iterations.

It takes $O(N \times D)$ time to initialize the population. The assessment of the fitness values of all individuals needs $O(N \times t_{\max})$ time. Sorting according to the fitness values takes $O(N^2 \times t_{\max})$ time. It takes $O(D^2 \times t_{\max})$ time to implement the orthogonal learning strategy based on refraction opposition for the optimal individual. It takes $O((1-a\%) \times N \times D \times t_{\max})$ time to perform Archimedes spiral mechanism based on Levy flight for superior individuals. It takes $O(a\% \times N \times D \times t_{\max})$ time to perform AOA for general individuals. The multi-strategy boundary processing mechanism needs $O(N \times D \times t_{\max})$ time. It takes $O(N \times t_{\max})$ time to carry out greedy selection technique in HCAOA.

On the whole, the total time of HCAOA is $O(N \times D \times t_{\max} + N^2 \times t_{\max})$. Thus, in terms of time complexity, HCAOA is more complex than the original AOA.

3.6. Parameters sensitivity analysis

In the new HCAOA, five parameters C_1, C_2, C_3, C_4 and $a\%$ need to be specified. Among them, C_1, C_2, C_3 and C_4 are the parameters in the traditional AOA, and $a\%$ is a new parameter that indicates the proportion of general agents in a population. In the proposed HCAOA, the parameter $a\%$ is insensitive to these four parameters, so these four parameters are deferred to the values in the original AOA. Multiple tests are executed to judge the parameter $a\%$ in this subsection. Sensitivity analysis is performed on four benchmark functions (f1, f10, f11 and f29) picked from various classes of the CEC 2017 test suite. The dimensionality is set to 30. The values of the parameter $a\%$ are assigned to $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. The outcomes of different $a\%$ are offered in Table 3. It is evident that when parameter $a\%$ is set to 80%, HCAOA realizes excellent results.

Table 3. The experimental results of HCAOA using different *a%* value.

a%	10%	20%	30%	40%	50%	60%	70%	80%	90%
f1	1.25×10^6	1.51×10^6	5.78×10^4	3.59×10^3	3.21×10^3	3.35×10^3	3.54×10^3	**2.66×10^3**	6.07×10^3
f10	4.26×10^3	4.24×10^3	4.43×10^3	4.17×10^3	4.39×10^3	4.42×10^3	4.23×10^3	**4.13×10^3**	4.20×10^3
f11	1.30×10^3	1.28×10^3	1.25×10^3	1.24×10^3	1.25×10^3	1.23×10^3	1.24×10^3	**1.21×10^3**	1.23×10^3
f29	3.91×10^3	3.80×10^3	3.78×10^3	3.69×10^3	3.70×10^3	3.74×10^3	3.69×10^3	**3.67×10^3**	3.71×10^3

4. Experimental results

The optimization capability of HCAOA is examined on the CEC 2017 test suite, and the effects are weighed against nine other classical or new metaheuristics. First, the specific knowledge of the CEC 2017 test suite and parameter settings of different algorithms are presented. Next, the effectiveness of the various strategies in HCAOA are rated. Finally, the proposed HCAOA is evaluated from various perspectives, including qualitative analysis, statistical analysis, stability analysis, convergence analysis and statistical tests.

4.1. Parameter setting and CEC 2017 test suite

The optimization ability of HCAOA is further assessed with the CEC 2017 suite, which is a universal measurement for modern algorithms. There are four types of problems in this suite: unimodal (f1−f3), multimodal (f4−f10), hybrid (f11−f20) and composite cases (f21−f30). As f2 has been removed from this test suite due to its instability, 29 test functions with distinct modalities and complexity are employed to test HCAOA and other competing algorithms. All functions take values in the range [−100,100]. In conclusion, CEC 2017 suite is quite sophisticated and appropriate for studying algorithms' exploration and exploitation abilities.

The proposed HCAOA is compared with AOA, PSO, GWO, SCA, FA, WOA, HHO, BOA and SHO to evaluate the optimization performance of these algorithms from both qualitative and quantitative perspectives. Their parameters are demonstrated in Table 4. It is rational and appropriate to regulate parameters to default values [43]. The dimensions of these testing functions are typed to 30, the population size is 100 and the maximal iteration is 1000. Since the results of these algorithms are randomized, they run separately 30 times to ensure the fairness and objectivity of competitions.

Table 4. Parameter settings of HCAOA and all other algorithms.

Algorithms	Parameter setting
AOA[27]	$C_1 = 2, C_2 = 6, C_3 = 2, C_4 = 0.5$
PSO[16]	$v_{\max} = 10, v_{\min} = -10, c1 = 2, c2 = 2, w = 0.8$
GWO[34]	Convergence constant $a = [0,2]$
SCA[25]	$A = 2$
FA[17]	$\beta_0 = 1, \gamma = 1, \alpha = 0.2, \alpha_\delta = 0.98, m = 2$
WOA[9]	α descends proportionally from 2 to 0, a_2 linearly reduces from −1 to −2
HHO[19]	$\beta = 1.5$
BOA[20]	$c = 0.01, a = 0.1$ *to* $0.3, p = 0.8$
SHO[18]	$r1 = 0, r2 = 0.1$
HCAOA	$C_1 = 2, C_2 = 6, C_3 = 2, C_4 = 0.5, a\% = 80\%$

4.2. Impact of strategies

To overcome the shortcomings of the traditional AOA, this paper proposes an improved HCAOA. There are three improvement strategies in HCAOA. The first strategy is to perform the orthogonal learning mechanism based on refraction opposition on the optimal individual. The second is that superior individuals are handled by Archimedes spiral mechanism based on Levy flight. The third strategy is to implement a multi-strategy boundary processing mechanism for all individuals. To evaluate the impact of each strategy, this subsection fuses each of the three strategies into the AOA and constructs three improved versions: AOA-S1, AOA-S2 and AOA-S3. The means and standard deviations (STD) obtained by AOA and its improved versions on CEC 2017 test functions are listed in Table 5. The optimal means on all functions are shown in bold.

Comparing AOA-S1 with AOA, it is clear that AOA-S1 achieves lower means than AOA, except for f18 and f21. The STD values obtained by AOA-S1 are smaller than those obtained by AOA in most cases. These indicate that the first strategy improves the optimization ability of AOA. The optimization results of AOA-S2 are in comparison with those of AOA. AOA-S2 offers lower means on 24 functions

and significant improvements on several functions, suggesting that the second strategy enhances the search capability to some extent. AOA-S3 slightly outperforms AOA in terms of averages on 23 functions, and the differences between the two are not great for the other 6 functions. The third strategy is a slight improvement over AOA. The main reason is that the third strategy does not change the exploitation and exploration strategy of AOA, but simply increases population diversity through boundary treatments. Finally, HCAOA with all three strategies applied simultaneously produces more competitive results than the use of any one strategy alone.

Table 5. Comparison results of AOA, AOA-S1, AOA-S2, AOA-S3 and HCAOA.

Function	AOA		AOA-S1		AOA-S2		AOA-S3		HCAOA	
Measure	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
f1	2.83×10^{10}	6.03×10^{9}	1.01×10^{10}	6.07×10^{9}	3.72×10^{3}	3.56×10^{3}	2.68×10^{10}	6.82×10^{9}	$\mathbf{2.66 \times 10^{3}}$	2.78×10^{3}
f3	5.29×10^{3}	2.80×10^{3}	3.98×10^{3}	1.85×10^{3}	3.02×10^{4}	1.20×10^{4}	$\mathbf{4.83 \times 10^{3}}$	1.85×10^{3}	1.51×10^{4}	6.62×10^{3}
f4	7.04×10^{3}	2.15×10^{3}	2.32×10^{3}	2.12×10^{3}	5.08×10^{2}	1.40×10^{1}	5.98×10^{3}	1.71×10^{3}	$\mathbf{5.03 \times 10^{2}}$	1.74×10^{1}
f5	6.84×10^{2}	1.94×10^{1}	6.75×10^{2}	2.90×10^{1}	$\mathbf{5.50 \times 10^{2}}$	1.08×10^{1}	6.79×10^{2}	1.46×10^{1}	5.62×10^{2}	1.25×10^{1}
f6	6.49×10^{2}	4.16	6.35×10^{2}	8.42	6.04×10^{2}	9.39×10^{-1}	$\mathbf{6.00 \times 10^{2}}$	9.84×10^{-5}	6.03×10^{2}	1.25
f7	1.13×10^{3}	5.74×10^{1}	1.05×10^{3}	7.94×10^{1}	7.90×10^{2}	1.33×10^{1}	1.10×10^{3}	4.33×10^{1}	$\mathbf{7.85 \times 10^{2}}$	1.37×10^{1}
f8	9.23×10^{2}	1.03×10^{1}	9.19×10^{2}	1.69×10^{1}	8.55×10^{2}	1.17×10^{1}	9.24×10^{2}	1.23×10^{1}	$\mathbf{8.51 \times 10^{2}}$	1.18×10^{1}
f9	3.83×10^{3}	4.52×10^{2}	3.38×10^{3}	5.09×10^{2}	1.09×10^{3}	7.96×10^{1}	3.86×10^{3}	5.88×10^{2}	$\mathbf{1.07 \times 10^{3}}$	7.74×10^{1}
f10	4.52×10^{3}	5.76×10^{2}	4.48×10^{3}	6.19×10^{2}	5.10×10^{3}	8.03×10^{2}	4.42×10^{3}	5.62×10^{2}	$\mathbf{4.13 \times 10^{3}}$	5.56×10^{2}
f11	2.72×10^{3}	9.83×10^{2}	2.30×10^{3}	7.60×10^{2}	1.28×10^{3}	6.47×10^{1}	2.34×10^{3}	8.24×10^{2}	$\mathbf{1.21 \times 10^{3}}$	4.17×10^{1}
f12	4.69×10^{9}	1.15×10^{9}	1.75×10^{9}	1.98×10^{9}	3.93×10^{5}	3.02×10^{5}	4.63×10^{9}	1.51×10^{9}	$\mathbf{3.43 \times 10^{5}}$	2.20×10^{5}
f13	2.49×10^{9}	1.34×10^{9}	2.17×10^{9}	1.84×10^{9}	9.26×10^{3}	2.63×10^{3}	2.44×10^{9}	1.91×10^{9}	$\mathbf{8.16 \times 10^{3}}$	3.29×10^{3}
f14	3.61×10^{4}	7.11×10^{4}	1.85×10^{4}	1.86×10^{4}	2.64×10^{4}	2.00×10^{4}	2.10×10^{4}	2.81×10^{4}	$\mathbf{1.05 \times 10^{4}}$	7.23×10^{3}
f15	1.25×10^{7}	2.26×10^{7}	5.51×10^{6}	1.22×10^{7}	4.98×10^{3}	6.71×10^{3}	1.22×10^{7}	2.84×10^{7}	$\mathbf{2.16 \times 10^{3}}$	4.39×10^{2}
f16	2.86×10^{3}	2.26×10^{2}	2.82×10^{3}	2.40×10^{2}	2.42×10^{3}	2.52×10^{2}	2.95×10^{3}	3.34×10^{2}	$\mathbf{2.39 \times 10^{3}}$	2.11×10^{2}
f17	2.38×10^{3}	2.51×10^{2}	2.27×10^{3}	1.93×10^{2}	1.94×10^{3}	1.07×10^{2}	2.35×10^{3}	2.65×10^{2}	$\mathbf{1.91 \times 10^{3}}$	1.25×10^{2}

