Expert Systems With Applications 236 (2024) 121219

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Dung beetle optimization algorithm based on quantum computing and multi-strategy fusion for solving engineering problems

Fang Zhu [a], Guoshuai Li [a,*], Hao Tang [a], Yingbo Li [a], Xvmeng Lv [a], Xi Wang [b]

[a] School of Computer and Communication Engineering, Northeastern University, Qinhuangdao, Hebei, Qinhuangdao, 066004, PR China
[b] School of Public Safety and Emergency Response, Kunming University of Science and Technology, Yunnan, Kunming, 650093, PR China

ARTICLE INFO

Keywords:
Quantum computing
Dung beetle optimization algorithm
Good point collection
Inertia factor

ABSTRACT

The Dung beetle optimization algorithm is a kind of group intelligence optimization algorithm proposed by Jiankai Xue in 2022, which has the characteristics of strong optimization-seeking ability and fast convergence but suffers from the defect of easily falling into local optimum at the late stage of optimization-seeking as other group intelligence optimization algorithms. To address this problem, this paper proposes a dung beetle search algorithm (QHDBO) based on quantum computing and a multi-strategy hybrid. The good point set strategy is used to initialize the initial population of dung beetles . That makes the initial population more evenly distributed, and reduces the likelihood of the algorithm falling into a local optimum solution. The convergence factor and dynamic balance between the number of Spawning and foraging dung beetles is proposed. That allows the algorithm to focus on the global search in the early stages and local exploration in the later stages. The quantum computing based t-distribution variation strategy is used to variate the optimal global solution, that prevents the algorithm from falling into a local optimum. To verify the performance of the QHDBO algorithm, this paper compares QHDBO with six other swarm intelligence algorithms through 37 test functions and practical engineering application problems. The experimental results show that the improved dung beetle optimization algorithm significantly improves convergence speed and optimization accuracy and has good robustness.

1. Introduction

An optimization problem is a search for the maximum or minimum of an objective function under a set of constraints. Optimization problems exist widely in many fields,such as UAV path planning (Yu et al., 2023), image processing (Luo et al., 2023), mechanical design (Shen et al., 2023), and social media sentiment analysis (Yildirim, 2022). However, in real life, most problems abstracted to obtain the optimization problem are black box problems. The specific expressions, gradient information, and derivability are unknown, so it is difficult to solve this problem using the traditional modified optimization methods.

Meta-heuristic algorithms are self-organizing and self-learning compared with traditional optimization methods, therefore they can effectively deal with problems that are difficult to be solved by conventional optimization algorithms. The mainstream metaheuristic algorithms mainly contain two types. The first is evolutionary algorithms, which are inspired by the evolutionary process of organisms in nature, and evolve the optimal solution to the problem by simulating the genetic, crossover and mutation operations of organisms in nature. Genetic Algorithms (GA) (Goldberg & Holland, 1988) are the classical evolutionary algorithms. The second is the swarm intelligence optimization algorithm, which is an algorithm that solves the optimization problem by simulating the behaviour of a population of organisms in the natural world. Compared with evolutionary algorithms, these algorithms are easy to implement, and the algorithm's performance is not weakened. In recent years, many papers have appeared in the field of swarm intelligence optimization algorithms, such as Ant Colony optimization algorithm (Dorigo et al., 2006) (ACO), Particle Swarm optimization algorithm (Kennedy & Eberhart, 1995) (PSO), Sparrow Optimization algorithm (Xue & Shen, 2020) (SSA), Whale Optimization algorithm (Mirjalili & Lewis, 2016) (WOA), Grey Wolf optimization algorithm (Mirjalili et al., 2014) (GWO), Dung Beetle optimization algorithm (Xue, 2023) (DBO), Butterfly optimization algorithm (Arora, 2019) (BOA), Sine Cosine optimization algorithm (Mirjalili, 2016) (SCA),Sand Cat swarm optimization algorithm (Seyyedabbasi, 2022) (SCSO), Manta Ray Foraging optimization algorithm (Zhao

* Corresponding author.
E-mail addresses: zhufang@neuq.edu.cn (F. Zhu), 2272206@stu.neu.edu.cn (G. Li), thtangh@163.com (H. Tang), lybydck@163.com (Y. Li), lxm15503195893@163.com (X. Lv), wx17860552813@163.com (X. Wang).

https://doi.org/10.1016/j.eswa.2023.121219
Received 30 May 2023; Received in revised form 28 July 2023; Accepted 14 August 2023
Available online 22 August 2023

et al., 2020) and so on. They are applied in various fields and have achieved good results.

The Dung Beetle Optimization algorithm is a novel swarm intelligence optimization algorithm proposed in 2022, which simulates the habits of dung beetles in nature to build a search framework based on the "Rall-rolling-Spawning-Foraging-Stealing" model. However, according to the "No Free Lunch Theorem" (Wolpert & Macready, 1997), no single swarm intelligence optimization algorithm can solve all optimization problems, and each swarm intelligence optimization algorithm has limitations and restrictions, so many scholars have proposed improvements to these original swarm intelligence optimization algorithms. For example:

- A multi-strategy hybrid sparrow algorithm is proposed by Wu et al. (2023). The paper presents a series of strategies to optimize the sparrow algorithm : (1) using quantum computing to improve circular chaos mapping to make the initial population distribution more uniform (2) improving the update strategy of the discoverer in the sparrow algorithm to accelerate the convergence of the algorithm (3) improving multiple stages of the algorithm using mutation factors obeying t-distribution so that the algorithm obtains better global exploration ability in the early stage and stronger local exploration ability in the later stage.
- Yuchen Duan et al. proposed an optimization algorithm that mixes the grey wolf algorithm and the sine and cosine algorithm (Duan & Yu, 2023). The specific improvements are: (1) To simulate the real wolf hunting process, the positive cosine function is used to replace the linear function in the original algorithm. (2) A weight-based position updating strategy is proposed, which enables the algorithm to solve the high-latitude better optimization problem. (3) To explore the regions adjacent to the best positions of individuals to prevent missing solutions.
- Yongjun Sun et al. proposed an adaptive whale algorithm based on Lévy flight and quadratic interpolation improvement (Sun et al., 2018). The specific improvements are (1) enhancing the ability of the whale algorithm to jump out of the optimal local solution based on the Lévy flight characteristics (frequent short-distance search and occasional long-distance exploration), (2) adjusting the local exploration and global search ability by adaptive parameters so that the algorithm does not converge too fast leading to premature maturity or converge too slow leading to insufficient accuracy of the solution.

The dung beetle algorithm is a new type of algorithm that shows good performance in the original text, but the dung beetle optimization algorithm also has some defects, for example: in the reproduction phase, if the local optimum is taken at the origin, i.e., point 0, it will lead to the spawning dung beetles are concentrated in point 0, resulting in the failure of the algorithm iteration; The convergence factor of the dung beetle algorithm does not balance the global search in the early stage of the algorithm and the local exploration in the late stage of the algorithm, which leads to a decrease in solution accuracy. Therefore, this paper proposes a series of improvement measures as follows:

- The population is initialized using the good point set strategy to increase the diversity of the initial population and thus avoid getting trapped in a locally optimal solution.
- Dynamically balance the number of foraging dung beetles and spawning dung beetles and improve the convergence factor, which in turn enables the algorithm to obtain better global search capability in the early stage and better local exploration capability in the later stage.
- The optimal spawning region in the original algorithm is improved so that the algorithm can explore the solution space more fully.
- Using a t-distribution variation strategy based on quantum computing enables the algorithm to jump out of the local optimal solution in time.

2. DBO

The basic dung beetle algorithm performs population location updates by simulating four behaviours of dung beetles in nature (Rolling, Spawning, Foraging, and Stealing).

2.1. Rollerball dung beetle

In nature, the dung beetle must navigate through the sun to make a straight line for the route it takes during its ball rolling. Eq. (1) was used in the original paper to update the position of the rolling dung beetle:

$$\begin{aligned} &x_i(t+1) = x_i(t) + \alpha \cdot k \cdot x_i(t-1) + b \cdot \Delta x \\ &\Delta x = \left| x_i(t) - X^w \right| \end{aligned} \tag{1}$$

t denotes the number of current iterations, and $x_i(t)$ denotes the position of the dung beetle in t iterations. In the original text α indicates whether the dung beetle deviates from its original direction, and the value of α is set to 1 or -1 by probability, with 1 indicating no deviation and -1 indicating deviation. $k \in (0, 0.2]$ indicates that the defect factor is set to 0.1 in the original text, and b indicates a constant belonging to $[0, 1]$, assigned to 0.3 in the code. X^w represents the global worst value. Δx is used to simulate solar illumination and a larger Δx indicates that the dung beetle is further away from the light source. In nature, when a dung beetle encounters an obstacle, the dung beetle will get a new rolling direction by dancing behaviour. To simulate this situation, the original article simulates whether the dung beetle encounters an obstacle during the ball-rolling process by probability, and when it encounters an obstacle, the original article uses a tangent function to get a new rolling direction in order to simulate the dancing behaviour of the dung beetle. The update of the position of the rolling dung beetle becomes Eq. (2)

$$x_i(t+1) = x_i(t) + \tan(\theta)|x_i(t) - x_i(t-1)| \tag{2}$$

$\theta \subseteq [0, \pi]$,The position is not updated when $\theta = 0$, $\pi/2$ and π.

2.2. Spawning dung beetles

In nature, dung beetles select a safe area for spawning, and to simulate this behaviour, the original paper proposes a boundary selection strategy to model this area, as defined below:

$$\begin{aligned} &Lb^* = \max\left(X^* \times (1-R), Lb\right) \\ &Ub^* = \min\left(X^* \times (1+R), Ub\right) \end{aligned} \tag{3}$$

Lb^* and Ub^* represent the lower bound and upper bound of the spawning region, X^* is the current local optimum, $R = 1 - t/Tmax$, and $Tmax$ represents the maximum number of iterations. Once the spawning dung beetle determines the optimal spawning region, it spawns in this region. In the original text, each spawning dung beetle spawning is an update of the dung beetle's position. The spawning region is dynamically changing so that the search for the region where the current optimal solution is located can be guaranteed while preventing falling into a local optimum. The position update of the spawning dung beetle is given by Eq. (4)

$$X_i(t+1) = X^* + b_1 \times (X_i(t) - Lb^*) + b_2 \times (X_i(t) - Ub^*) \tag{4}$$

where b_1 and b_2 are random variables of size $1 \times Dim$ and Dim represents the dimensionality of the optimization problem. The dimension of the problem

2.3. Foraging dung beetles

In nature, dung beetles foraging will also be like laying eggs to choose a safe area. The original text for this area has chosen to redefine the specific definition as the formula:

$$\begin{aligned} Lb^b &= \max\left(X^b \times (1-R), Lb\right) \\ Ub^b &= \min\left(X^b \times (1+R), Ub\right) \end{aligned} \tag{5}$$

Here X^b denotes the optimal global position, Lb^b and Ub^b denote the lower and upper bounds of the optimal foraging region, Lb and Ub represent the lower and upper boundaries for problem solving, and each foraging behaviour of the foraging dung beetle corresponds to one position update of the foraging dung beetle and the foraging dung beetle position is updated as follows:

$$x_i(t+1) = x_i(t) + C_1 \times \left(x_i(t) - Lb^b\right) + C_2 \times \left(x_i(t) - Ub^b\right) \tag{6}$$

C_1 is a random number obeying normal distribution and C_2 is a vector belonging to $[0,1]$ of size $1 \times Dim$.