Continued on next page

Function	AOA		AOA-S1		AOA-S2		AOA-S3		HCAOA	
Measure	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD	MEAN	STD
f18	1.71×10^5	1.43×10^5	1.72×10^5	1.85×10^5	2.70×10^5	1.69×10^5	1.58×10^5	1.61×10^5	$\mathbf{1.23 \times 10^5}$	9.25×10^4
f19	2.03×10^7	2.76×10^7	1.93×10^7	3.39×10^7	6.18×10^3	3.82×10^3	1.85×10^7	4.08×10^7	$\mathbf{5.50 \times 10^3}$	1.87×10^3
f20	2.31×10^3	7.58×10^1	2.29×10^3	5.15×10^1	2.42×10^3	1.42×10^2	2.35×10^3	1.08×10^2	$\mathbf{2.26 \times 10^3}$	1.07×10^2
f21	2.46×10^3	2.31×10^1	2.46×10^3	1.99×10^1	2.36×10^3	9.15	2.46×10^3	1.16×10^1	$\mathbf{2.35 \times 10^3}$	1.38×10^1
f22	3.51×10^3	8.88×10^2	2.96×10^3	5.21×10^2	4.85×10^3	2.14×10^3	4.30×10^3	1.22×10^3	$\mathbf{2.30 \times 10^3}$	3.16
f23	3.09×10^3	7.13×10^1	3.01×10^3	1.14×10^2	$\mathbf{2.69 \times 10^3}$	1.33×10^1	3.10×10^3	5.45×10^1	2.72×10^3	1.52×10^1
f24	3.41×10^3	1.20×10^2	3.06×10^3	7.65×10^1	2.89×10^3	1.28×10^1	3.56×10^3	1.13×10^2	$\mathbf{2.88 \times 10^3}$	1.58×10^1
f25	3.81×10^3	2.96×10^2	3.32×10^3	2.19×10^2	2.90×10^3	9.98	3.76×10^3	2.18×10^2	$\mathbf{2.89 \times 10^3}$	6.44
f26	8.55×10^3	3.64×10^2	8.11×10^3	8.38×10^2	$\mathbf{4.18 \times 10^3}$	1.32×10^2	8.44×10^3	6.14×10^2	4.69×10^3	2.41×10^2
f27	3.72×10^3	2.18×10^2	3.60×10^3	2.71×10^2	3.25×10^3	6.20	3.70×10^3	1.77×10^2	$\mathbf{3.24 \times 10^3}$	1.34×10^1
f28	5.33×10^3	4.15×10^2	4.30×10^3	5.52×10^2	3.26×10^3	1.12×10^1	5.32×10^3	4.46×10^2	$\mathbf{3.23 \times 10^3}$	2.26×10^1
f29	4.87×10^3	3.56×10^2	4.83×10^3	4.56×10^2	3.74×10^3	1.71×10^2	4.86×10^3	3.93×10^2	$\mathbf{3.67 \times 10^3}$	1.50×10^2
f30	2.35×10^8	1.82×10^8	6.14×10^7	1.06×10^8	8.07×10^3	1.78×10^3	8.90×10^7	1.04×10^8	$\mathbf{7.00 \times 10^3}$	1.11×10^3

4.3. Qualitative analysis results

Five benchmark functions are randomly selected from different categories to present qualitative measures of HCAOA. The qualitative results in Figure 5 include five subplots, namely (1) 2D visualization of benchmark functions, (2) search history, (3) the trajectory in 1st dimension, (4) average fitness of populations and (5) convergence curve.

2D visualization of benchmark functions shows the changes in function values corresponding to the first and second dimensional data. As the dimensions increase, the results become more complex and difficult to plot.

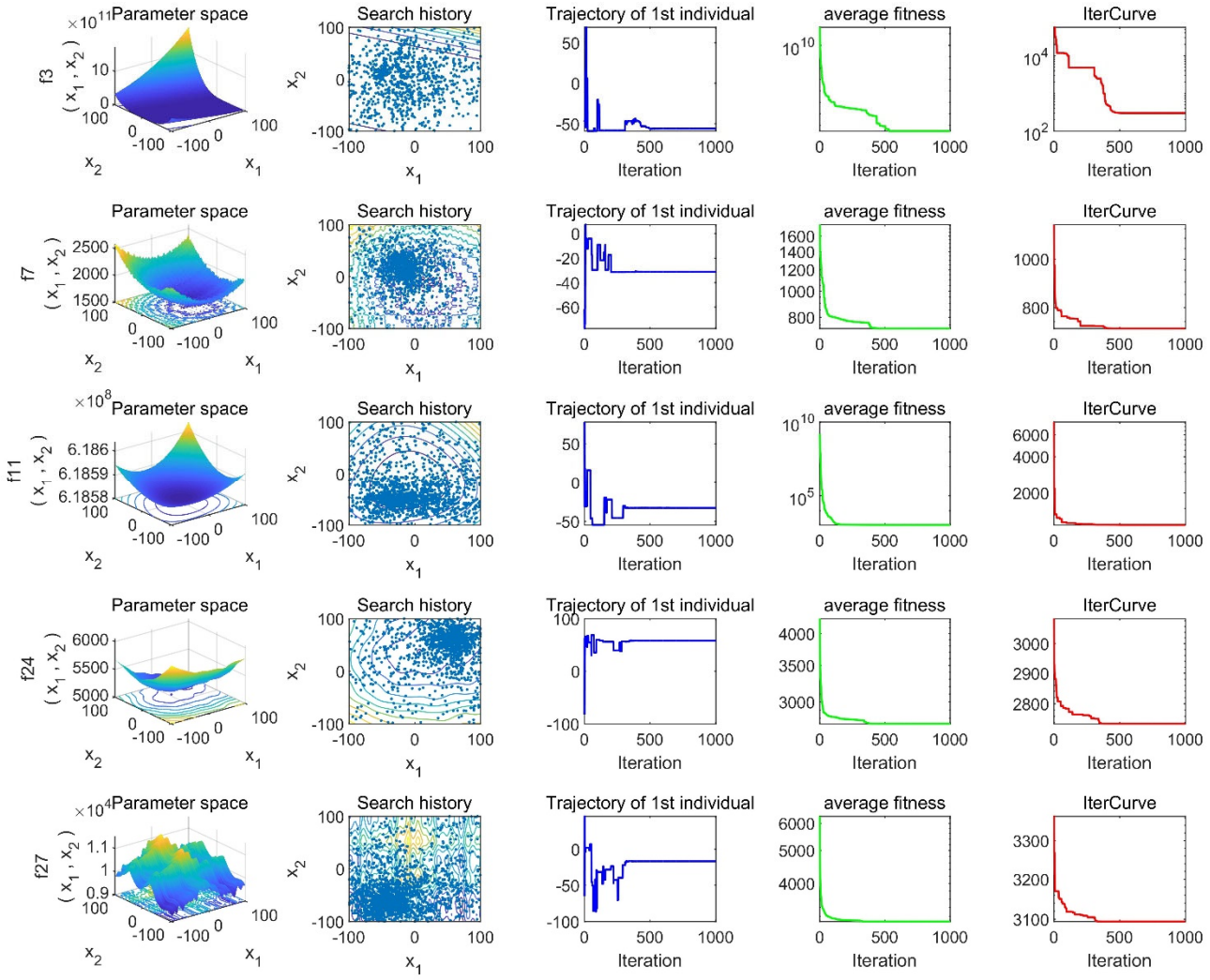

Figure 5. Qualitative results of HCAOA.

The search history charts illustrate the positions of all individuals from the first to the last generation. The search space is described by chromatic contours, starting with blue (lowest value) and ending with yellow (highest value). Two conclusions can be drawn from these search history graphs. First, the locations of search agents cover almost the entire search space, which indicates that HCAOA has good exploration capability. Second, search agents gather more around the blue line, suggesting that HCAOA achieves good exploitation. In summary, these two points demonstrate that HCAOA has an outstanding collaboration between exploration and exploitation.

The trajectories of the first individual in the first dimension reflect the search behavior. The third column of Figure 5 shows that the trajectories vary greatly in the early phase and tend to flatten out in the late stage. This process highlights a gradual shift from exploration to exploitation of HCAOA as the iterations progress. The trajectories of some graphs have some oscillations, which denotes that HCAOA has the ability to escape from a local extremum. In addition, the more complex the function is, the greater the number of trajectory fluctuations and the longer the oscillation time. These graphs show the global search capability of HCAOA.

The fourth column of Figure 5 displays the average fitness of whole population, reflecting the changes in fitness of entire population during iterations. Overall, the average fitness continues to decrease due to the greedy strategy in HCAOA, which guarantees continuous optimization. It is also found that the average fitness values decrease rapidly in the early period and gradually taper off in the later phase, demonstrating a transition of HCAOA from exploration to exploitation.

Convergence curves represent the behavior of the optimal solution up to now. As described in the fifth column of Figure 5, the convergence curves on all functions are gradually decreasing, indicating that HCAOA is continuously optimizing the results as iterations continue. In addition, convergence curves fall rapidly in the early stage and slowly in the late stage, revealing that HCAOA has a high cooperation between exploration and exploitation.

4.4. Statistical analysis results

This subsection measures the performance of HCAOA and the other nine algorithms by statistical analysis. The results of 30 independent optimizations are summarized and four indictors including mean, standard deviation, best value and worst value are selected for statistical analysis. The statistical grades are shown in Table 6, and optimal values are indicated in bold. Overall, HCAOA scores are best in 97 out of 116 comparisons of four indicators on 29 functions. First, HCAOA gets the best mean values on 26 functions. For f3, AOA and HHO perform better, and HCAOA is third. GWO outperforms HCAOA on f16 and f26, with HCAOA in second place. Second, HCAOA has the 24 best values out of 29 functions for the standard deviation indicator. HCAOA ranks second on f8 and f29, third on f3 and f16, and fourth on f10. Furthermore, from the perspective of best value, HCAOA achieves the optimal value on 21 functions. HCAOA is third on f3 and second on f5, f10, f19, f21, f23, f26 and f27. Finally, HCAOA achieves the best on 26 functions for the worst value metric. For the worst value, HCAOA ranks third on f3 and second on f16 and f29. Even for many functions, the worst results acquired by HCAOA are even more satisfactory than the best value by other algorithms. In summary, HCAOA achieves the best results in terms of mean, standard deviation, best and worst on most benchmarks.

Table 6. Statistical results on the CEC 2017 functions with Dim = 30.