2.4. Stealing dung beetles

In nature, some dung beetles choose to steal dung balls from other dung beetles. To simulate this behaviour, the original article uses the optimal global location X^b to act as the location of the contested dung balls. The stealing behaviour of the stealing dung beetle is defined as the location update of the stealing dung beetle, and the location update of the stealing dung beetle is given by the following equation:

$$x_i(t+1) = X^b + S \cdot g \cdot \left(\left|x_i(t) - X^*\right| + \left|x_i(t) - X^b\right|\right) \tag{7}$$

S denotes a constant in the original text with a value of 0.5, g is the size of the random variable, and Dim represents the problem's dimensionality. In the original text, the numbers of rolling dung beetles, breeding dung beetles, foraging dung beetles, and stealing dung beetles are 6, 6, 7, and 11, respectively.

2.5. DBO algorithm implementation steps

Algorithm 1 The framework of the DBO algorithm

```
Require: The maximum iteration $TMAX$,the size of the particle's population N.
Ensure: Optimal position $X^b$ and its fitness value $f_b$
 1: Initialize the particle's population i ← 1,2,.....,N and define its relevant parameters
 2: while $t \le TMAX$ do
 3:   for $i = 1$ to Number of rollingdung beetles do
 4:     $\alpha = rand(1)$
 5:     if $\alpha \le 0.9$ then
 6:       Update Rolling Dung Beetle Location by Eq.(1).
 7:     else
 8:       Rolling the ball in the encounter of obstacles by Eq.(2) to update.
 9:     end if
10:   end for
11:   The value of the nonlinear convergence factor is calculated by $R = 1 - t/TMAX$.
12:   for $i = 1$ to Number of Spawning dung beetles do
13:     Updating of Spawning dung beetles by Eq.(3) and Eq.(4).
14:   end for
15:   for $i = 1$ to Number of Foraging dung beetles do
16:     Update foraging dung beetles by Eq.(5) and Eq.(6).
17:   end for
18:   for $i = 1$ to Number of Stealing Dung Beetles do
19:     Use Eq.(7) to update the location of the stealing dung beetle
20:   end for
21: end while
22: Return $X^b$ and its fitness value $f_b$;
```

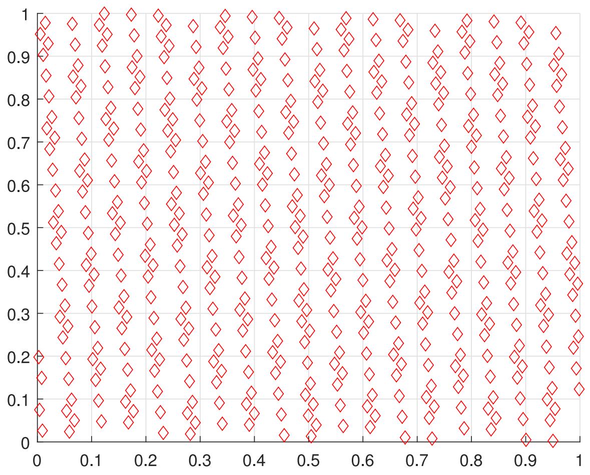

Fig. 1. Good point set generates point set diagram.

3. Improving the dung beetle optimization algorithm

3.1. Motivation

The DBO algorithm outperforms many intelligent optimization algorithms in solving optimization problems, but it is still a challenge for the DBO algorithm to obtain the ideal optimal solution for some optimization problems, so a series of improvements are proposed to increase the accuracy of the DBO algorithm for solving optimization problems.

3.2. Population initialization strategy based on good point set

Hua Luogeng and other well-known mathematicians in China-proposed the good point set (Hua Luogeng, 1978). Suppose there exists such a set $P_n(k) = \left\{\left(\left\{r_1^{(n)} \cdot k\right\}, \left\{r_2^{(n)} \cdot k\right\}, \ldots, \left\{r_s^{(n)} \cdot k\right\}\right), 1 \le k \le n\right\}$ in the Euclidean space H_s of dimension s that holds and its deviation satisfies $\varphi(n) = C(r,\varepsilon)\, n^{-1+\varepsilon}$, then we can call this set a good point set, and r is a good point. The value of the good point r is $r^i = \left\{2cos\left(\frac{2\pi i}{p}\right)\right\}; 1 \le i \le n$, where p is the smallest prime number that satisfies $p \ge 2 * s + 3$. It can be mapped to the search space by Eq. (11). Where $upper_j$ is the upper bound of the search space and low_j is the lower bound.

$$x_i(k) = \left(upper_j - low_j\right)\left\{P_n(k)\right\} + low_j \tag{8}$$

Fig. 1 is the initial population distribution of 500 points generated by the good point set in two-dimensional space, and Fig. 2 is the initial population distribution of 500 points generated by the random strategy. To prove that the point set generated by the good point set is more uniform than the randomly generated point set, the two-dimensional coordinate axes are divided into small $1 * 1$ unit squares to see whether the points falling in each small square are uniform. From Fig. 3 ,we can see that the points generated by the good point set are more uniform than the randomly generated points. Therefore, the population diversity can be effectively enhanced by introducing the good point set, which is helpful for the algorithm to get rid of the local optimum.

3.3. Improved convergence factor

From $R = 1 - t/Tmax$, we can see that the area of the optimal spawning and foraging area decreases gradually as the Convergence factor R decreases. The area surrounded by the optimal spawning and foraging area is larger in the early stage, which enables the algorithm

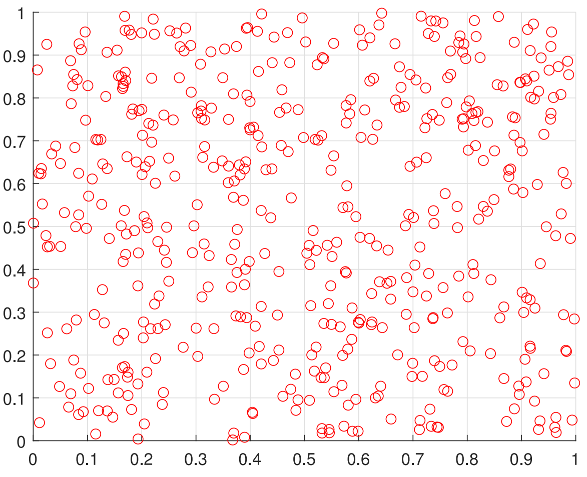

Fig. 2. Random strategy to generate point set graphs.

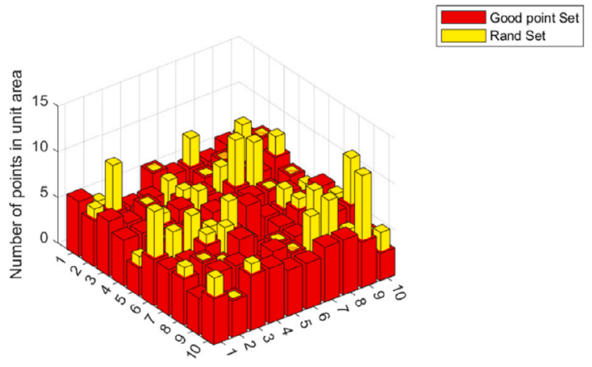

Fig. 3. Point distribution map.

to have better global search ability. The area surrounded by the optimal spawning and foraging area is smaller in the later stage, which enables the algorithm to have better local exploration ability. We want the algorithm to get better global exploration ability in the early stage to increase the chance of exploring better points and get better local exploration ability in the late stage to increase the speed of convergence of the algorithm, so we need to adjust the factor to have a slower decreasing speed in the early stage and a faster-decreasing speed in the late stage, so this paper improves the inertia factor in the original paper, and the improved Convergence factor R is shown in Eq. (9) is shown.

$$R = (\cos(\pi * (t/T\max)) + 1) * 0.5 \tag{9}$$

From Fig. 4, we can see that the improved boundary Convergence factor decreases more slowly in the early stage relative to the Convergence factor in the original text, which can make the optimal foraging and spawning boundaries larger and thus obtain better global search ability, and in the later stage the improved boundary Convergence factor decreases more quickly in the later stage relative to the Convergence factor in the original text, thus speeding up the convergence ability of the algorithm.

3.4. Balancing global search and local exploration

- The number of foraging dung beetles and spawning dung beetles is kept constant in the original paper. From the formula, we can see that the spawning dung beetles contribute local exploration capability to the algorithm because the location of each update of the Spawning dung beetles are strictly controlled in the optimal spawning boundary, i.e., the vicinity of the current optimal location, and from the formula, we can see that the foraging dung beetles contribute global exploration capability to the algorithm because the location update of the foraging dung beetles are not strictly limited to the optimal foraging area, and C_1 is a random variable that obeys a normal distribution and may have a large step size during the iterative process. We want the algorithm to obtain better global search ability in the early stage and a better local exploration ability in the later stage, which can make fewer spawning dung beetles and more foraging dung beetles in the early stage of the iteration and more spawning dung beetles and less foraging dung beetles in the later stage, so this paper merges the two populations and designs a dynamic critical value formula Eq. (10):

$$P = 0.2 + t/T\max * 0.6 \tag{10}$$

- There is a failure in the original paper for the selection of the optimal spawning region, such as the current local optimal location X^* has a component of 0 in each dimension, which will lead to the upper bound of the optimal Spawning region being equal to the lower bound,i.e. Lb^* and Ub^* has a component of 0 in each, which will lead to the location of all Spawning dung beetles being concentrated at a single point,i.e. X^* location. In order to solve this problem, this paper defines the optimal Spawning region optimization as a region surrounded by an n-dimensional ball with X^* as the centre of the circle, the expression of this ball is

$$(X - X_1^*)^2 + (X - X_2^*)^2 + \cdots + (X - X_n^*)^2 = Ridus^2 \tag{11}$$

where $Ridus = (Ub - Lb)/2 * R$, Ub and Lb are the upper and lower bounds of the optimization problem, respectively. In the early iteration of the algorithm, the value of Ridus is larger, and the dung beetle position is updated to explore the solution space more fully so as to avoid falling into the local optimum, and in the later iteration of the algorithm, the value of Ridus is smaller, and the dung beetle position is updated to explore the region near the current local optimum more fully so as to speed up the convergence of the algorithm. The spawning dung beetle position