Function	Measure	AOA	PSO	GWO	SCA	FA	WOA	HHO	BOA	SHO	HCAOA
f1	MEAN	2.83×10^{10}	9.00×10^{10}	1.00×10^{9}	1.47×10^{10}	5.54×10^{10}	6.52×10^{7}	1.24×10^{7}	3.90×10^{10}	1.37×10^{10}	$\mathbf{2.66 \times 10^{3}}$
	STD	6.03×10^{9}	1.76×10^{10}	5.55×10^{8}	1.91×10^{9}	8.70×10^{9}	2.50×10^{7}	2.58×10^{6}	6.38×10^{9}	4.08×10^{9}	$\mathbf{2.78 \times 10^{3}}$
	BEST	1.38×10^{10}	5.07×10^{10}	1.45×10^{8}	1.19×10^{10}	3.24×10^{10}	1.69×10^{7}	7.42×10^{6}	2.81×10^{10}	6.33×10^{9}	$\mathbf{4.16 \times 10^{2}}$
	WORST	3.90×10^{10}	1.18×10^{11}	2.19×10^{9}	2.08×10^{10}	7.60×10^{10}	1.11×10^{8}	1.67×10^{7}	5.32×10^{10}	2.46×10^{10}	$\mathbf{1.64 \times 10^{4}}$
f3	MEAN	$\mathbf{5.29 \times 10^{3}}$	2.27×10^{5}	3.54×10^{4}	4.70×10^{4}	9.83×10^{4}	2.10×10^{5}	1.32×10^{4}	6.41×10^{4}	5.08×10^{4}	1.51×10^{4}
	STD	$\mathbf{2.80 \times 10^{3}}$	6.33×10^{4}	1.12×10^{4}	1.01×10^{4}	1.53×10^{4}	5.87×10^{4}	3.60×10^{3}	7.92×10^{3}	9.35×10^{3}	6.62×10^{3}
	BEST	$\mathbf{1.12 \times 10^{3}}$	1.24×10^{5}	1.52×10^{4}	2.64×10^{4}	6.70×10^{4}	1.19×10^{5}	6.79×10^{3}	4.55×10^{4}	3.68×10^{4}	7.19×10^{3}
	WORST	$\mathbf{1.38 \times 10^{4}}$	3.84×10^{5}	6.13×10^{4}	6.58×10^{4}	1.26×10^{5}	3.58×10^{5}	1.96×10^{4}	7.44×10^{4}	7.09×10^{4}	3.99×10^{4}
f4	MEAN	7.04×10^{3}	2.85×10^{4}	5.55×10^{2}	1.76×10^{3}	1.39×10^{4}	5.93×10^{2}	5.34×10^{2}	1.68×10^{4}	1.86×10^{3}	$\mathbf{5.03 \times 10^{2}}$
	STD	2.15×10^{3}	8.09×10^{3}	4.40×10^{1}	4.02×10^{2}	2.43×10^{3}	5.70×10^{1}	3.77×10^{1}	2.67×10^{3}	8.41×10^{2}	$\mathbf{1.74 \times 10^{1}}$
	BEST	4.01×10^{3}	8.82×10^{3}	4.88×10^{2}	1.16×10^{3}	7.64×10^{3}	5.25×10^{2}	4.73×10^{2}	1.17×10^{4}	7.71×10^{2}	$\mathbf{4.72 \times 10^{2}}$
	WORST	1.23×10^{4}	4.53×10^{4}	6.55×10^{2}	2.90×10^{3}	1.80×10^{4}	7.60×10^{2}	6.04×10^{2}	2.30×10^{4}	4.44×10^{3}	$\mathbf{5.16 \times 10^{2}}$

Continued on next page

Function	Measure	AOA	PSO	GWO	SCA	FA	WOA	HHO	BOA	SHO	HCAOA
f5	MEAN	6.84×10^{2}	9.90×10^{2}	5.93×10^{2}	7.96×10^{2}	8.99×10^{2}	7.89×10^{2}	7.52×10^{2}	8.80×10^{2}	6.96×10^{2}	$\mathbf{5.62 \times 10^{2}}$
	STD	1.94×10^{1}	6.39×10^{1}	3.34×10^{1}	1.51×10^{1}	3.18×10^{1}	4.91×10^{1}	3.12×10^{1}	2.27×10^{1}	3.29×10^{1}	$\mathbf{1.25 \times 10^{1}}$
	BEST	6.49×10^{2}	8.58×10^{2}	$\mathbf{5.39 \times 10^{2}}$	7.67×10^{2}	8.15×10^{2}	6.82×10^{2}	6.86×10^{2}	8.27×10^{2}	6.43×10^{2}	5.45×10^{2}
	WORST	7.32×10^{2}	1.14×10^{3}	7.28×10^{2}	8.21×10^{2}	9.64×10^{2}	9.14×10^{2}	8.14×10^{2}	9.22×10^{2}	7.62×10^{2}	$\mathbf{6.00 \times 10^{2}}$
f6	MEAN	6.49×10^{2}	6.93×10^{2}	6.05×10^{2}	6.53×10^{2}	6.75×10^{2}	6.72×10^{2}	6.62×10^{2}	6.78×10^{2}	6.37×10^{2}	$\mathbf{6.03 \times 10^{2}}$
	STD	4.16	1.20×10^{1}	2.21	4.54	6.39	1.26×10^{1}	5.93	6.57	8.80	**1.25**
	BEST	6.42×10^{2}	6.65×10^{2}	6.02×10^{2}	6.40×10^{2}	6.63×10^{2}	6.50×10^{2}	6.49×10^{2}	6.60×10^{2}	6.23×10^{2}	$\mathbf{6.01 \times 10^{2}}$
	WORST	6.56×10^{2}	7.13×10^{2}	6.13×10^{2}	6.63×10^{2}	6.97×10^{2}	7.02×10^{2}	6.74×10^{2}	6.90×10^{2}	6.62×10^{2}	$\mathbf{6.06 \times 10^{2}}$
f7	MEAN	1.13×10^{3}	2.42×10^{3}	8.48×10^{2}	1.15×10^{3}	1.76×10^{3}	1.21×10^{3}	1.22×10^{3}	1.33×10^{3}	1.06×10^{3}	$\mathbf{7.85 \times 10^{2}}$
	STD	5.74×10^{1}	2.73×10^{2}	3.69×10^{1}	4.77×10^{1}	1.04×10^{2}	8.18×10^{1}	7.47×10^{1}	3.91×10^{1}	6.82×10^{1}	$\mathbf{1.37 \times 10^{1}}$
	BEST	9.99×10^{2}	1.74×10^{3}	7.94×10^{2}	1.07×10^{3}	1.56×10^{3}	1.09×10^{3}	1.05×10^{3}	1.27×10^{3}	9.41×10^{2}	$\mathbf{7.59 \times 10^{2}}$
	WORST	1.24×10^{3}	2.90×10^{3}	9.35×10^{2}	1.25×10^{3}	1.96×10^{3}	1.49×10^{3}	1.37×10^{3}	1.41×10^{3}	1.26×10^{3}	$\mathbf{8.16 \times 10^{2}}$
f8	MEAN	9.23×10^{2}	1.22×10^{3}	8.82×10^{2}	1.07×10^{3}	1.14×10^{3}	1.01×10^{3}	9.63×10^{2}	1.11×10^{3}	9.48×10^{2}	$\mathbf{8.51 \times 10^{2}}$
	STD	$\mathbf{1.03 \times 10^{1}}$	6.19×10^{1}	3.48×10^{1}	1.83×10^{1}	2.85×10^{1}	5.04×10^{1}	3.13×10^{1}	1.84×10^{1}	3.02×10^{1}	1.18×10^{1}
	BEST	8.99×10^{2}	1.10×10^{3}	8.46×10^{2}	1.02×10^{3}	1.06×10^{3}	9.28×10^{2}	8.84×10^{2}	1.07×10^{3}	9.04×10^{2}	$\mathbf{8.19 \times 10^{2}}$
	WORST	9.40×10^{2}	1.36×10^{3}	1.00×10^{3}	1.10×10^{3}	1.17×10^{3}	1.17×10^{3}	1.02×10^{3}	1.14×10^{3}	1.01×10^{3}	$\mathbf{8.76 \times 10^{2}}$
f9	MEAN	3.83×10^{3}	1.68×10^{4}	1.43×10^{3}	5.96×10^{3}	9.26×10^{3}	7.63×10^{3}	6.75×10^{3}	9.76×10^{3}	4.63×10^{3}	$\mathbf{1.07 \times 10^{3}}$
	STD	4.52×10^{2}	4.88×10^{3}	2.35×10^{2}	1.16×10^{3}	1.53×10^{3}	1.87×10^{3}	7.10×10^{2}	9.35×10^{2}	6.46×10^{2}	$\mathbf{7.74 \times 10^{1}}$
	BEST	2.79×10^{3}	8.81×10^{3}	1.06×10^{3}	4.16×10^{3}	7.15×10^{3}	4.44×10^{3}	5.19×10^{3}	7.66×10^{3}	3.24×10^{3}	$\mathbf{9.51 \times 10^{2}}$
	WORST	5.03×10^{3}	3.42×10^{4}	1.95×10^{3}	9.30×10^{3}	1.34×10^{4}	1.07×10^{4}	8.48×10^{3}	1.14×10^{4}	5.77×10^{3}	$\mathbf{1.24 \times 10^{3}}$
f10	MEAN	4.52×10^{3}	8.08×10^{3}	4.21×10^{3}	8.38×10^{3}	7.82×10^{3}	6.65×10^{3}	5.48×10^{3}	8.62×10^{3}	4.97×10^{3}	$\mathbf{4.13 \times 10^{3}}$
	STD	5.76×10^{2}	6.93×10^{2}	1.39×10^{3}	$\mathbf{2.83 \times 10^{2}}$	5.25×10^{2}	7.85×10^{2}	6.39×10^{2}	3.91×10^{2}	5.89×10^{2}	5.56×10^{2}
	BEST	3.67×10^{3}	6.86×10^{3}	$\mathbf{2.72 \times 10^{3}}$	7.85×10^{3}	6.56×10^{3}	5.00×10^{3}	4.55×10^{3}	7.40×10^{3}	3.75×10^{3}	2.86×10^{3}
	WORST	5.62×10^{3}	9.39×10^{3}	9.22×10^{3}	8.93×10^{3}	8.76×10^{3}	8.06×10^{3}	6.94×10^{3}	9.16×10^{3}	6.08×10^{3}	$\mathbf{5.02 \times 10^{3}}$
f11	MEAN	2.72×10^{3}	2.73×10^{4}	1.48×10^{3}	2.39×10^{3}	7.41×10^{3}	2.88×10^{3}	1.26×10^{3}	4.72×10^{3}	2.28×10^{3}	$\mathbf{1.21 \times 10^{3}}$
	STD	9.83×10^{2}	1.23×10^{4}	2.18×10^{2}	5.86×10^{2}	2.09×10^{3}	1.24×10^{3}	4.83×10^{1}	1.25×10^{3}	8.07×10^{2}	$\mathbf{4.17 \times 10^{1}}$
	BEST	1.22×10^{3}	1.12×10^{4}	1.27×10^{3}	1.85×10^{3}	3.71×10^{3}	1.48×10^{3}	1.18×10^{3}	2.42×10^{3}	1.36×10^{3}	$\mathbf{1.14 \times 10^{3}}$
	WORST	5.14×10^{3}	6.68×10^{4}	2.24×10^{3}	4.38×10^{3}	1.13×10^{4}	7.19×10^{3}	1.40×10^{3}	8.06×10^{3}	4.50×10^{3}	$\mathbf{1.28 \times 10^{3}}$
f12	MEAN	4.69×10^{9}	1.99×10^{10}	5.50×10^{7}	1.43×10^{9}	1.07×10^{10}	9.90×10^{7}	1.05×10^{7}	7.12×10^{9}	5.72×10^{8}	$\mathbf{3.43 \times 10^{5}}$
	STD	1.15×10^{9}	4.85×10^{9}	6.46×10^{7}	4.58×10^{8}	2.24×10^{9}	7.40×10^{7}	7.42×10^{6}	2.83×10^{9}	6.30×10^{8}	$\mathbf{2.20 \times 10^{5}}$
	BEST	1.49×10^{9}	6.28×10^{9}	2.29×10^{6}	6.77×10^{8}	7.09×10^{9}	6.22×10^{6}	2.53×10^{6}	2.20×10^{9}	2.45×10^{7}	$\mathbf{5.43 \times 10^{4}}$
	WORST	6.46×10^{9}	2.96×10^{10}	2.30×10^{8}	2.54×10^{9}	1.49×10^{10}	2.56×10^{8}	2.99×10^{7}	1.53×10^{10}	3.00×10^{9}	$\mathbf{8.52 \times 10^{5}}$
f13	MEAN	2.49×10^{9}	1.58×10^{10}	3.08×10^{6}	5.87×10^{8}	6.54×10^{9}	1.77×10^{5}	4.53×10^{5}	4.76×10^{9}	3.94×10^{7}	$\mathbf{8.16 \times 10^{3}}$
	STD	1.34×10^{9}	7.85×10^{9}	9.00×10^{6}	2.25×10^{8}	2.79×10^{9}	8.51×10^{4}	6.13×10^{5}	1.71×10^{9}	1.15×10^{8}	$\mathbf{3.29 \times 10^{3}}$
	BEST	3.57×10^{7}	2.86×10^{9}	3.74×10^{4}	1.63×10^{8}	7.90×10^{8}	5.42×10^{4}	1.13×10^{5}	4.86×10^{8}	7.45×10^{4}	$\mathbf{2.38 \times 10^{3}}$
	WORST	5.48×10^{9}	4.03×10^{10}	3.48×10^{7}	1.16×10^{9}	1.32×10^{10}	4.05×10^{5}	3.67×10^{6}	7.61×10^{9}	5.52×10^{8}	$\mathbf{1.49 \times 10^{4}}$
f14	MEAN	3.61×10^{4}	8.23×10^{6}	1.39×10^{5}	2.57×10^{5}	1.42×10^{6}	1.67×10^{6}	1.24×10^{5}	6.86×10^{5}	4.61×10^{5}	$\mathbf{1.05 \times 10^{4}}$
	STD	7.11×10^{4}	1.04×10^{7}	2.00×10^{5}	2.63×10^{5}	1.49×10^{6}	1.64×10^{6}	1.61×10^{5}	5.19×10^{5}	4.19×10^{5}	$\mathbf{7.23 \times 10^{3}}$
	BEST	1.98×10^{3}	8.06×10^{4}	2.07×10^{3}	2.06×10^{4}	1.15×10^{4}	3.05×10^{4}	4.37×10^{3}	4.91×10^{4}	5.25×10^{4}	$\mathbf{1.95 \times 10^{3}}$
	WORST	3.24×10^{5}	5.45×10^{7}	9.19×10^{5}	1.30×10^{6}	5.71×10^{6}	5.79×10^{6}	6.81×10^{5}	1.98×10^{6}	2.22×10^{6}	$\mathbf{3.19 \times 10^{4}}$
f15	MEAN	1.25×10^{7}	1.42×10^{9}	7.20×10^{4}	2.73×10^{7}	4.10×10^{8}	9.08×10^{4}	6.60×10^{4}	8.45×10^{7}	2.70×10^{5}	$\mathbf{2.16 \times 10^{3}}$
	STD	2.26×10^{7}	9.99×10^{8}	1.03×10^{5}	3.23×10^{7}	2.88×10^{8}	4.94×10^{4}	3.87×10^{4}	9.67×10^{7}	7.23×10^{5}	$\mathbf{4.39 \times 10^{2}}$
	BEST	3.75×10^{3}	1.58×10^{7}	1.49×10^{4}	1.31×10^{6}	1.65×10^{7}	3.45×10^{4}	1.51×10^{4}	4.75×10^{6}	3.00×10^{3}	$\mathbf{1.62 \times 10^{3}}$

Continued on next page

Function	Measure	AOA	PSO	GWO	SCA	FA	WOA	HHO	BOA	SHO	HCAOA
	WORST	1.00×10^{8}	4.00×10^{9}	5.84×10^{5}	1.45×10^{8}	1.35×10^{9}	2.38×10^{5}	1.85×10^{5}	4.97×10^{8}	3.42×10^{6}	$\mathbf{3.46 \times 10^{3}}$
f16	MEAN	2.86×10^{3}	5.55×10^{3}	$\mathbf{2.33 \times 10^{3}}$	3.75×10^{3}	5.17×10^{3}	3.79×10^{3}	3.33×10^{3}	6.00×10^{3}	2.72×10^{3}	2.39×10^{3}
	STD	2.26×10^{2}	1.01×10^{3}	$\mathbf{1.91 \times 10^{2}}$	2.08×10^{2}	8.44×10^{2}	4.96×10^{2}	4.57×10^{2}	7.31×10^{2}	2.89×10^{2}	2.11×10^{2}
	BEST	2.33×10^{3}	3.03×10^{3}	1.93×10^{3}	3.39×10^{3}	3.67×10^{3}	2.92×10^{3}	2.40×10^{3}	4.70×10^{3}	2.04×10^{3}	$\mathbf{1.87 \times 10^{3}}$
	WORST	3.22×10^{3}	8.61×10^{3}	$\mathbf{2.74 \times 10^{3}}$	4.11×10^{3}	8.22×10^{3}	4.77×10^{3}	4.48×10^{3}	8.02×10^{3}	3.59×10^{3}	2.78×10^{3}
f17	MEAN	2.38×10^{3}	5.16×10^{3}	1.95×10^{3}	2.53×10^{3}	3.25×10^{3}	2.61×10^{3}	2.47×10^{3}	4.40×10^{3}	2.19×10^{3}	$\mathbf{1.91 \times 10^{3}}$
	STD	2.51×10^{2}	3.07×10^{3}	1.53×10^{2}	1.82×10^{2}	5.61×10^{2}	2.56×10^{2}	2.55×10^{2}	1.30×10^{3}	1.82×10^{2}	$\mathbf{1.25 \times 10^{2}}$
	BEST	1.96×10^{3}	2.74×10^{3}	1.77×10^{3}	2.09×10^{3}	2.34×10^{3}	2.17×10^{3}	1.96×10^{3}	2.91×10^{3}	1.93×10^{3}	$\mathbf{1.75 \times 10^{3}}$
	WORST	2.97×10^{3}	1.70×10^{4}	2.34×10^{3}	2.76×10^{3}	4.88×10^{3}	3.18×10^{3}	2.88×10^{3}	7.99×10^{3}	2.66×10^{3}	$\mathbf{2.16 \times 10^{3}}$
f18	MEAN	1.71×10^{5}	1.35×10^{8}	8.32×10^{5}	5.38×10^{6}	2.31×10^{7}	4.74×10^{6}	1.53×10^{6}	1.00×10^{7}	1.11×10^{6}	$\mathbf{1.23 \times 10^{5}}$
	STD	1.43×10^{5}	1.32×10^{8}	9.64×10^{5}	2.70×10^{6}	2.47×10^{7}	4.12×10^{6}	1.63×10^{6}	1.09×10^{7}	1.57×10^{6}	$\mathbf{9.25 \times 10^{4}}$
	BEST	4.00×10^{4}	8.73×10^{5}	3.99×10^{4}	1.18×10^{6}	9.63×10^{5}	8.42×10^{4}	7.05×10^{4}	5.43×10^{5}	5.87×10^{4}	$\mathbf{2.97 \times 10^{4}}$
	WORST	5.48×10^{5}	4.53×10^{8}	3.33×10^{6}	1.30×10^{7}	9.96×10^{7}	1.32×10^{7}	7.90×10^{6}	4.66×10^{7}	8.05×10^{6}	$\mathbf{4.95 \times 10^{5}}$
f19	MEAN	2.03×10^{7}	2.17×10^{9}	3.24×10^{5}	4.35×10^{7}	6.28×10^{8}	5.87×10^{6}	3.90×10^{5}	8.83×10^{7}	7.59×10^{4}	$\mathbf{5.50 \times 10^{3}}$
	STD	2.76×10^{7}	1.60×10^{9}	3.99×10^{5}	2.22×10^{7}	4.88×10^{8}	4.50×10^{6}	3.16×10^{5}	8.05×10^{7}	1.16×10^{5}	$\mathbf{1.87 \times 10^{3}}$
	BEST	3.35×10^{3}	2.38×10^{8}	1.59×10^{4}	1.30×10^{7}	4.09×10^{7}	1.63×10^{6}	3.04×10^{4}	8.60×10^{6}	$\mathbf{2.25 \times 10^{3}}$	2.80×10^{3}
	WORST	1.09×10^{8}	7.84×10^{9}	1.68×10^{6}	1.17×10^{8}	1.82×10^{9}	2.01×10^{7}	1.18×10^{6}	3.62×10^{8}	3.63×10^{5}	$\mathbf{1.12 \times 10^{4}}$
f20	MEAN	2.31×10^{3}	3.14×10^{3}	2.35×10^{3}	2.74×10^{3}	2.71×10^{3}	2.77×10^{3}	2.65×10^{3}	2.90×10^{3}	2.51×10^{3}	$\mathbf{2.26 \times 10^{3}}$
	STD	7.58×10^{1}	2.27×10^{2}	1.26×10^{2}	1.66×10^{2}	1.08×10^{2}	1.66×10^{2}	1.74×10^{2}	1.10×10^{2}	1.28×10^{2}	$\mathbf{1.07 \times 10^{2}}$
	BEST	2.25×10^{3}	2.64×10^{3}	2.17×10^{3}	2.43×10^{3}	2.48×10^{3}	2.41×10^{3}	2.34×10^{3}	2.65×10^{3}	2.28×10^{3}	$\mathbf{2.08 \times 10^{3}}$
	WORST	2.54×10^{3}	3.46×10^{3}	2.66×10^{3}	3.10×10^{3}	2.92×10^{3}	3.08×10^{3}	3.18×10^{3}	3.10×10^{3}	2.76×10^{3}	$\mathbf{2.48 \times 10^{3}}$
f21	MEAN	2.46×10^{3}	2.77×10^{3}	2.39×10^{3}	2.57×10^{3}	2.69×10^{3}	2.56×10^{3}	2.55×10^{3}	2.52×10^{3}	2.48×10^{3}	$\mathbf{2.35 \times 10^{3}}$
	STD	2.31×10^{1}	6.50×10^{1}	2.32×10^{1}	1.35×10^{1}	3.27×10^{1}	5.74×10^{1}	4.64×10^{1}	1.42×10^{2}	2.68×10^{1}	$\mathbf{1.38 \times 10^{1}}$
	BEST	2.42×10^{3}	2.60×10^{3}	2.36×10^{3}	2.54×10^{3}	2.63×10^{3}	2.43×10^{3}	2.46×10^{3}	$\mathbf{2.30 \times 10^{3}}$	2.43×10^{3}	2.33×10^{3}
	WORST	2.51×10^{3}	2.88×10^{3}	2.49×10^{3}	2.59×10^{3}	2.76×10^{3}	2.68×10^{3}	2.64×10^{3}	2.74×10^{3}	2.53×10^{3}	$\mathbf{2.38 \times 10^{3}}$
f22	MEAN	3.51×10^{3}	9.66×10^{3}	4.61×10^{3}	8.47×10^{3}	8.62×10^{3}	6.64×10^{3}	5.78×10^{3}	3.64×10^{3}	5.28×10^{3}	$\mathbf{2.30 \times 10^{3}}$
	STD	8.88×10^{2}	9.30×10^{2}	1.59×10^{3}	2.40×10^{3}	6.11×10^{2}	2.22×10^{3}	2.09×10^{3}	3.41×10^{2}	1.32×10^{3}	**3.16**
	BEST	2.70×10^{3}	7.30×10^{3}	2.39×10^{3}	3.50×10^{3}	7.49×10^{3}	2.34×10^{3}	2.32×10^{3}	2.95×10^{3}	3.43×10^{3}	$\mathbf{2.30 \times 10^{3}}$
	WORST	5.87×10^{3}	1.15×10^{4}	7.28×10^{3}	1.04×10^{4}	9.58×10^{3}	9.70×10^{3}	8.27×10^{3}	4.37×10^{3}	7.89×10^{3}	$\mathbf{2.31 \times 10^{3}}$
f23	MEAN	3.09×10^{3}	3.60×10^{3}	2.75×10^{3}	3.02×10^{3}	3.66×10^{3}	3.07×10^{3}	3.09×10^{3}	3.31×10^{3}	2.92×10^{3}	$\mathbf{2.72 \times 10^{3}}$
	STD	7.13×10^{1}	1.32×10^{2}	4.72×10^{1}	3.35×10^{1}	1.67×10^{2}	1.03×10^{2}	8.33×10^{1}	8.92×10^{1}	3.94×10^{1}	$\mathbf{1.52 \times 10^{1}}$
	BEST	2.89×10^{3}	3.36×10^{3}	$\mathbf{2.68 \times 10^{3}}$	2.96×10^{3}	3.33×10^{3}	2.87×10^{3}	2.94×10^{3}	3.17×10^{3}	2.80×10^{3}	2.69×10^{3}
	WORST	3.22×10^{3}	3.79×10^{3}	2.88×10^{3}	3.08×10^{3}	3.99×10^{3}	3.25×10^{3}	3.29×10^{3}	3.57×10^{3}	3.01×10^{3}	$\mathbf{2.75 \times 10^{3}}$
f24	MEAN	3.41×10^{3}	3.90×10^{3}	2.90×10^{3}	3.19×10^{3}	4.07×10^{3}	3.19×10^{3}	3.35×10^{3}	3.76×10^{3}	3.21×10^{3}	$\mathbf{2.88 \times 10^{3}}$
	STD	1.20×10^{2}	1.99×10^{2}	3.81×10^{1}	2.60×10^{1}	1.99×10^{2}	1.08×10^{2}	1.39×10^{2}	2.23×10^{2}	5.58×10^{1}	$\mathbf{1.58 \times 10^{1}}$
	BEST	3.13×10^{3}	3.49×10^{3}	2.87×10^{3}	3.15×10^{3}	3.47×10^{3}	2.94×10^{3}	3.13×10^{3}	3.30×10^{3}	3.11×10^{3}	$\mathbf{2.85 \times 10^{3}}$
	WORST	3.64×10^{3}	4.30×10^{3}	3.04×10^{3}	3.25×10^{3}	4.38×10^{3}	3.37×10^{3}	3.74×10^{3}	4.09×10^{3}	3.33×10^{3}	$\mathbf{2.92 \times 10^{3}}$
f25	MEAN	3.81×10^{3}	1.14×10^{4}	2.96×10^{3}	3.31×10^{3}	5.85×10^{3}	3.02×10^{3}	2.92×10^{3}	5.15×10^{3}	3.23×10^{3}	$\mathbf{2.89 \times 10^{3}}$
	STD	2.96×10^{2}	2.19×10^{3}	2.70×10^{1}	1.01×10^{2}	6.37×10^{2}	4.51×10^{1}	1.91×10^{1}	4.16×10^{2}	1.45×10^{2}	**6.44**
	BEST	3.28×10^{3}	6.73×10^{3}	2.93×10^{3}	3.07×10^{3}	4.44×10^{3}	2.93×10^{3}	2.89×10^{3}	4.20×10^{3}	3.05×10^{3}	$\mathbf{2.88 \times 10^{3}}$
	WORST	4.56×10^{3}	1.56×10^{4}	3.05×10^{3}	3.51×10^{3}	6.92×10^{3}	3.14×10^{3}	2.96×10^{3}	5.79×10^{3}	3.56×10^{3}	$\mathbf{2.91 \times 10^{3}}$
f26	MEAN	8.55×10^{3}	1.26×10^{4}	$\mathbf{4.50 \times 10^{3}}$	7.17×10^{3}	1.13×10^{4}	7.78×10^{3}	7.30×10^{3}	1.07×10^{4}	6.52×10^{3}	4.69×10^{3}