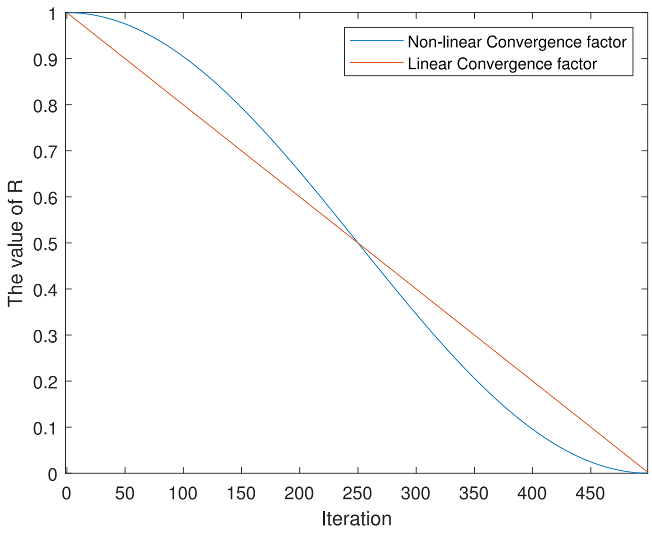

Fig. 4. Comparison of convergence factors.

update becomes:

$$\begin{aligned}
&X_1^i(t+1) = X_1^* + Ridus * K * \cos\theta_1 \\
&X_2^i(t+1) = X_2^* + Ridus * K * \sin\theta_1 \cos\theta_2 \\
&X_3^i(t+1) = X_3^* + Ridus * K * \sin\theta_1 \sin\theta_2 \cos\theta_3 \\
&\ldots \\
&X_{n-1}^i(t+1) = X_{n-1}^* + Ridus * K * \sin\theta_1 \sin\theta_2 \ldots \sin\theta_{,n-2} \cos\theta_{n-1} \\
&X_n^i(t+1) = X_n^* + Ridus * K * \sin\theta_1 \sin\theta_2 \ldots \sin\theta_{,n-2} \sin\theta_{n-1}
\end{aligned} \tag{12}$$

The formula is the conversion of n-dimensional spherical coordinates to Euclidean coordinates, and K belongs to the random number in the scaling factor taking the value of only $[0,1]$.$\theta_1 \ldots \theta_{n-2}$ is a random number in $[0, pi]$, and θ_{n-1} is a random number in $[0, 2pi]$.

3.5. Quantum computing based t-distribution variation

The quantum algorithm for prime factorization proposed by Shor in 1994 caused a huge sensation at that time, with a time complexity of only, and thus quantum computing is gaining more and more attention from scholars around the world. The quantum heuristic evolutionary algorithm was originally developed by Han and Kim (2000), who used the properties of quantum bits (the biggest difference from traditional computing is that it uses quantum bits instead of traditional binary bits for computation, and the difference between quantum bits and traditional binary bits is that it can have multiple states at the same time) to encode individuals, which can ensure the good exploration of the search space even with few initial points. Rui Wu uses the quantum bit feature to optimize the circular chaos mapping to initialize the population, thus obtaining a better -initialized population of individuals and increasing the accuracy of the algorithm for solving the optimal value.

In this paper, we propose a novel variation strategy using quantum computing to interfere with the variation factor to perform various operations on the optimal position X^b, enabling the algorithm to increase the ability to jump out of the optimal local solution. Firstly, this paper introduces the concept of quantum bits, which are the objects underlying the information in quantum computing. Quantum bits can be classified as single quantum bits, double quantum bits, and multiple quantum bits, and the method proposed in this paper uses single quantum bits. The state of a single quantum bit is given by the unit's two-dimensional column vector $\begin{bmatrix}\alpha \\ \beta\end{bmatrix}$, i.e., $\alpha^2 + \beta^2 = 1$, the quantum state vector in $\begin{bmatrix}1 \\ 0\end{bmatrix}$ and $\begin{bmatrix}0 \\ 1\end{bmatrix}$ play a special role because any quantum state vector can be written as the sum of these two states i.e. $|\psi\rangle = \alpha \begin{bmatrix}1 \\ 0\end{bmatrix} + \beta \begin{bmatrix}0 \\ 1\end{bmatrix}$, where ψ denotes quantum bits, so the above properties of quanta can be used to construct the mutation factor by first establishing a quantum space consisting of multiple quantum bits at the same latitude as the problem,i.e:

$$|Z_R\rangle = \begin{bmatrix}\alpha_1, \alpha_2 \ldots \alpha_{\dim} \\ \beta_1, \beta_2 \ldots \beta_{\dim}\end{bmatrix} \tag{13}$$

Then the optimal position X^b needs to be mapped to the quantum space, which is done in this paper using the following method:

$$\begin{aligned}
&\alpha_j = \frac{X_j^b}{\|X^b\|} \\
&\beta_j = P_j * sqrt(1 - (\frac{X_j^b}{\|X^b\|})^2)
\end{aligned} \tag{14}$$

where x_j^b denotes the value on dimension j of the optimal global position, $\|X^b\|$ denotes the modal length of x^b, and P is set to 1 or -1 by probability so that the diversity of points after mutation can be guaranteed, and the quantum space can be converted to the solution space by the following Eq. (13) after performing the transformation.

$$\begin{aligned}
&X_{tem\alpha} = [\alpha_1, \alpha_2 \ldots \alpha_{\dim}] * \|X^b\| \\
&X_{tem\beta} = [\beta_1, \beta_2 \ldots \beta_{\dim}] * \|X^b\|
\end{aligned} \tag{15}$$

In order to explore the solution space more fully, this paper uses the quantum rotation (Xiong et al., 2018) technique to rotate $|Z_R\rangle$, which can increase the diversity of the population and thus avoid falling into local optima, as follows:

$$R(\theta) = \begin{bmatrix}\cos\theta - \sin\theta \\ \sin\theta \cos\theta\end{bmatrix} \tag{16}$$

$$|Z_R\rangle = R(\theta) \begin{bmatrix}\alpha_1, \alpha_2 \ldots \alpha_{\dim} \\ \beta_1, \beta_2 \ldots \beta_{\dim}\end{bmatrix} \tag{17}$$

$\theta \in [0, 2\pi]$. Then the variation is carried out by the following Eq. (14).

$$X^b = \begin{cases} X^b + trnd(t) \cdot X_{tem\alpha} \\ X^b + trnd(t) \cdot X_{tem\beta} \\ X^b + trnd(t) \cdot X_{Rtem\alpha} \\ X^b + trnd(t) \cdot X_{Rtem\alpha} \end{cases} \tag{18}$$

where $X_{Rtem\alpha}$ and $X_{Rtem\beta}$ denote the points in the solution space mapped to the quantum bits after rotation,trnd(Iter) is the variance factor that obeys the t-distribution with degree of freedom Iter(Current number of iterations). The individual positions are updated by elite selection.

The t-distribution (Wu et al., 2023), also known as the student distribution, is a common sampling method in statistics. Its distribution state is related to the degrees of freedom, and the smaller the degrees of freedom, the flatter the curve. When the degrees of freedom are 1, the t-distribution is converted to a Gaussian distribution, and when the degrees of freedom tend to infinity, the t-distribution becomes a standard normal-terrestrial distribution. Its probability density function is Eq. (19). Fig. 5 below shows the image of the function corresponding to different degrees of freedom. In the algorithm, when performing the mutation operation, when the mutation factor is close to 0, it can make the algorithm focus on local exploitation and,vice versa, focus on global exploration (Jianhua & Zhiheng, 2021). For the sake of observation, defining the value of the mutation factor obeying the t-distribution lies in $[-1, 1]$ makes the algorithm focus more on the local exploitation stage, and when the value of the mutation factor obeying the t-distribution lies in $[-\infty, -1] \cup [1, +\infty]$ makes the law focus more on the global exploration.

$$\begin{aligned}
&\mathrm{trnd}(x,n) = \frac{\Gamma(\frac{n+1}{2})}{\sqrt{n\pi} \times \Gamma(\frac{n}{2})}(1 + \frac{x^2}{n})^{-1\frac{n+1}{2}}, -\infty < x < +\infty \\
&\Gamma(a) = \int_0^\infty x^{a-1} e^{-x} dx
\end{aligned} \tag{19}$$

As shown in Fig. 5, the value of the mutation factor of the t-distribution function obeying degree of freedom 0.1 leads the algorithm to local exploration with probability $P = \int_{-1}^{1} trnd(Iter, x) dx$, i.e., the area surrounded by the t-distribution function at $x = -1$ and $x = 1$ and the x-axis, and trnd(Iter) is the t-distribution obeying degree of freedom Iter of the probability density function. The probability of leading the algorithm into the global search that is $1 - \mathrm{P}$, according to Fig. 5,can be seen when the freedom gradually increases, leading the algorithm into the local exploration of the probability of P is also more significant. This paper hopes that the algorithm in the first iteration to obtain a larger search space, in the later more active local exploration, which can increase the iteration speed of the algorithm and the convergence rate of the algorithm can also be guaranteed, so this paper is designed to use the iteration. The number of iterations is used to determine the value of degrees of freedom.

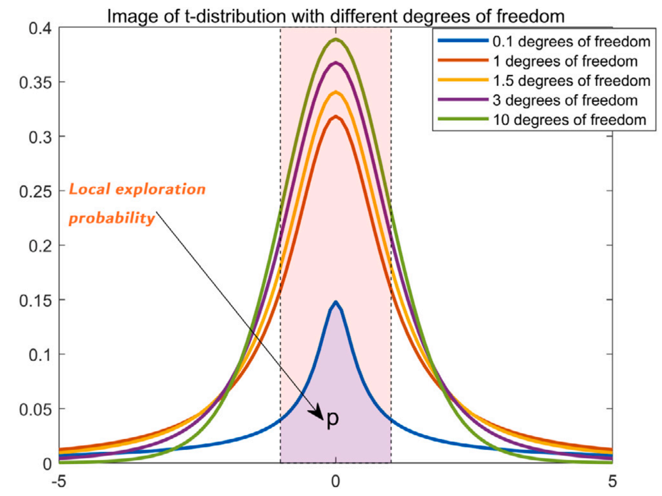

Fig. 5. t distribution image.

3.6. Complexity analysis

Assume that N represents the population size, D represents the dimension of the optimization problem and Iter represents the total number of iterations. The time complexity of the initial phase is $T1 = O(N * D)$ and the complexity during iteration is $T2 = O(Iter * N * D)$ so the complexity of the DBO algorithm is $T1 + T2$ which is $O(N)$. The complexity of the QHDBO algorithm is the same as the DBO algorithm in the initialization phase which is $T1$. Assuming that spawning dung beetles and foraging dung beetles account for the total proportion of dung beetles is $P\%$ so the complexity of the improved dung beetles updating the spawning dung beetles is $T3 = O((P * N * \sum_{i=1}^{Iter} (0.2 + 0.6 * (i/Iter))) * D) = O(\frac{Iter}{2} P * N * D + 0.3 * N * P * D)$ so the complexity of the updating of the other dung beetles remaining is $T4 = O((1 - P) * N * D)$ The complexity of the QHDBO algorithm using mutation strategy is $T5 = O(P * Iter)$,so the complexity of the QHDBO algorithm is $T1 + T3 + T4 + T5$ which is $O(N)$, so the QHDBO algorithm has the same complexity as the DBO algorithm and the performance is not based on higher complexity.

3.7. QHDBO algorithm implementation steps

4. Experimental results and discussion

In this paper, we evaluate the performance of the QHDBO algorithm using six comparison algorithms and the results of the QHDBO algorithm for 37 test functions. The test functions are from CEC2017 (Wu et al., 2016) and CEC2020 (Liang et al., 2019) (The function details are presented in Table 1), respectively, and the comparison algorithms are selected from six popular metaheuristics (GA, WOA, BOA, SCA, SSA, DBO) in this paper. The parameters of the comparison algorithms are set in Table 2. For the fairness of the experiment, the initial population size of all algorithms is set to 30, and the maximum number of iterations is set to 500 in this paper. To eliminate the chance of the experiment, the evaluation criteria used in this paper are the mean and standard deviation of the solution results of the six comparison algorithms and the QHDBO algorithm run 100 times independently on each test function.

The experiments were conducted on a computer system with an Intel(R) i7-10875H processor, 16 GB RAM and running a 64-bit version of Microsoft Windows 10. The MATLAB (R2022a) programming environment was used to implement the source code. This configuration provides a reliable and an efficient platform for conducting experiments. On the CEC2017 test function, the program takes about 7 h to execute 100 times. On the CEC2020 function, the program takes about 1 h to execute 100 times.

In Table 2, the BOA algorithm parameters pe stands for power_exponent, sm stands for sensory_modality, and the DBO algorithm parameters, RDB stands for rolling dung beetle, EDB stands for Spawning dung beetle, FDB stands for foraging dung beetle, and SDB stands for stealing dung beetle. The degree row field in Tables 3–5, is a ranking of the mean values, e.g. 1 means that the algorithm has the smallest mean value of the solution after 100 runs, i.e. the best search capability.

4.1. Results and analysis of CEC2017 benchmark functions

CEC2017 consists of 29 single objective test functions, of which F1 and F2 are single peak functions, F3-F10 are simple multimodal functions, F11-F20 are hybrid functions, and F20-F29 are composite functions. The F2 function lost its experimental power due to uncontrollable factors, so we did not perform experiments on F2. Figs. 6 and 7 show the average ranking of the solution results for QHDBO and its comparison algorithm after solving each function in CEC2017 for 100 independent runs. The analysis for the test results is discussed as follows:

4.1.1. Analysis of CEC2017 statistical results.

The experimental results for the 30 and 100-dimensional cases are recorded in Tables 3 and 4. Tables 3 and 4 show the mean and standard deviation of the objective function values for each algorithm. The results of these experimental analyses will be discussed as follows:

(1) Experimental results for the single-peaked problem F1 show that QHDBO excels in its ability to find the global optimum. In the 30-dimensional experiments, QHDBO only partially outperformed GA on F1 but quickly outperformed GA to converge to the optimum, however, in the 100-dimensional experiments, QHDBO outperformed on the F1 function, showing a lead from start to finish. This suggests that QHDBO is more likely to find and converge to the global optimum for single-peaked problems, validating its greater global exploration and exploitation capabilities.
(2) Experimental results on simple multimodal problems F3-F10 show that QHDBO outperforms the other six comparison algorithms. In the 30-dimensional experiments, QHDBO converges later than the other functions on function F3 and is slightly weaker than GA; it is weaker than GA in the first and middle stages on function F8 and eventually outperforms GA; it performs slightly weaker than SSA on function F10; in the 100-dimensional experiments, QHDBO performs comparably to the other algorithms on function F3, weaker than SSA on functions F9 and F10, and on F10 and WOA was comparable to GA. Nevertheless, QHDBO shows excellent performance in the remaining functions. These results further validate the superiority and even better robustness of the QHDBO algorithm in solving complex problems.
(3) QHDBO shows significant performance in the experimental results from F11-F20 when dealing with mixed problems. In the 30-dimensional experiments, QHDBO is comparable to GA on F11, weaker than GA on F14, and slightly weaker than SSA and GA on F15 and F18; while in 100 dimensions, QHDBO outperforms and even far outperforms the other algorithms on some functions in F12, F15, F16, F17, F19 and F20. qHDBO The excellent performance here is due to its diverse solution search strategies. Its powerful global search capabilities make it excellent at solving mixed problems.
(4) In terms of solving composite problems, QHDBO's experimental results on functions F21-F29 demonstrate its strong competitiveness. In all 30-dimensional experiments, QHDBO consistently outperforms other comparative algorithms, and in 100-dimensional experiments, QHDBO outperforms other comparative algorithms in all remaining functions, except for SSA and

Algorithm 2 The framework of the QHDBO algorithm

```
Require: The maximum iterations $TMAX$,the size of the particle's population N,Obtain an initialized population X of dung beetles using the good
    point set strategy.
Ensure: Optimal position $X^b$ and its fitness value $f_b$
 1: while $t \le TMAX$ do
 2:   for $i = 1$ to $Number\ of\ rollingdung\ beetles$ do
 3:     $\alpha = rand(1)$
 4:     if $\alpha < 0.9$ then
 5:       Update Rolling Dung Beetle Location by Eq.(1).
 6:     else
 7:       Rolling the ball in the encounter of obstacles by Eq.(2) to update.
 8:     end if
 9:   end for
10:   The value of the nonlinear convergence factor is calculated by Eq.(9).
11:   for $i = 1$ to $Number\ of\ Foraging\ dung\ beetles\ and\ Spawning\ dung\ beetles$ do
12:     if RAND < Eq.(10) then
13:       $Ridus = (Ub - Lb)/2 * Eq.(9)$,Updating Spawning dung beetles via Eq.(12).
14:     else
15:       Update foraging dung beetles by Eq.(5) and Eq.(6).
16:     end if
17:   end for
18:   for $i = 1$ to $Number\ of\ Stealing\ Dung\ Beetles$ do
19:     Use Eq.(7) to update the location of the stealing dung beetle
20:   end for
21:   The variation of the global optimal solution by using quantum computational interference obeying the t-distribution of the variation factor
      and selecting the optimal point by elite selection strategy.
22: end while
23: Return $X^b$ and its fitness value $f_b$.
```

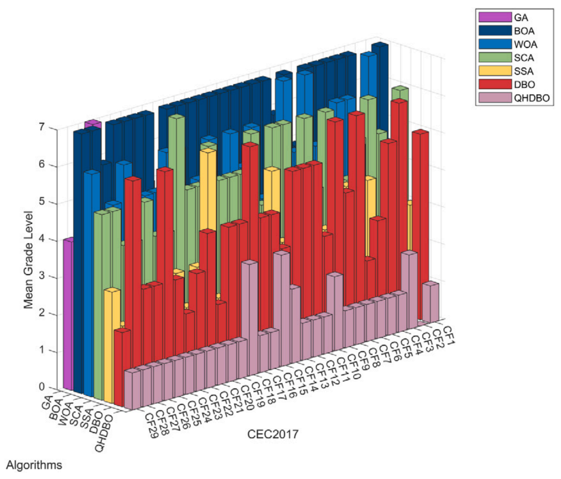

Fig. 6. CEC2017 30 dimensions mean ranking level.

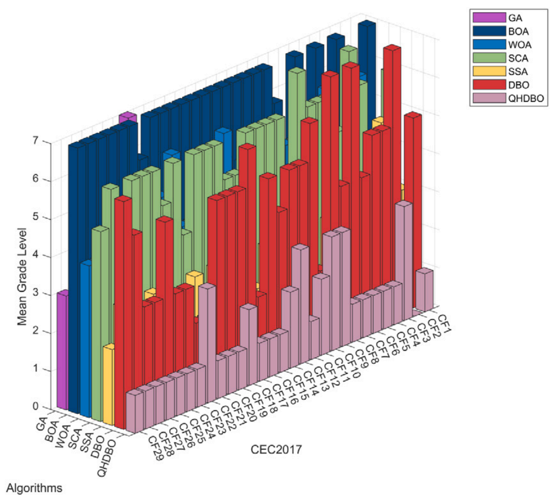

Fig. 7. CEC2017 100 dimensions mean ranking level.

GA in function F22. These findings clearly highlight the unique advantages and adaptability of QHDBO in solving compound problems, enabling more efficient exploration and optimization of complex search spaces.

4.1.2. *Comparison of CEC2017 convergence curves*

The convergence speed and accuracy of QHDBO, GA, DBO, BOA, WOA, SCA and SSA for CEC2017 are shown in Fig. 8. These curves show that QHDBO has a faster convergence rate and is less volatile and tends to be stable compared to other algorithms. This means that QHDBO is able to approach the optimal solution faster, improves the efficiency of the problem solution and has better robustness. In addition, for most of the test problems, QHDBO can achieve an accelerated convergence trend on the convergence curve, which means that the search capability of QHDBO gradually increases as the iteration progresses, allowing it to find a better solution faster. In the iterative plots of some hybrid test functions, SSA's search speeds up in the later stages, leading to its late convergence speed over QHDBO on the F13, F15 and F18 test functions, which is quickly overtaken by QHDBO in functions such as F19, although slightly in the later stages; and in functions such as F14, F15 and F18, due to reduced population diversity, crossover and variation manipulations, GA's converged significantly faster in the middle of the period while being overtaken by QHDBO in the late period. This may be due to the fact that for the reduced spawning area, making the QHDBO algorithm exhibits less ability in F15, F18 and some other functions.

From the above analysis, we can derive the following results:

Table 1
CEC2017 and CEC2020 functions.