Continued on next page

Function	Measure	AOA	PSO	GWO	SCA	FA	WOA	HHO	BOA	SHO	HCAOA
	STD	3.64×10^{2}	1.73×10^{3}	3.62×10^{2}	3.41×10^{2}	8.49×10^{2}	1.02×10^{3}	7.85×10^{2}	7.07×10^{2}	8.23×10^{2}	$\mathbf{2.41 \times 10^{2}}$
	BEST	7.79×10^{3}	9.45×10^{3}	$\mathbf{3.43 \times 10^{3}}$	6.48×10^{3}	9.46×10^{3}	5.91×10^{3}	5.99×10^{3}	8.81×10^{3}	4.62×10^{3}	4.07×10^{3}
	WORST	9.51×10^{3}	1.75×10^{4}	5.38×10^{3}	7.72×10^{3}	1.31×10^{4}	9.97×10^{3}	9.52×10^{3}	1.19×10^{4}	7.72×10^{3}	$\mathbf{5.15 \times 10^{3}}$
f27	MEAN	3.72×10^{3}	4.63×10^{3}	$\mathbf{3.24 \times 10^{3}}$	3.45×10^{3}	4.66×10^{3}	3.43×10^{3}	3.40×10^{3}	3.91×10^{3}	3.40×10^{3}	$\mathbf{3.24 \times 10^{3}}$
	STD	2.18×10^{2}	3.34×10^{2}	1.60×10^{1}	4.65×10^{1}	2.92×10^{2}	1.25×10^{2}	1.01×10^{2}	1.88×10^{2}	5.52×10^{1}	$\mathbf{1.34 \times 10^{1}}$
	BEST	3.33×10^{3}	3.93×10^{3}	$\mathbf{3.20 \times 10^{3}}$	3.39×10^{3}	4.02×10^{3}	3.26×10^{3}	3.28×10^{3}	3.51×10^{3}	3.28×10^{3}	3.21×10^{3}
	WORST	4.08×10^{3}	5.29×10^{3}	3.29×10^{3}	3.58×10^{3}	5.24×10^{3}	3.70×10^{3}	3.78×10^{3}	4.21×10^{3}	3.56×10^{3}	$\mathbf{3.27 \times 10^{3}}$
f28	MEAN	5.33×10^{3}	9.83×10^{3}	3.38×10^{3}	3.97×10^{3}	6.96×10^{3}	3.37×10^{3}	3.28×10^{3}	7.27×10^{3}	3.89×10^{3}	$\mathbf{3.23 \times 10^{3}}$
	STD	4.15×10^{2}	1.77×10^{3}	5.56×10^{1}	1.77×10^{2}	6.35×10^{2}	5.13×10^{1}	2.78×10^{1}	3.82×10^{2}	3.22×10^{2}	$\mathbf{2.26 \times 10^{1}}$
	BEST	4.64×10^{3}	6.17×10^{3}	3.27×10^{3}	3.60×10^{3}	5.90×10^{3}	3.26×10^{3}	3.22×10^{3}	6.58×10^{3}	3.43×10^{3}	$\mathbf{3.20 \times 10^{3}}$
	WORST	6.49×10^{3}	1.29×10^{4}	3.51×10^{3}	4.57×10^{3}	8.41×10^{3}	3.49×10^{3}	3.34×10^{3}	7.90×10^{3}	4.59×10^{3}	$\mathbf{3.27 \times 10^{3}}$
f29	MEAN	4.87×10^{3}	8.63×10^{3}	3.69×10^{3}	4.85×10^{3}	6.62×10^{3}	5.07×10^{3}	4.49×10^{3}	8.10×10^{3}	4.10×10^{3}	$\mathbf{3.67 \times 10^{3}}$
	STD	3.56×10^{2}	2.45×10^{3}	$\mathbf{1.21 \times 10^{2}}$	2.33×10^{2}	9.63×10^{2}	4.56×10^{2}	3.80×10^{2}	1.59×10^{3}	2.07×10^{2}	1.50×10^{2}
	BEST	4.20×10^{3}	5.82×10^{3}	3.46×10^{3}	4.39×10^{3}	5.10×10^{3}	4.27×10^{3}	3.66×10^{3}	5.67×10^{3}	3.69×10^{3}	$\mathbf{3.43 \times 10^{3}}$
	WORST	5.68×10^{3}	1.54×10^{4}	$\mathbf{3.94 \times 10^{3}}$	5.54×10^{3}	9.42×10^{3}	6.23×10^{3}	5.26×10^{3}	1.23×10^{4}	4.53×10^{3}	4.11×10^{3}
f30	MEAN	2.35×10^{8}	1.46×10^{9}	6.33×10^{6}	9.81×10^{7}	6.98×10^{8}	1.97×10^{7}	1.90×10^{6}	4.08×10^{8}	2.70×10^{6}	$\mathbf{7.00 \times 10^{3}}$
	STD	1.82×10^{8}	9.75×10^{8}	4.38×10^{6}	3.23×10^{7}	3.22×10^{8}	1.96×10^{7}	1.03×10^{6}	2.90×10^{8}	3.02×10^{6}	$\mathbf{1.11 \times 10^{3}}$
	BEST	1.71×10^{5}	9.44×10^{7}	6.69×10^{5}	4.35×10^{7}	2.22×10^{8}	2.16×10^{6}	2.86×10^{5}	3.83×10^{7}	3.33×10^{5}	$\mathbf{5.46 \times 10^{3}}$
	WORST	7.24×10^{8}	4.16×10^{9}	2.00×10^{7}	1.79×10^{8}	1.67×10^{9}	8.18×10^{7}	4.74×10^{6}	1.34×10^{9}	1.20×10^{7}	$\mathbf{1.10 \times 10^{4}}$

4.5. Stability analysis results

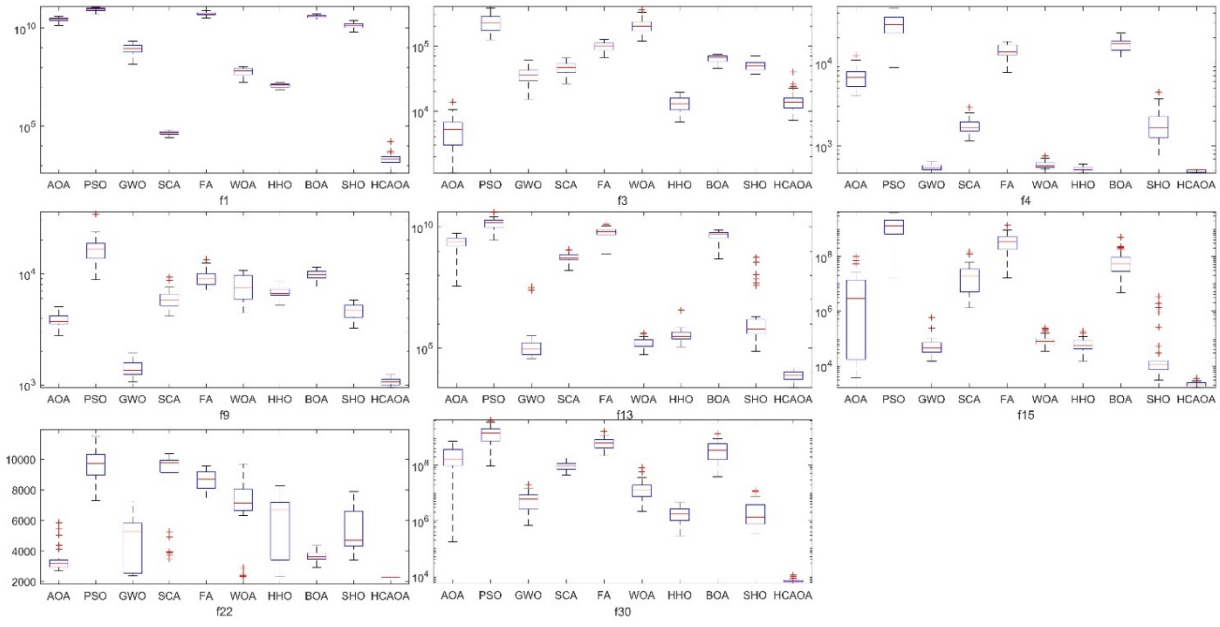

Figure 6. The box plots of different algorithms on 8 functions.

To observe the stability of HCAOA, all algorithms are analyzed through box diagrams on 30 optimization results. The box plots of various algorithms on 8 functions from diverse classifications are exhibited in Figure 6. Medians, lower quartiles, upper quartiles and outliers are presented in box

plots. The narrow block indicates that the results obtained by an algorithm are relatively steady. Three conclusions can be drawn from Figure 6. First, HCAOA can get lower boxes than other algorithms in most cases, which means that HCAOA has good optimization effects. Second, the boxplots obtained by HCAOA are always narrowest, which means that HCAOA receives stable optimization results. Finally, there are relatively few outliers in the box charts of HCAOA. In combination, HCAOA produces superior and stable optimization results.

4.6. Convergence analysis results

To validate the convergence efficiency of HCAOA, the same functions are selected from subsection 4.5, and their convergence curves are presented in Figure 7. Their global exploration and local exploitation are more intuitively represented in convergence curves. Figure 7 reveals that HCAOA achieves better results and faster convergence than the other nine algorithms on most functions. In the early stages, GWO converges faster than HCAOA on f1, f4 and f9, WOA has faster convergence on f13 and f15, and the early convergence of HHO on f30 is faster than that of HCAOA. However, as the iterations progress, GWO, HHO and WOA converge slower and enter the local exploitation phase earlier, which may fall into local optima. In contrast, HCAOA has a stronger exploration capability and applies more iterations to search for the global best solution. Therefore, from the final optimization results, it can be concluded that HCAOA maintains fast convergence for a longer time and has better optimization effects. In addition, Figure 7 shows that the curve shapes of HCAOA are very similar to those of AOA. They transfer from global exploration to local exploitation at the same iteration, mainly because they employ the same transfer operator. However, a hierarchical chain, with different processing mechanisms for agents at different levels, allows HCAOA to have faster convergence and better local search capabilities.

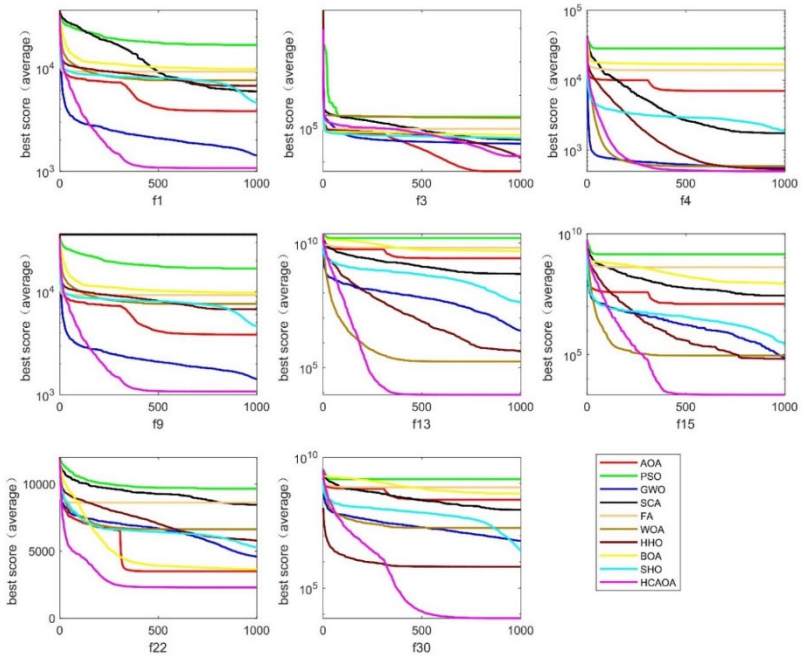

Figure 7. The convergence curves of different algorithms on 8 functions.

4.7. Statistical test results

In the previous subsections, it has been verified that HCAOA has a superior optimization ability compared to other algorithms from the statistical results, convergence analysis results and stability analysis results. To avoid chance, this subsection analyses whether the optimization capability of HCAOA is significantly superior to other algorithms. In the statistical results in subsection 4.3, the mean values of 30 optimization results of different algorithms on 29 functions are compared. It is found that AOA and HHO outperform HCAOA on f3, GWO is superior to HCAOA on f16 and f26, and HCAOA wins over other algorithms on the remaining functions. Statistical tests are conducted based on 30 optimization results, using Wilcoxon rank-sum statistical method to test the null hypothesis of no remarkable distinction between the optimized effects of two algorithms. If $p-value > 0.05$, it implies that the distinction between the optimization effects is not noticeable. In contrast, if $p-value \le 0.05$, it denotes the divergence is statistically significant, namely $H = 1$.

Table 7 lists Wilcoxon rank-sum test results for HCAOA against other algorithms on CEC 2017. It can be judged that HCAOA wins 251 times, is equal 8 times and loses 2 times in the 261 (29 × 9) comparisons. Among them, the optimization effects of AOA are superior to that of HCAOA on f3, and the optimization effects of GWO on f26 have a marked predominance over that of HCAOA. However, the hypotheses that the results obtained by HHO are significantly better than that obtained by HCAOA on f3 and GWO beats HCAOA on f16 do not hold. In addition, the optimization outcomes of AOA and HCAOA have no apparent distinctions on f14 and f18. On f10, f16, f17, f27 and f29, the differences between the optimization effects obtained by GWO and those obtained by HCAOA are not significant. There is no significant difference between the optimization results of HHO and HCAOA on f3. Finally, the effects of HCAOA are apparently superior to others on the remaining functions. Therefore, from the perspective of statistical tests, the optimization results of HCAOA have a marked predominance over that of other algorithms on most cases.

Table 7. Wilcoxon rank-sum test results for HCAOA against other algorithms on CEC 2017.

	AOA vs HCAOA		PSO vs HCAOA		GWO vs HCAOA		SCA vs HCAOA		FA vs HCAOA		WOA vs HCAOA		HHO vs HCAOA		BOA vs HCAOA		SHO vs HCAOA	
	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H	P	H
f1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f3	8.89×10^{-10}	1	3.02×10^{-11}	1	4.57×10^{-9}	1	6.70×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.95×10^{-1}	0	3.02×10^{-11}	1	4.98×10^{-11}	1
f4	3.02×10^{-11}	1	3.02×10^{-11}	1	1.03×10^{-6}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	8.29×10^{-6}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f5	3.02×10^{-11}	1	3.02×10^{-11}	1	3.57×10^{-6}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f6	3.02×10^{-11}	1	3.02×10^{-11}	1	1.17×10^{-2}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f7	3.02×10^{-11}	1	3.02×10^{-11}	1	1.33×10^{-10}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f8	3.02×10^{-11}	1	3.02×10^{-11}	1	1.60×10^{-7}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1

Continued on next page

	AOA vs HCAOA		PSO vs HCAOA		GWO vs HCAOA		SCA vs HCAOA		FA vs HCAOA		WOA vs HCAOA		HHO vs HCAOA		BOA vs HCAOA		SHO vs HCAOA	
f9	3.02×10^{-11}	1	3.02×10^{-11}	1	1.86×10^{-9}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f10	3.78×10^{-2}	1	3.02×10^{-11}	1	4.12×10^{-1}	0	3.02×10^{-11}	1	3.02×10^{-11}	1	3.34×10^{-11}	1	3.20×10^{-9}	1	3.02×10^{-11}	1	6.74×10^{-6}	1
f11	1.09×10^{-10}	1	3.02×10^{-11}	1	6.70×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.37×10^{-4}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f12	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f13	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f14	5.79×10^{-1}	0	3.02×10^{-11}	1	9.83×10^{-8}	1	4.08×10^{-11}	1	7.39×10^{-11}	1	3.34×10^{-11}	1	2.83×10^{-8}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f15	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.69×10^{-11}	1
f16	8.48×10^{-9}	1	3.02×10^{-11}	1	2.97×10^{-1}	0	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	9.76×10^{-10}	1	3.02×10^{-11}	1	5.46×10^{-6}	1
f17	1.07×10^{-9}	1	3.02×10^{-11}	1	2.23×10^{-1}	0	5.49×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.16×10^{-10}	1	3.02×10^{-11}	1	1.47×10^{-7}	1
f18	3.11×10^{-1}	0	3.02×10^{-11}	1	1.07×10^{-7}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	2.87×10^{-10}	1	6.12×10^{-10}	1	3.02×10^{-11}	1	1.85×10^{-8}	1
f19	4.20×10^{-10}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	1.02×10^{-5}	1
f20	1.76×10^{-2}	1	3.02×10^{-11}	1	1.08×10^{-2}	1	4.98×10^{-11}	1	3.34×10^{-11}	1	4.08×10^{-11}	1	2.15×10^{-10}	1	3.02×10^{-11}	1	1.31×10^{-8}	1
f21	3.02×10^{-11}	1	3.02×10^{-11}	1	1.07×10^{-9}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	6.53×10^{-8}	1	3.02×10^{-11}	1
f22	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f23	3.02×10^{-11}	1	3.02×10^{-11}	1	4.71×10^{-4}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f24	3.02×10^{-11}	1	3.02×10^{-11}	1	6.55×10^{-4}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f25	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	1.29×10^{-9}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f26	3.02×10^{-11}	1	3.02×10^{-11}	1	1.99×10^{-2}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	6.12×10^{-10}	1
f27	3.02×10^{-11}	1	3.02×10^{-11}	1	8.65×10^{-1}	0	3.02×10^{-11}	1	3.02×10^{-11}	1	3.69×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f28	3.02×10^{-11}	1	3.02×10^{-11}	1	3.34×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.69×10^{-11}	1	4.31×10^{-8}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
f29	3.02×10^{-11}	1	3.02×10^{-11}	1	2.97×10^{-1}	0	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	2.87×10^{-10}	1	3.02×10^{-11}	1	2.67×10^{-9}	1

Continued on next page

	AOA vs HCAOA		PSO vs HCAOA		GWO vs HCAOA		SCA vs HCAOA		FA vs HCAOA		WOA vs HCAOA		HHO vs HCAOA		BOA vs HCAOA		SHO vs HCAOA	
f30	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1	3.02×10^{-11}	1
Win	26/29	26	29/29	29	23/29	23	29/29	29	29/29	29	29/29	29	28/29	28	29/29	29	29/29	29
Lose	1/29	1	0/29	0	1/29	1	0/29	0	0/29	0	0/29	0	0/29	0	0/29	0	0/29	0
Equal	2/29	2	0/29	0	5/29	5	0/29	0	0/29	0	0/29	0	1/29	1	0/29	0	0/29	0

Finally, Friedman rank test is applied to sort ten algorithms on 29 functions of CEC 2017. The test is a nonparametric approach to compare the comprehensive average performance. From the Friedman test grades in Table 8, HCAOA ranks first, followed by GWO, HHO, SHO, AOA, WOA, SCA, BOA, FA and PSO. The results of Friedman rank test show that HCAOA is effective and stable.

Table 8. Friedman rank test results for ten algorithms on CEC 2017.

Algorithm	Friedman rank test Rank	Rank
AOA	4.844828	5
PSO	9.793103	10
GWO	2.672414	2
SCA	5.948276	7
FA	8.62069	9
WOA	5.706897	6
HHO	4.103448	3
BOA	8.034483	8
SHO	4.12069	4
HCAOA	1.155172	1

5. The application of HCAOA to engineering design problems

This section evaluates the ability of HCAOA to solve practical problems through four classical engineering problems. The optimization consequences are compared with those of methods proposed in recent literature. The possible optimal results are deepened in bold.