Type	Function	Minimum value	CEC type
Unimodal functions	Shifted and Rotated Bent Cigar Function	100	CEC2017
	Shifted and Rotated Zakharov Function	200	CEC2017
	Shifted and Rotated Bent Cigar Function	100	CEC2020
Simple multimodal functions	Shifted and Rotated Rosenbrock's Function	300	CEC2017
	Shifted and Rotated Rastrigin's Function	400	CEC2017
	Shifted and Rotated Expanded Scaffer's F6 Function	500	CEC2017
	Shifted and Rotated Lunacek Bi_Rastrigin Function	600	CEC2017
	Shifted and Rotated Non-Continuous Rastrigin's Function	700	CEC2017
	Shifted and Rotated Levy Function	800	CEC2017
	Shifted and Rotated Schwefel's Function	900	CEC2017
Basic functions	Shifted and Rotated Schwefel's Function	700	CEC2020
	Shifted and Rotated Lunacek bi-Rastrigin	1900	CEC2020
	Expanded Rosenbrock's plus Griewangk's Function	1700	CEC2020
Hybrid functions	Hybrid Function 1 (N = 3)	1000	CEC2017
	Hybrid Function 2 (N = 3)	1100	CEC2017
	Hybrid Function 3 (N = 3)	1200	CEC2017
	Hybrid Function 4 (N = 4)	1300	CEC2017
	Hybrid Function 5 (N = 4)	1400	CEC2017
	Hybrid Function 6 (N = 4)	1500	CEC2017
	Hybrid Function 6 (N = 5)	1600	CEC2017
	Hybrid Function 6 (N = 5)	1700	CEC2017
	Hybrid Function 6 (N = 5)	1800	CEC2017
	Hybrid Function 6 (N = 6)	1900	CEC2017
	Hybrid Function 1 (N = 3)	1700	CEC2020
	Hybrid Function 2 (N = 4)	1600	CEC2020
	Hybrid Function 3 (N = 5)	2100	CEC2020
Composition functions	Composition Function 1 (N = 3)	2000	CEC2017
	Composition Function 2 (N = 3)	2100	CEC2017
	Composition Function 3 (N = 4)	2200	CEC2017
	Composition Function 4 (N = 4)	2300	CEC2017
	Composition Function 5 (N = 5)	2400	CEC2017
	Composition Function 6 (N = 5)	2500	CEC2017
	Composition Function 7 (N = 6)	2600	CEC2017
	Composition Function 7 (N = 6)	2700	CEC2017
	Composition Function 9 (N = 3)	2800	CEC2017
	Composition Function 10 (N = 3)	2900	CEC2017
	Composition Function 1 (N = 3)	2200	CEC2020
	Composition Function 2 (N = 4)	2400	CEC2020
	Composition Function 3 (N = 5)	2500	CEC2020
Search range: $[-100, 100]^D$			

Table 2
Parameter setting table.

Algorithm	Population size	Number of iterations	Parameters
GA	30	500	Cross = 0.2 mutation = 0.1 (Chen et al., 2020)
BOA	30	500	P = 0.8, pe = 0.1, sm = 0.01
WOA	30	500	a = 2 ∗ (1 − t/tmax), k = 1
SCA	30	500	a = 2
SSA	30	500	PD = 0.2; SD = 0.1; R2 = 0.8
DBO	30	500	RDB = 6, EDB = 6, FDB = 7, SDB = 11
QHDBO	30	500	RDB = 6, EFDBO = 13, SDB = 11

(a) QHDBO achieves the best results on a number of statistical values, including mean, standard deviation and convergence speed. This shows that QHDBO is more than capable of handling a wide range of performance metrics, demonstrating its overall powerful optimization capabilities.
(b) QHDBO shows better performance in comparison with other algorithms. It achieves optimal values in several functions, showing versatility and adaptability across different problem domains and complexities.
(c) By comparing it with other algorithms, we can see that QHDBO is better in terms of its global exploration and exploitation capabilities. It is more likely to find and converge on the optimal global solution, which indicates its stronger global search capability and ability to solve complex problems.

In summary, the results of the comparison with the other six algorithms demonstrate the significant advantages of QHDBO in terms of performance and superiority, and validate its excellent performance on different types of problems.

4.2. Results and analysis of the CEC2020 benchmark function

The CEC2020 benchmark test contains 10 single objective optimization functions and for CEC2020, the dimension chosen is 20. In this experiment, we have chosen GA, DBO, BOA, WOA, SCA and SSA as the algorithms for comparison. Since the F6 function is extremely easy to obtain its minimum value, the mean and standard deviation of the results of all algorithms are shown as a straight line,i.e. the optimal value, so we only analyse the experimental results for the remaining functions.

4.2.1. Analysis of CEC2020 statistical results

Compared to CEC2017, CEC2020 is more complex in terms of difficulty. Based on the statistical values of all algorithms run on

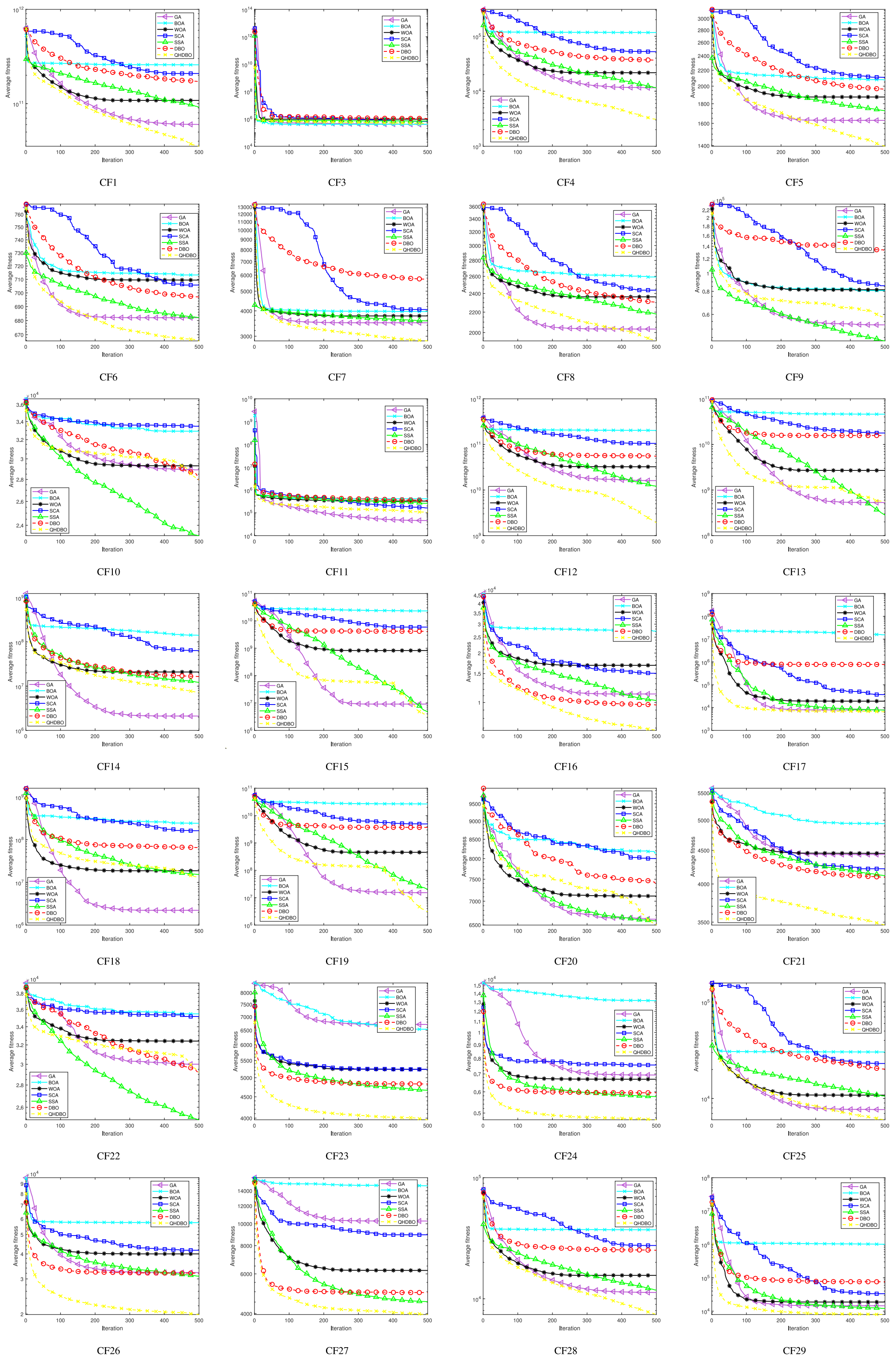

Fig. 8. CEC2017 iteration curve chart.

Table 3
CEC2017 test results 30-dimension.

		D = 30						
		GA	BOA	WOA	SCA	SSA	DBO	QHDBO
CF1	mean	2.53E+08	4.83E+10	5.16E+09	1.98E+10	4.79E+08	1.49E+10	**1.97E+06**
	degree	2	7	4	6	3	5	**1**
	std	2.26E+08	9.04E+09	1.80E+09	1.96E+09	2.83E+08	1.43E+10	**2.88E+06**
CF2	mean							
	degree							
	std							
CF3	mean	**2.47E+04**	8.30E+04	2.81E+05	9.02E+04	7.84E+04	2.16E+05	5.08E+04
	degree	**1**	4	7	5	3	6	2
	std	**1.67E+04**	9.44E+03	5.02E+04	1.50E+04	5.96E+03	6.52E+04	9.38E+03
CF4	mean	6.15E+02	2.26E+04	1.43E+03	2.89E+03	6.05E+02	2.42E+03	**5.15E+02**
	degree	3	7	4	6	2	5	**1**
	std	9.51E+01	3.75E+03	3.55E+02	9.65E+02	3.20E+01	1.18E+03	**3.36E+01**
CF5	mean	7.19E+02	9.18E+02	8.89E+02	8.25E+02	7.91E+02	7.85E+02	**6.26E+02**
	degree	2	7	6	5	4	3	**1**
	std	3.56E+01	2.89E+01	6.61E+01	3.65E+01	1.59E+01	5.98E+01	**5.16E+01**
CF6	mean	6.59E+02	6.91E+02	6.87E+02	6.61E+02	6.64E+02	6.58E+02	**6.27E+02**
	degree	3	7	6	4	5	2	**1**
	std	7.67E+00	3.90E+00	1.39E+01	8.49E+00	6.28E+00	1.10E+01	**5.75E+00**
CF7	mean	1.17E+03	1.40E+03	1.33E+03	1.29E+03	1.32E+03	1.34E+03	**9.12E+02**
	degree	2	7	5	3	4	6	**1**
	std	8.81E+01	4.54E+01	6.36E+01	6.91E+01	2.89E+01	3.43E+02	**3.30E+01**
CF8	mean	9.63E+02	1.15E+03	1.08E+03	1.09E+03	9.91E+02	1.05E+03	**9.60E+02**
	degree	2	7	5	6	3	4	**1**
	std	1.88E+01	1.66E+01	5.50E+01	2.34E+01	2.64E+01	7.36E+01	**3.93E+01**
CF9	mean	5.16E+03	1.10E+04	1.31E+04	8.02E+03	5.88E+03	1.13E+04	**3.11E+03**
	degree	2	5	7	4	3	6	**1**
	std	1.11E+03	1.73E+03	3.69E+03	1.67E+03	5.00E+02	4.07E+03	**1.55E+03**
CF10	mean	7.16E+03	9.12E+03	7.69E+03	8.82E+03	**5.54E+03**	6.42E+03	5.98E+03
	degree	4	7	5	6	**1**	3	2
	std	1.05E+03	5.62E+02	5.96E+02	3.52E+02	**5.40E+02**	1.27E+03	4.38E+02
CF11	mean	1.35E+03	9.75E+03	1.19E+04	3.88E+03	2.17E+03	7.47E+03	**1.31E+03**
	degree	2	6	7	4	3	5	**1**
	std	4.90E+01	3.01E+03	4.37E+03	1.17E+03	6.97E+02	8.06E+03	**4.70E+01**
CF12	mean	4.22E+07	1.45E+10	3.80E+08	3.25E+09	8.09E+06	9.74E+08	**2.77E+06**
	degree	3	7	4	6	2	5	**1**
	std	6.39E+07	2.35E+09	1.80E+08	9.78E+08	4.97E+06	1.31E+09	**1.70E+06**
CF13	mean	5.42E+04	1.29E+10	1.00E+07	1.08E+09	5.73E+04	6.12E+08	**4.14E+04**
	degree	2	7	4	6	3	5	**1**
	std	2.83E+04	4.29E+09	1.30E+07	7.46E+08	4.42E+04	1.12E+09	**2.94E+04**
CF14	mean	**2.23E+04**	3.44E+06	3.14E+06	4.82E+05	1.06E+06	3.22E+05	5.87E+04
	degree	**1**	7	6	4	5	3	2
	std	**4.19E+04**	2.71E+06	3.32E+06	3.77E+05	7.42E+05	5.50E+05	3.88E+04
CF15	mean	1.04E+04	6.04E+08	6.19E+06	3.06E+07	**4.83E+03**	6.97E+04	2.95E+04
	degree	2	7	5	6	**1**	4	3
	std	3.33E+03	5.73E+08	1.29E+07	2.64E+07	**2.55E+03**	3.64E+04	1.74E+04
CF16	mean	3.06E+03	7.78E+03	4.62E+03	3.98E+03	3.21E+03	3.33E+03	**2.82E+03**
	degree	2	7	6	5	3	4	**1**
	std	3.69E+02	2.05E+03	8.11E+02	2.77E+02	4.13E+02	5.37E+02	**3.13E+02**
CF17	mean	2.58E+03	1.43E+04	2.70E+03	2.75E+03	2.69E+03	2.79E+03	**2.21E+03**
	degree	2	7	4	5	3	6	**1**
	std	1.12E+02	1.01E+04	2.71E+02	1.62E+02	2.77E+02	4.21E+02	**2.12E+02**
CF18	mean	**1.87E+05**	5.30E+07	1.28E+07	8.59E+06	1.95E+06	4.69E+06	2.88E+06
	degree	**1**	7	6	5	2	4	3
	std	**9.81E+04**	2.80E+07	1.01E+07	4.86E+06	1.65E+06	4.60E+06	1.97E+06
CF19	mean	3.90E+04	1.19E+09	3.79E+07	1.05E+08	1.41E+04	1.01E+07	**1.00E+04**
	degree	3	7	5	6	2	4	**1**
	std	4.66E+04	1.17E+09	3.43E+07	6.95E+07	1.82E+04	2.36E+07	**1.66E+04**
CF20	mean	2.82E+03	3.05E+03	2.82E+03	2.85E+03	2.85E+03	2.80E+03	**2.47E+03**
	degree	4	7	3	5	6	2	**1**
	std	2.76E+02	1.58E+02	3.52E+02	1.78E+02	2.89E+02	2.49E+02	**2.21E+02**
CF21	mean	2.52E+03	2.72E+03	2.63E+03	2.61E+03	2.55E+03	2.56E+03	**2.43E+03**
	degree	2	7	6	5	3	4	**1**
	std	4.24E+01	6.82E+01	3.85E+01	1.99E+01	7.48E+01	3.99E+01	**4.35E+01**