5.1. Welded beam design

The welded beam design problem aims to minimize the cost by specifying four parameters, including weld thickness (h), length connected to bar (l), bar height (t) and bar thickness (b). The optimization results obtained from HCAOA are compared with those of IGWO [44], AGWO [45], DAQUILA [46], DMO [21], MAOA [47], GJO [48], AVOA [49], MCWOA [50], CQFFA [51], HAO [52] and IHAOAVOA [53]. Based on the results in Table 9, it is clear that HCAOA has the lowest cost.

Table 9. Optimization results for welded beam design problem.

Algorithm	h	l	t	b	Optimal cost
IGWO [44]	0.20573	3.47049	9.036624	0.20573	1.724853
AGWO [45]	0.20555	3.4744	9.0378	0.20572	1.7253
DAQUILA [46]	0.20572964	3.47048867	9.03662391	0.20572964	1.72452751
DMO [21]	0.2055705	3.2567724	9.036177	0.2057696	1.6953
MAOA [47]	0.2057	3.4705	9.0366	0.2057	1.7246
GJO [48]	0.20562	3.4719	9.0392	0.20572	1.72522
AVOA [49]	0.205730	3.470474	9.036621	0.205730	1.724852
MCWOA [50]	0.2024	3.3292	9.0486	0.2057	1.7023
CQFFA [51]	0.20573	3.47041	9.03661	0.20573	1.72485
HAO [52]	0.19952608	3.384869727	9.064048595	0.206681757	1.715727482
IHAOAVOA [53]	0.20573	3.4705	9.0366	0.20573	1.7249
HCAOA	**0.20573**	**3.25312**	**9.036624**	**0.20573**	**1.69524**

5.2. Pressure vessel design

The pressure vessel design problem is to determine four parameters including head thickness (T_h), shell thickness (T_s), inner radius (R) and cylindrical cross-section length without considering the head (L), so as to reduce the total fee. IGWO [44], AGWO [45], MAOA [47], GJO [48], AVOA [49], SCSO [54], QLGCTSA [55], CQFFA [51] and HAO [52] are applied to solve this problem. The optimal solutions obtained by them are contrasted with those optimized by HCAOA. Table 10 indicates that HCAOA and CQFFA achieve competitive results and outperform the other algorithms.

Table 10. Optimization results for pressure vessel design problem.

Algorithm	T_h	T_S	R	L	Optimal cost
IGWO [44]	0.779031	0.385501	40.36313	199.4017	5888.34
AGWO [45]	0.778496	0.386451	40.32684	199.9135	5892.243
MAOA [47]	0.7953	0.3931	41.2274	187.7371	5914.48511
GJO [48]	0.7782955	0.3848046	40.32187	200	5887.07112
AVOA [49]	0.778954	0.3850374	40.360312	199.434299	5886.67659
SCSO [54]	0.7798	0.939	40.3864	199.2918	5917.46
QLGCTSA [55]	13.39411	7.075665	42.09845	176.636596	6059.7143
CQFFA [51]	**0.778168**	**0.384649**	**40.319618**	**199.9900**	**5885.3**
HAO [52]	0.810726461	0.400897167	42.16466765	175.8460143	5935.56831
HCAOA	**0.778169**	**0.384649**	**40.319619**	**200**	**5885.33277**

5.3. Cantilever beam design

The cantilever beam design problem consists of five squares with heights to be determined, and vertical forces act on their free nodes. So the decision variables are the heights of five hollow squares, and the objective is to lessen a cantilever's weight. TQA [56], ECSOA [57], GBO [23], AGWO [45], MFO [58], SHO [18], MGA [24], QLGCTSA [55] and IHAOAVOA [53] are also applied to address the cantilever beam design problem and the best solutions optimized by them are shown in Table 11.

It is evident that the HCAOA proposed in this paper provides high-quality optimization results and has the reliable ability to solve practical problems.

Table 11. Optimization results for cantilever beam design problem.

Algorithm	x_1	x_2	x_3	x_4	x_5	Optimal weight
TQA[56]	6.013593	5.307230	4.498556	3.502111	2.152181	1.339957
ECSOA[57]	5.993352	5.332166	4.470567	3.542109	2.137287	1.339958
GBO[23]	6.0124	5.3129	4.4941	3.5036	2.1506	1.339957
AGWO[45]	6.015647	5.310255	4.500563	3.494626	2.15301	1.339984
MFO[58]	5.984871	5.316726	4.497332	3.513616	2.16162	1.339988
SHO[18]	6.0049	5.3227	4.4737	3.5065	2.16637	1.339987
MGA [24]	6.011660	5.315676	4.510682	3.485698	2.150251	1.339975661
QLGCTSA[55]	6.01604	5.30915	4.4943	3.5015	2.1527	1.3401
IHAOAVOA[53]	**6.0108**	**5.3170**	**4.4678**	**3.5324**	**2.1466**	**1.3400**
HCAOA	**6.016837**	**5.307664**	**4.494078**	**3.501990**	**2.153092**	**1.339956**

5.4. Three-bar truss design

This problem is resolved by designing two cross-sectional areas (A_1, A_2) to ensure the stress constraints on each of the truss members, with the aim of reducing the overall weight. The optimization results obtained by HCAOA on this problem are compared with those obtained by algorithms such as GJO [48], PRO [29], IDARSOA [59], CSOAOA [60], AVOA [49], SNS [31], MCWOA [50], SRS [26], CQFFA [51] and QLGCTSA [55]. Table 12 indicates that HCAOA obtains the competitive optimization results.

Table 12. Optimization results for three-bar truss design problem.

Algorithm	A_1	A_2	Optimal weight
GJO [48]	0.788657163	0.408299125	263.8958439
PRO [29]	0.7886475	0.4083262	263.8958439
IDARSOA [59]	0.788906	0.4076	263.8960
CSOAOA [60]	**0.78867513**	**0.40824831**	**263.8958434**
AVOA [49]	0.788680395	0.408233412	263.895843396802
SNS [31]	**0.78868473**	**0.4082211**	**263.8958434**
MCWOA [50]	0.7937	0.3943	263.914
SRS [26]	0.78863	0.40837	263.958434
CQFFA [51]	**0.7886684**	**0.4082672**	**263.8958434**
QLGCTSA [55]	0.78866378	0.40828041	263.8958435
HCAOA	**0.788673916**	**0.408251736**	**263.895843377559**

6. Conclusions

This paper discusses in detail the shortcomings of the canonical AOA. To compensate for these deficiencies, this paper introduces a new idea of hierarchical chains to AOA and proposes an improved HCAOA. This algorithm combines an orthogonal learning mechanism based on refraction opposition

and an Archimedes spiral mechanism based on Levy flight to deal with individuals at different levels, effectively avoiding local optimization, preventing clueless random mining and improving optimization speed. Apart from that, a multi-strategy boundary processing mechanism is introduced to maintain the variety of populations.

To validate the effectiveness of HCAOA, multiple experiments based on the CEC 2017 test suite are conducted. The qualitative results demonstrate that HCAOA has outstanding exploration and exploitation abilities and strikes a good collaboration between exploitation and exploration. The quantitative outcomes prove that HCAOA can obtain superior and more stable optimization results than the other nine recent algorithms. In addition, four real-world engineering problems are used to test the ability of HCAOA to solve practical problems. HCAOA achieves competitive optimization results when comparing with the optimal results in the recent literature for the same problems.

In summary, the proposed HCAOA provides outstanding and stable optimization effects, fast convergence and a good ability to jump out from a local extremum. In the future, HCAOA will be extended to complex single-objective optimization problems, for example, imaging segmentation, feature selection, workshop scheduling and parameter optimization. Meanwhile, HCAOA is being developed to solve unconstrained and constrained multi-objective problems. Furthermore, the idea of hierarchical chains can be generalized to other algorithms.

Use of AI tools declaration

The authors declare we have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

The support of the National Natural Science Foundation of China (Grant No. 72131005, 71771066 and 71901014) is greatly appreciated.

Conflict of interest

The authors declare there is no conflict of interest.

References

1. X. Li, J. Gu, Z. Huang, W. Wang, J. Li, Optimal design of model predictive controller based on transient search optimization applied to robotic manipulators, *Math. Biosci. Eng.*, **19** (2022), 9371–9387. https://doi.org/10.3934/mbe.2022436
2. J. Wang, C. Zhan, S. Li, Q. Zhao, J. Liu, Z. Xie, Adaptive variational mode decomposition based on Archimedes optimization algorithm and its application to bearing fault diagnosis, *Measurement*, **191** (2022), 110798. https://doi.org/10.1016/j.measurement.2022.110798
3. K. Balakrishnan, R. Dhanalakshmi, U. M. Khaire, Excogitating marine predators algorithm based on random opposition-based learning for feature selection, *Concurr. Comput. Pract. Exp.*, **34** (2021), e6630. https://doi.org/10.1002/cpe.6630

4. A. Y. Mahdi, S. S. Yuhaniz, Optimal feature selection using novel flamingo search algorithm for classification of COVID-19 patients from clinical text, *Math. Biosci. Eng.*, **20** (2022), 5268–5297. https://doi.org/10.3934/mbe.2023244
5. S. Das, A. Bhattacharya, A. K. Chakraborty, Quasi-reflected ions motion optimization algorithm for short-term hydrothermal scheduling, *Neural Comput. Appl.*, **29** (2018), 123–149. https://doi.org/10.1007/s00521-016-2529-8
6. H. D. Quoc, MEMINV: A hybrid efficient approximation method solving the multi skill-resource constrained project scheduling problem, *Math. Biosci. Eng.*, **20** (2023), 15407–15430. https://doi.org/10.3934/mbe.2023688
7. C. Wang, S. Jiao, Y. Li, Q. Zhang, Capacity optimization of a hybrid energy storage system considering wind-Solar reliability evaluation based on a novel multi-strategy snake optimization algorithm, *Expert Syst. Appl.*, **231** (2023), 120602. https://doi.org/10.1016/j.eswa.2023.120602
8. S. Jiao, C. Wang, R. Gao, Y. Li, Q. Zhang, A novel hybrid harris hawk sine cosine optimization algorithm for reactive power optimization problem, *J. Exp. Theor. Artif. Intell.*, (2022), https://doi.org/10.1080/0952813X.2022.2115144
9. S. Mirjalili, A. Lewis, The whale optimization algorithm, *Adv. Eng. Softw.*, **95** (2016), 51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008
10. Y. Xiao, X. Sun, Y. Guo, H. Cui, Y. Wang, J. Li, et al., An enhanced honey badger algorithm based on Levy flight and refraction opposition-based learning for engineering design problems, *J. Intell. Fuzzy Syst.*, **43** (2022), 4517–4540. https://doi.org/10.3233/JIFS-213206
11. S. Jiao, C. Wang, R. Gao, Y. Li, Q. Zhang, Harris Hawks optimization with multi-strategy search and application, *Symmetry-Basel*, **13** (2021), 2364. https://doi.org/10.3390/sym13122364
12. J. H. Holland, Genetic algorithms, *Sci. Am.*, **267** (1992), 66–73. https://doi.org/10.1038/scientificamerican0792-66
13. R. Storn, K. Price, Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces, *J. Glob. Optim.*, **11** (1997), 341–359. https://doi.org/10.1023/A:1008202821328
14. R. Nand, B. N. Sharma, K. Chaudhary, Stepping ahead firefly algorithm and hybridization with evolution strategy for global optimization problems, *Appl. Soft Comput.*, **109** (2021), 107517. https://doi.org/10.1016/j.asoc.2021.107517
15. X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, *IEEE Trans. Evol. Comput.*, **3** (1999), 82–102. https://doi.org/10.1109/4235.771163
16. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95 - International Conference on Neural Networks*, (1995), 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
17. X. S. Yang, Firefly algorithm, stochastic test functions and design optimization, *Int. J. Bio-Inspired Comput.*, **2** (2010), 78–84. https://doi.org/10.1504/IJBIC.2010.032124
18. S. Zhao, T. Zhang, S. Ma, M. Wang, Sea-horse optimizer: A novel nature-inspired meta-heuristic for global optimization problems, *Appl. Intell.*, **53** (2022), 11833–11860. https://doi.org/10.1007/s10489-022-03994-3
19. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris Hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872. https://doi.org/10.1016/j.future.2019.02.028
20. S. Arora, S. Singh, Butterfly optimization algorithm: A novel approach for global optimization, *Soft Comput.*, **23** (2019), 715–734. https://doi.org/10.1007/s00500-018-3102-4