(*continued on next page*)

Table 3 (*continued*).

CF22	mean	7.51E+03	7.49E+03	8.52E+03	1.04E+04	7.07E+03	7.09E+03	**4.18E+03**
	degree	5	4	6	7	2	3	**1**
	std	1.17E+03	9.09E+02	9.04E+02	3.14E+02	1.71E+03	2.10E+03	**2.45E+03**
CF23	mean	3.54E+03	3.56E+03	3.15E+03	3.09E+03	3.05E+03	3.03E+03	**2.82E+03**
	degree	6	7	5	4	3	2	**1**
	std	2.11E+02	1.96E+02	1.19E+02	2.44E+01	8.30E+01	9.50E+01	**4.21E+01**
CF24	mean	3.56E+03	4.03E+03	3.34E+03	3.27E+03	3.15E+03	3.18E+03	**3.00E+03**
	degree	6	7	5	4	2	3	**1**
	std	9.76E+01	2.59E+02	6.33E+01	4.25E+01	8.72E+01	7.32E+01	**4.14E+01**
CF25	mean	2.99E+03	5.86E+03	3.19E+03	3.57E+03	2.99E+03	3.84E+03	**2.92E+03**
	degree	3	7	4	5	2	6	**1**
	std	3.89E+01	4.98E+02	7.13E+01	2.06E+02	4.47E+01	1.12E+03	**2.87E+01**
CF26	mean	8.28E+03	1.17E+04	8.69E+03	7.80E+03	6.40E+03	7.54E+03	**5.20E+03**
	degree	5	7	6	4	2	3	**1**
	std	4.46E+02	4.83E+02	8.83E+02	1.91E+02	2.02E+03	6.89E+02	**3.54E+02**
CF27	mean	4.52E+03	4.46E+03	3.53E+03	3.51E+03	3.32E+03	3.33E+03	**3.27E+03**
	degree	7	6	5	4	2	3	**1**
	std	3.37E+02	3.24E+02	1.66E+02	5.77E+01	9.19E+01	4.76E+01	**3.51E+01**
CF28	mean	3.44E+03	7.84E+03	3.86E+03	4.57E+03	3.37E+03	5.69E+03	**3.27E+03**
	degree	3	7	4	5	2	6	**1**
	std	9.14E+01	5.35E+02	2.37E+02	3.93E+02	6.13E+01	1.46E+03	**3.76E+01**
CF29	mean	5.03E+03	1.62E+04	5.55E+03	5.16E+03	4.52E+03	4.38E+03	**3.92E+03**
	degree	4	7	6	5	3	2	**1**
	std	2.59E+02	1.56E+04	7.03E+02	3.36E+02	1.83E+02	2.38E+02	**1.89E+02**

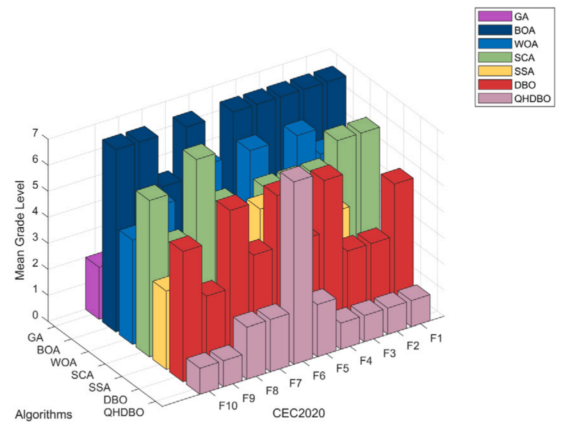

Fig. 9. CEC2020 20 dimensions mean ranking level.

CEC2020 shown in Table 5 and the average ranking of the solution results of QHDBO and its comparison algorithms after solving each function in CEC2020 for one hundred independent runs in Fig. 9. It can be concluded that QHDBO achieves the best results on six of these functions. These results indicate that QHDBO is able to find better solutions for these functions and that the stability of the solutions is high.

It is also important to note that on functions F5 and F7, GA performs better than QHDBO. Whereas on the functions F4 and F10, GA and SSA were able to match QHDBO and outperform the algorithm in terms of performance. However, in the other functions tested, QHDBO outperformed the other comparative algorithms. These statistics further validate that QHDBO has better global search and optimization capabilities in most of the CEC2020 functions. It was able to find excellent solutions in these functions and performed well in terms of solution quality and stability.

4.2.2. *Comparison of CEC2020 convergence curves*

Based on the data shown in Fig. 10, we can compare the convergence speed and accuracy of different algorithms (GA, QHDBO, DBO, BOA, WOA, SCA and SSA) in CEC2020. It can be observed from the graph that GA converges faster than QHDBO in the mid-stage of function F1. This means that GA may exhibit faster convergence in the mid-stage of function F1. However,this advantage does not last long, and soon QHDBO overtakes GA and maintains a faster convergence rate in the subsequent optimization process. In function F4 and function F10, although QHDBO excels in terms of early convergence speed, other algorithms such as GA, SSA and WOA gradually catch up and achieve results similar to QHDBO through their strengths in exploration capabilities, local search capabilities and parameter tuning. The reason why GA performs best in F5 and F7 and SSA performs best in F8 is that the characteristics of the problem differ from the algorithm's match, with F5 and F7 better suited to the characteristics of the genetic algorithm, while F8 is better suited to the search mechanism of the SSA algorithm. In addition, among other functions, QHDBO showed fast convergence and strong stability with the best results. This means that QHDBO is able to converge to better results in a fast and stable manner in most of the functions of CEC2020.

In conclusion, QHDBO demonstrates fast convergence speed and high stability on most functions in CEC2020 compared to other algorithms (GA, DBO, BOA, WOA, SCA and SSA). This provides strong support for the usefulness and effectiveness of QHDBO in solving complex optimization problems.

4.3. *Wilcoxon rank sum test*

The Wilcoxon rank sum test (Derrac et al., 2011) is a nonparametric statistical test used to determine whether the comparative QHDBO algorithm is significantly different from other algorithms. Therefore, the results of 100 independent tests of each of the six algorithms on 37 test functions were used as samples. The Wilcoxon rank sum test was performed at the significance level of 0.05 to determine the significant difference between the solution results of the six compared algorithms and the solution results of QHDBO, and the test results are shown in Tables 6, 7, 8.

When $P < 0.05$ can be considered as rejecting the null hypothesis, it means that the two algorithms compared are significantly different, and $P > 0.05$ means that the two algorithms have comparable search results. From Tables 3, 4, 5, it can be seen that the QHDBO algorithm is significantly different from the other algorithms. In summary, QHDBO has a significant advantage over GA,DBO, SSA, SCA, WOA, and BOA, so the superiority of the QHDBO algorithm is statistically significant.

Table 4
CEC2017 test results 100-dimension.

		D = 100						
		GA	BOA	WOA	SCA	SSA	DBO	QHDBO
	mean	6.02E+10	2.58E+11	1.08E+11	2.09E+11	9.16E+10	1.72E+11	**3.54E+10**
CF1	degree	2	7	4	6	3	5	**1**
	std	1.02E+10	1.32E+10	1.32E+10	1.28E+10	1.21E+10	5.82E+10	**8.09E+09**
	mean							
CF2	degree							
	std							
	mean	**3.93E+05**	4.00E+05	9.41E+05	6.43E+05	6.97E+05	1.04E+06	4.53E+05
CF3	degree	**1**	2	6	4	5	7	3
	std	**7.45E+04**	2.90E+04	1.59E+05	8.12E+04	1.51E+05	2.19E+05	6.83E+04
	mean	1.15E+04	1.17E+05	2.17E+04	5.25E+04	1.16E+04	3.67E+04	**3.07E+03**
CF4	degree	2	7	4	6	3	5	**1**
	std	2.60E+03	1.09E+04	5.19E+03	6.60E+03	2.83E+03	2.44E+04	**6.88E+02**
	mean	1.63E+03	2.08E+03	1.87E+03	2.11E+03	1.73E+03	1.96E+03	**1.40E+03**
CF5	degree	2	6	4	7	3	5	**1**
	std	6.67E+01	5.52E+01	7.23E+01	4.03E+01	7.71E+01	1.78E+02	**1.15E+02**
	mean	6.82E+02	7.13E+02	7.09E+02	7.06E+02	6.82E+02	6.97E+02	**6.66E+02**
CF6	degree	2	7	6	5	3	4	**1**
	std	7.02E+00	4.77E+00	1.10E+01	5.09E+00	3.71E+00	8.52E+00	**8.97E+00**
	mean	3.49E+03	3.97E+03	3.78E+03	4.06E+03	3.57E+03	5.72E+03	**2.84E+03**
CF7	degree	2	5	4	6	3	7	**1**
	std	1.77E+02	7.18E+01	1.13E+02	2.43E+02	7.26E+01	2.10E+03	**1.68E+02**
	mean	2.03E+03	2.59E+03	2.36E+03	2.44E+03	2.18E+03	2.30E+03	**1.92E+03**
CF8	degree	2	7	5	6	3	4	**1**
	std	7.13E+01	4.06E+01	4.69E+01	5.46E+01	5.36E+01	1.71E+02	**2.27E+02**
	mean	5.27E+04	8.07E+04	8.15E+04	8.58E+04	**4.32E+04**	1.33E+05	5.79E+04
CF9	degree	2	4	5	6	**1**	7	3
	std	7.46E+03	5.32E+03	1.93E+04	6.96E+03	**2.85E+03**	4.39E+04	7.63E+03
	mean	2.89E+04	3.29E+04	2.93E+04	3.35E+04	**2.32E+04**	2.80E+04	2.80E+04
CF10	degree	4	6	5	7	**1**	2	3
	std	2.34E+03	5.72E+02	1.74E+03	3.35E+02	**1.34E+03**	5.25E+03	1.33E+03
	mean	**4.60E+04**	4.15E+05	3.27E+05	1.64E+05	2.60E+05	3.36E+05	1.07E+05
CF11	degree	**1**	7	5	3	4	6	2
	std	**1.41E+04**	2.42E+05	1.06E+05	4.58E+04	6.58E+04	6.27E+04	1.23E+04
	mean	1.61E+10	2.03E+11	3.24E+10	1.05E+11	1.19E+10	5.53E+10	**1.95E+09**
CF12	degree	3	7	4	6	2	5	**1**
	std	7.66E+09	7.48E+09	6.81E+09	1.18E+10	4.71E+09	1.35E+10	**9.44E+08**
	mean	5.33E+08	4.60E+10	2.69E+09	1.78E+10	**2.79E+08**	1.57E+10	5.63E+08
CF13	degree	2	7	4	6	**1**	5	3
	std	3.16E+08	2.66E+09	9.21E+08	2.97E+09	**1.36E+08**	9.50E+09	1.73E+09
	mean	**2.11E+06**	1.41E+08	2.09E+07	6.39E+07	1.21E+07	1.65E+07	7.32E+06
CF14	degree	**1**	7	5	6	3	4	2
	std	**1.47E+06**	7.25E+07	1.26E+07	3.17E+07	7.00E+06	1.39E+07	3.23E+06
	mean	9.14E+06	2.31E+10	8.26E+08	5.94E+09	4.69E+06	4.16E+09	**3.36E+06**
CF15	degree	3	7	4	6	2	5	**1**
	std	7.10E+06	4.47E+09	8.13E+08	1.98E+09	2.78E+06	2.96E+09	**7.42E+06**
	mean	1.13E+04	2.72E+04	1.68E+04	1.51E+04	1.03E+04	9.65E+03	**6.77E+03**
CF16	degree	4	7	6	5	3	2	**1**
	std	1.39E+03	2.18E+03	2.22E+03	1.07E+03	1.09E+03	1.81E+03	**7.72E+02**
	mean	7.90E+03	1.54E+07	1.91E+04	3.70E+04	7.72E+03	7.76E+05	**6.54E+03**
CF17	degree	3	7	4	5	2	6	**1**
	std	1.71E+03	1.14E+07	1.69E+04	3.61E+04	5.65E+02	2.03E+06	**5.51E+02**
	mean	**2.22E+06**	2.44E+08	1.88E+07	1.63E+08	1.53E+07	6.56E+07	1.33E+07
CF18	degree	**1**	7	4	6	3	5	2
	std	**9.59E+05**	9.01E+07	6.88E+06	9.93E+07	7.23E+06	6.35E+07	8.76E+06
	mean	1.53E+07	2.67E+10	4.56E+08	4.96E+09	2.07E+07	3.70E+09	**3.05E+06**
CF19	degree	2	7	4	6	3	5	**1**
	std	1.25E+07	5.76E+09	2.16E+08	1.50E+09	1.97E+07	4.40E+09	**1.93E+06**
	mean	6.61E+03	8.14E+03	7.11E+03	8.00E+03	6.58E+03	7.38E+03	**6.50E+03**
CF20	degree	3	7	4	6	2	5	**1**
	std	1.09E+03	3.34E+02	5.47E+02	2.54E+02	4.89E+02	8.37E+02	**5.79E+02**
	mean	4.44E+03	4.94E+03	4.45E+03	4.22E+03	4.12E+03	4.09E+03	**3.46E+03**
CF21	degree	5	7	6	4	3	2	**1**
	std	1.44E+02	1.30E+02	2.23E+02	9.20E+01	3.25E+02	2.15E+02	**9.79E+01**
	mean	3.00E+04	3.55E+04	3.24E+04	3.52E+04	**2.49E+04**	2.92E+04	2.92E+04
CF22	degree	4	7	5	6	**1**	2	3
	std	3.63E+03	7.47E+02	1.47E+03	7.44E+02	**1.69E+03**	5.10E+03	2.16E+03

(*continued on next page*)

Table 4 (*continued*).

	mean	6.71E+03	6.53E+03	5.24E+03	5.24E+03	4.66E+03	4.83E+03	**3.96E+03**
CF23	degree	7	6	4	5	2	3	**1**
	std	6.41E+02	2.94E+02	2.38E+02	1.99E+02	2.19E+02	1.61E+02	**1.89E+02**
	mean	6.94E+03	1.31E+04	6.69E+03	7.56E+03	5.76E+03	5.95E+03	**4.72E+03**
CF24	degree	5	7	4	6	2	3	**1**
	std	2.94E+02	1.38E+03	4.01E+02	2.97E+02	4.01E+02	1.98E+02	**2.57E+02**
	mean	7.69E+03	3.04E+04	1.08E+04	2.31E+04	1.05E+04	2.02E+04	**6.02E+03**
CF25	degree	2	7	4	6	3	5	**1**
	std	1.01E+03	2.06E+03	1.62E+03	2.60E+03	1.06E+03	7.71E+03	**9.21E+02**
	mean	3.23E+04	5.74E+04	4.00E+04	4.17E+04	3.09E+04	3.21E+04	**1.99E+04**
CF26	degree	4	7	5	6	2	3	**1**
	std	3.72E+03	2.97E+03	3.25E+03	3.40E+03	2.13E+03	3.92E+03	**1.47E+03**
	mean	1.03E+04	1.47E+04	6.21E+03	8.94E+03	4.52E+03	4.96E+03	**3.96E+03**
CF27	degree	6	7	4	5	2	3	**1**
	std	2.31E+03	1.69E+03	8.19E+02	7.11E+02	3.72E+02	5.64E+02	**2.55E+02**
	mean	1.13E+04	3.71E+04	1.56E+04	2.76E+04	1.18E+04	2.52E+04	**7.42E+03**
CF28	degree	2	7	4	6	3	5	**1**
	std	1.15E+03	1.39E+03	1.44E+03	3.03E+03	1.44E+03	5.14E+03	**1.07E+03**
	mean	1.44E+04	1.01E+06	1.87E+04	3.32E+04	1.20E+04	7.56E+04	**7.91E+03**
CF29	degree	3	7	4	5	2	6	**1**
	std	1.26E+03	6.68E+05	3.38E+03	1.08E+04	1.67E+03	1.25E+05	**9.04E+02**

Table 5
CEC2020 test results 20-dimension.

		D = 20						
		GA	BOA	WOA	SCA	SSA	DBO	QHDBO
	mean	1.43E+07	2.82E+10	1.44E+09	8.59E+09	6.06E+06	5.06E+09	**3.07E+04**
F1	degree	3	7	4	6	2	5	**1**
	std	9.89E+06	3.96E+09	1.02E+09	1.67E+09	5.80E+06	5.10E+09	**7.30E+04**
	mean	3.91E+03	5.67E+03	4.39E+03	5.37E+03	3.39E+03	3.91E+03	**2.86E+03**
F2	degree	4	7	5	6	2	3	**1**
	std	7.39E+02	2.49E+02	4.98E+02	2.67E+02	4.23E+02	4.41E+02	**3.33E+02**
	mean	8.89E+02	1.01E+03	9.82E+02	9.41E+02	9.34E+02	9.18E+02	**8.01E+02**
F3	degree	2	7	6	5	4	3	**1**
	std	5.74E+01	1.57E+01	3.12E+01	1.76E+01	3.39E+01	1.06E+02	**3.32E+01**
	mean	1.94E+03	9.28E+05	3.03E+03	5.92E+03	1.92E+03	2.55E+04	**1.91E+03**
F4	degree	3	7	4	5	2	6	**1**
	std	2.16E+01	6.27E+05	2.42E+03	2.77E+03	6.03E+00	3.46E+04	**2.52E+00**
	mean	**1.07E+05**	6.91E+06	2.63E+06	2.52E+06	1.13E+06	1.91E+06	3.56E+05
F5	degree	**1**	7	6	5	3	4	2
	std	**1.21E+05**	3.03E+06	1.50E+06	1.86E+06	7.49E+05	1.51E+06	2.71E+05
	mean	**1.61E+03**	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03	1.61E+03
F6	degree	**1**	2	3	4	5	6	7
	std	**2.40E−13**	2.40E−13	2.40E−13	2.40E−13	2.40E−13	2.40E−13	2.40E−13
	mean	**3.97E+04**	4.27E+06	2.04E+06	7.71E+05	3.39E+05	3.83E+05	2.24E+05
F7	degree	**1**	7	6	5	3	4	2
	std	**5.47E+04**	3.76E+06	1.54E+06	6.39E+05	2.48E+05	5.69E+05	2.19E+05
	mean	3.67E+03	4.22E+03	3.66E+03	4.56E+03	**2.99E+03**	4.55E+03	3.49E+03
F8	degree	4	5	3	7	**1**	6	2
	std	1.76E+03	7.43E+02	1.82E+03	1.84E+03	**1.43E+03**	1.27E+03	1.35E+03
	mean	3.16E+03	3.61E+03	3.04E+03	3.03E+03	2.98E+03	2.99E+03	**2.90E+03**
F9	degree	6	7	5	4	2	3	**1**
	std	9.57E+01	2.27E+02	1.22E+02	2.30E+01	8.32E+01	5.49E+01	**3.44E+01**
	mean	2.98E+03	6.29E+03	3.15E+03	3.34E+03	3.00E+03	3.21E+03	**2.97E+03**
F10	degree	2	7	4	6	3	5	**1**
	std	3.26E+01	8.75E+02	6.99E+01	1.84E+02	2.07E+01	5.56E+02	**3.23E+01**

4.4. *Friedman test*

The Friedman test was used to calculate the ranking of each optimization algorithm and the test results are displayed in Table 9. From the data in Table 9, the average ranking of the improved dung beetle algorithm is lower than the other algorithms, so the effectiveness of the improvement can be proved.