21. J. O. Agushaka, A. E. Ezugwu, L. Abualigah, Dwarf mongoose optimization algorithm, *Comput. Methods Appl. Mech. Engrg.*, **391** (2022), 114570. https://doi.org/10.1016/j.cma.2022.114570
22. S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, *Science*, **220** (1983), 671–680. https://doi.org/10.1126/science.220.4598.671
23. I. Ahmadianfar, O. Bozorg-Haddad, X. Chu, Gradient-based optimizer: A new metaheuristic optimization algorithm, *Inf. Sci.*, **540** (2020), 131–159. https://doi.org/10.1016/j.ins.2020.06.037
24. S. Talatahari, M. Azizi, A. H. Gandomi, Material generation algorithm: A novel metaheuristic algorithm for optimization of engineering problems, *Processes*, **9** (2021), 859. https://doi.org/10.3390/pr9050859
25. S. Mirjalili, SCA: A sine cosine algorithm for solving optimization problems, *Knowl. Based Syst.*, **96** (2016), 120–133. https://doi.org/10.1016/j.knosys.2015.12.022
26. V. Goodarzimehr, S. Talatahari, S. Shojaee, S. Hamzehei-Javaran, Special relativity search for applied mechanics and engineering, *Comput. Meth. Appl. Mech. Eng.*, **403** (2023), 115734. https://doi.org/10.1016/j.cma.2022.115734
27. F. A. Hashim, K. Hussain, E. H. Houssein, M. S. Mabrouk, W. Al-Atabany, Archimedes optimization algorithm: A new metaheuristic algorithm for solving optimization problems, *Appl. Intell.*, **51** (2020), 1531–1551. https://doi.org/10.1007/s10489-020-01893-z
28. M. Jahangiri, M. A. Hadianfard, M. A. Najafgholipour, M. Jahangiri, M. R. Gerami, Interactive autodidactic school: A new metaheuristic optimization algorithm for solving mathematical and structural design optimization problems, *Comput. Struct.*, **235** (2020), 106268. https://doi.org/10.1016/j.compstruc.2020.106268
29. S. H. S. Moosavi, V. K. Bardsiri, Poor and rich optimization algorithm: A new human-based and multi populations algorithm, *Eng. Appl. Artif. Intell.*, **86** (2019), 165–181. https://doi.org/10.1016/j.engappai.2019.08.025
30. A. Naik, S. C. Satapathy, Past present future: A new human-based algorithm for stochastic optimization, *Soft Comput.*, **25** (2021), 12915–12976. https://doi.org/10.1007/s00500-021-06229-8
31. H. Bayzidi, S. Talatahari, M. Saraee, C. P. Lamarche, Social network search for solving engineering optimization problems, *Comput. Intell. Neurosci.*, **2021** (2021), 8548639. https://doi.org/10.1155/2021/8548639
32. Q. Zhang, R. Wang, J. Yang, K. Ding, Y. Li, J. Hu, Collective decision optimization algorithm: A new heuristic optimization method, *Neurocomputing,* **221** (2017), 123–137. ttps://doi.org/10.1016/j.neucom.2016.09.068
33. M. A. Al-Betar, Z. A. A. Alyasseri, M. A. Awadallah, I. Abu Doush, Coronavirus herd immunity optimizer (CHIO), *Neural Comput. Appl.*, **33** (2021), 5011–5042. https://doi.org/10.1007/s00521-020-05296-6
34. S. Mirjalili, S. M. Mirjalili, A. Lewis, Grey wolf optimizer, *Adv. Eng. Softw.*, **69** (2014), 46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007
35. D. H. Wolpert, W. G. Macready, No free lunch theorem for optimization, *IEEE Trans. Evol. Comput.*, **1** (1997), 67–82. https://doi.org/10.1109/4235.585893
36. H. T. K. Abdelbadie, A. T. M. Taha, H. M. Hasanien, R. A. Turky, S. M. Muyeen, Stability enhancement of wind energy conversion systems based on optimal superconducting magnetic energy storage systems using the Archimedes optimization algorithm, *Processes*, **10** (2022), 366. https://doi.org/10.3390/pr10020366

37. I. Neggaz, H. Fizazi, An intelligent handcrafted feature selection using Archimedes optimization algorithm for facial analysis, *Soft Comput.*, **26** (2022), 10435–10464. https://doi.org/10.1007/s00500-022-06886-3
38. J. Balakrishnan, C. Govindaraju, Multi-phase permanent magnet generator with Halbach array for direct driven wind turbine: a hybrid technique, *Energy Sources Part A-Recovery Util. Environ. Eff.*, **44** (2022), 5699–5717. https://doi.org/10.1080/15567036.2022.2086324
39. J. Annrose, N. H. A. Rufus, C. R. E. S. Rex, D. G. Immanuel, A cloud-based platform for soybean plant disease classification using Archimedes optimization based hybrid deep learning model, *Wirel. Pers. Commun.*, **122** (2021), 2995–3017. https://doi.org/10.1007/s11277-021-09038-2
40. H. R. Tizhoosh, Opposition-based learning: A new scheme for machine intelligence, in *Proceedings of International Conference on Computational Intelligence for Modelling, Control & Automation Jointly with International Conference on Intelligent Agents, Web Technologies & Internet Commerce*, (2006), 695–701. https://doi.org/10.1109/CIMCA.2005.1631345
41. B. S. Yildiz, N. Pholdee, S. Bureerat, A. R. Yildiz, S. M. Sait, Enhanced grasshopper optimization algorithm using elite opposition-based learning for solving real-world engineering problems, *Eng. Comput.*, **38** (2021), 4207–4219. https://doi.org/10.1007/s00366-021-01368-w
42. E. H. Houssein, B. E. D. Helmy, H. Rezk, A. M. Nassef, An efficient orthogonal opposition-based learning slime mould algorithm for maximum power point tracking, *Neural Comput. Appl.*, **34** (2022), 3671–3695. https://doi.org/10.1007/s00521-021-06634-y
43. A. Arcuri, G. Fraser, Parameter tuning or default values? An empirical investigation in search-based software engineering, *Empir. Softw. Eng.*, **18** (2013), 594–623. https://doi.org/10.1007/s10664-013-9249-9
44. M. H. Nadimi-Shahraki, S. Taghian, S. Mirjalili, An improved grey wolf optimizer for solving engineering problems, *Expert Syst. Appl.*, **166** (2021), 113917. https://doi.org/10.1016/j.eswa.2020.113917
45. C. Ma, H. Huang, Q. Fan, J. Wei, Y. Du, W. Gao, Grey wolf optimizer based on Aquila exploration method, *Expert Syst. Appl.*, **205** (2022), 117629. https://doi.org/10.1016/j.eswa.2022.117629
46. O. E. Turgut, M. S. Turgut, Local search enhanced Aquila optimization algorithm ameliorated with an ensemble of wavelet mutation strategies for complex optimization problems, *Math. Comput. Simul.*, **206** (2022), 302–374. https://doi.org/10.1016/j.matcom.2022.11.020
47. E. V. Altay, Hybrid Archimedes optimization algorithm enhanced with mutualism scheme for global optimization problems, *Artif. Intell. Rev.*, **56** (2022), 6885–6946. https://doi.org/10.1007/s10462-022-10340-z
48. N. Chopra, M. M. Ansari, Golden jackal optimization: a novel nature-inspired optimizer for engineering applications, *Expert Syst. Appl.*, **198** (2022), 116924. https://doi.org/10.1016/j.eswa.2022.116924
49. B. Abdollahzadeh, F. S. Gharehchopogh, S. Mirjalili, African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems, *Comput. Ind. Eng.*, **158** (2021), 107408. https://doi.org/10.1016/j.cie.2021.107408
50. S. Rajmohan, E. Elakkiya, S. R. Sreeja, Multi-cohort whale optimization with search space tightening for engineering optimization problems, *Neural Comput. Appl.*, **35** (2023), 8967–8986. https://doi.org/10.1007/s00521-022-08139-8

51. F. S. Gharehchopogh, M. H. Nadimi-Shahraki, S. Barshandeh, B. Abdollahzadeh, H. Zamani, CQFFA: A chaotic quasi-oppositional farmland fertility algorithm for solving engineering optimization problems, *J. Bionic Eng.*, **20** (2022), 158–183. https://doi.org/10.1007/s42235-022-00255-4
52. J. Zhao, Z. Gao, The heterogeneous Aquila optimization algorithm, *Math. Biosci. Eng.*, **19** (2022), 5867–5904. https://doi.org/10.3934/mbe.2022275
53. Y. Xiao, Y. Guo, H. Cui, Y. Wang, J. Li, Y. Zhang, IHAOAVOA: An improved hybrid Aquila optimizer and African vultures optimization algorithm for global optimization problems, *Math. Biosci. Eng.*, **19** (2022), 10963–11017. https://doi.org/10.3934/mbe.2022512
54. A. Seyyedabbasi, F. Kiani, Sand cat swarm optimization: A nature-inspired algorithm to solve global optimization problems, *Eng. Comput.*, **39** (2022), 2627–2651. https://doi.org/10.1007/s00366-022-01604-x
55. F. S. Gharehchopogh, An improved tunicate swarm algorithm with best-random mutation strategy for global optimization problems, *J. Bionic Eng.*, **19** (2022), 1177–1202. https://doi.org/10.1007/s42235-022-00185-1
56. P. Chen, S. Zhou, Q. Zhang, N. Kasabov, A meta-inspired termite queen algorithm for global optimization and engineering design problems, *Eng. Appl. Artif. Intell.*, **111** (2022), 104805. https://doi.org/10.1016/j.engappai.2022.104805
57. S. Duan, H. Luo, H. Liu, An elastic collision seeker optimization algorithm for optimization constrained engineering problems, *Math. Probl. Eng.*, **2022** (2022), 1344667. https://doi.org/10.1155/2022/1344667
58. S. Mirjalili, Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm, *Knowl. Based Syst.,* **89** (2015), 228–249. https://doi.org/10.1016/j.knosys.2015.07.006
59. H. Yu, S. Qiao, A. A. Heidari, C. Bi, H. Chen, Individual disturbance and attraction repulsion strategy enhanced seagull optimization for engineering design, *Mathematics*, **10** (2022), 276. https://doi.org/10.3390/math10020276
60. G. Hu, J. Zhong, B. Du, G. Wei, An enhanced hybrid arithmetic optimization algorithm for engineering applications, *Comput. Meth. Appl. Mech. Eng.*, **394** (2022), 114901. https://doi.org/10.1016/j.cma.2022.114901

AIMS Press