5. Engineering optimization issues

To verify the performance of the QHDBO algorithm, in this section, the QHDBO algorithm is further investigated using three engineering problems: the sensor coverage problem (Yao et al., 2022), the three-rod truss design problem (Singh & Bansal, 2022), and the cantilever beam design problem (Bayzidi, 2021).

5.1. *Sensor coverage issues*

In this paper, we mainly use the following approach to calculate the coverage of sensors in a two-dimensional space(Boolean measurement model). Assume that there are N sensors $\left(p_1, p_2, \ldots, p_n\right)$ randomly distributed inside a two-dimensional space, and the coordinates of each sensor are $\{\left(x_i, y_i\right), (1 \leq i \leq N)\}$. Whether a point in the two-dimensional region is covered by a sensor node can be determined by

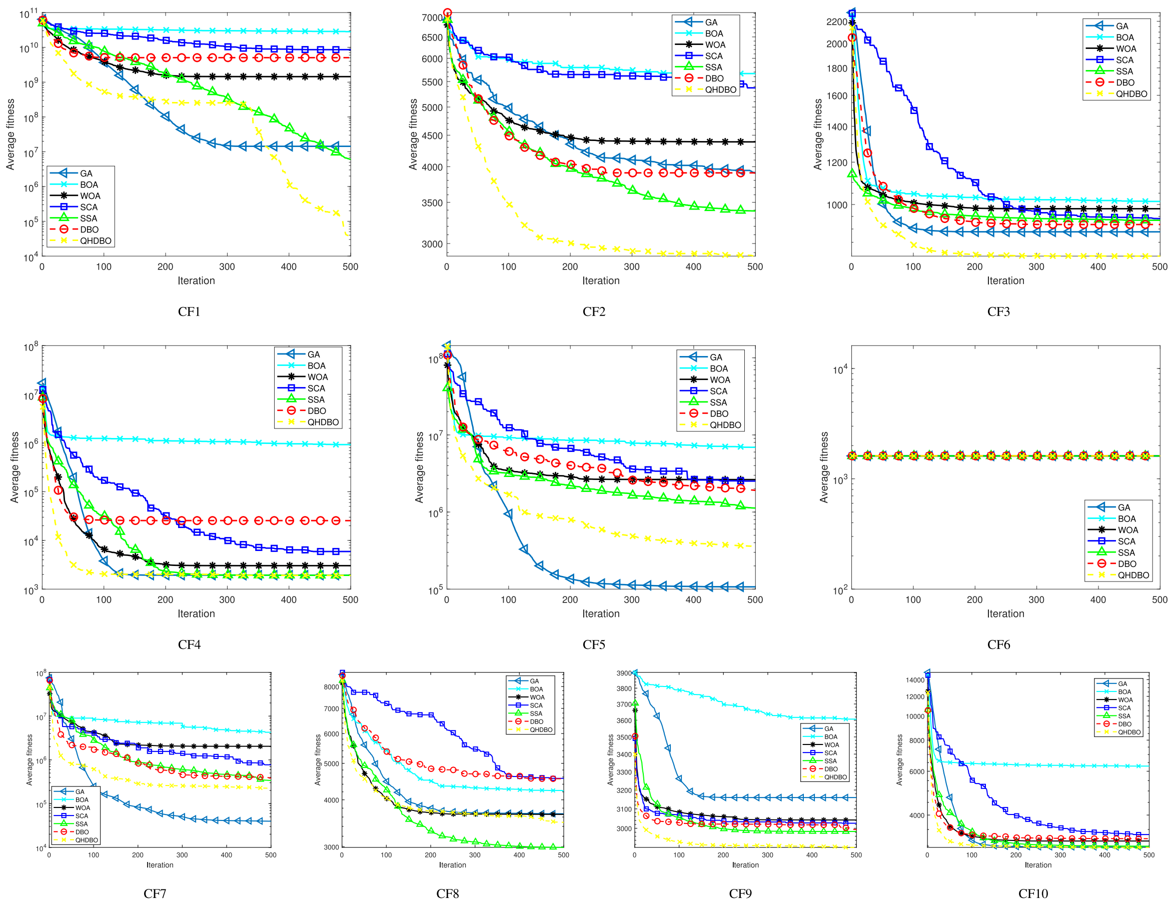

Fig. 10. CEC2020 iteration curve chart.

Table 6
CEC2017 D = 30 Wilcoxon rank sum test.

	D = 30					
	GA	BOA	WOA	SCA	SSA	DBO
CF1	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF2						
CF3	3.61E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF4	7.69E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	3.30E−04 < 0.05	1.83E−04 < 0.05
CF5	1.01E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	4.40E−04 < 0.05
CF6	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF7	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	7.69E−04 < 0.05
CF8	1.00E+00	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	8.90E−02	2.20E−03 < 0.05
CF9	3.61E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	7.69E−04 < 0.05	2.83E−03 < 0.05	2.46E−04 < 0.05
CF10	9.11E−03 < 0.05	1.83E−04 < 0.05	2.46E−04 < 0.05	1.83E−04 < 0.05	2.57E−02 < 0.05	3.85E−01
CF11	1.40E−01	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF12	2.20E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	3.61E−03 < 0.05	4.40E−04 < 0.05
CF13	2.73E−01	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	5.21E−01	1.13E−02 < 0.05
CF14	9.11E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	3.30E−04 < 0.05	1.83E−04 < 0.05	1.40E−01
CF15	1.73E−02 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	5.83E−04 < 0.05	1.13E−02 < 0.05
CF16	2.12E−01	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	3.76E−02 < 0.05	2.57E−02 < 0.05
CF17	7.69E−04 < 0.05	1.83E−04 < 0.05	1.31E−03 < 0.05	3.30E−04 < 0.05	1.01E−03 < 0.05	2.83E−03 < 0.05
CF18	1.83E−04 < 0.05	3.30E−04 < 0.05	1.71E−03 < 0.05	2.83E−03 < 0.05	2.12E−01	6.78E−01
CF19	1.40E−02 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	2.41E−01	2.46E−04 < 0.05
CF20	2.11E−02 < 0.05	2.46E−04 < 0.05	3.76E−02 < 0.05	1.01E−03 < 0.05	9.11E−03 < 0.05	1.13E−02 < 0.05
CF21	1.71E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	2.20E−03 < 0.05	2.46E−04 < 0.05
CF22	9.11E−03 < 0.05	4.59E−03 < 0.05	4.40E−04 < 0.05	1.83E−04 < 0.05	1.13E−02 < 0.05	1.73E−02 < 0.05
CF23	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF24	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	5.83E−04 < 0.05	3.30E−04 < 0.05
CF25	1.01E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.01E−03 < 0.05	1.83E−04 < 0.05
CF26	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.40E−01	1.83E−04 < 0.05
CF27	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	4.27E−01	7.28E−03 < 0.05
CF28	5.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	5.83E−04 < 0.05	1.83E−04 < 0.05
CF29	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.01E−03 < 0.05

Table 7
CEC2017 D = 100 Wilcoxon rank sum test.

	D = 100					
	GA	BOA	WOA	SCA	SSA	DBO
CF1	4.40E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF2						
CF3	4.52E−02 < 0.05	2.11E−02 < 0.05	1.83E−04 < 0.05	5.83E−04 < 0.05	4.59E−03 < 0.05	1.83E−04 < 0.05
CF4	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF5	1.31E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	3.30E−04 < 0.05	2.46E−04 < 0.05
CF6	1.31E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	4.40E−04 < 0.05	1.83E−04 < 0.05
CF7	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.71E−03 < 0.05
CF8	0.571	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	2.83E−03 < 0.05	2.20E−03 < 0.05
CF9	0.14	1.83E−04 < 0.05	2.83E−03 < 0.05	1.83E−04 < 0.05	3.30E−04 < 0.05	5.83E−04 < 0.05
CF10	0.521	1.83E−04 < 0.05	0.064	1.83E−04 < 0.05	2.46E−04 < 0.05	0.623
CF11	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	5.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF12	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF13	2.83E−03 < 0.05	1.83E−04 < 0.05	2.83E−03 < 0.05	1.83E−04 < 0.05	2.83E−03 < 0.05	2.46E−04 < 0.05
CF14	1.01E−03 < 0.05	1.83E−04 < 0.05	7.28E−03 < 0.05	1.83E−04 < 0.05	0.104	0.273
CF15	7.28E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	2.11E−02 < 0.05	1.83E−04 < 0.05
CF16	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	2.20E−03 < 0.05
CF17	1.73E−02 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.01E−03 < 0.05	1.83E−04 < 0.05
CF18	2.46E−04 < 0.05	1.83E−04 < 0.05	0.14	1.83E−04 < 0.05	0.345	1.40E−02 < 0.05
CF19	1.01E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	2.20E−03 < 0.05	1.83E−04 < 0.05
CF20	0.91	1.83E−04 < 0.05	3.12E−02 < 0.05	1.83E−04 < 0.05	0.734	1.40E−02 < 0.05
CF21	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	4.40E−04 < 0.05	4.40E−04 < 0.05
CF22	0.734	1.83E−04 < 0.05	4.59E−03 < 0.05	1.83E−04 < 0.05	1.31E−03 < 0.05	0.85
CF23	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	2.46E−04 < 0.05	1.83E−04 < 0.05
CF24	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF25	2.83E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF26	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF27	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.31E−03 < 0.05	2.20E−03 < 0.05
CF28	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	3.30E−04 < 0.05	1.83E−04 < 0.05
CF29	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	2.46E−04 < 0.05	1.83E−04 < 0.05

Table 8
CEC2020 D = 20 Wilcoxon rank sum test.

	D = 20					
	GA	BOA	WOA	SCA	SSA	DBO
CF1	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	4.40E−04 < 0.05
CF2	7.69E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	9.11E−03 < 0.05	7.69E−04 < 0.05
CF3	1.71E−03 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	2.57E−02 < 0.05
CF4	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05
CF5	3.12E−02 < 0.05	1.83E−04 < 0.05	2.46E−04 < 0.05	1.83E−04 < 0.05	3.61E−03 < 0.05	7.28E−03 < 0.05
CF6	NAN		NAN	NAN	NAN	NAN
CF7	5.80E−03 < 0.05	1.83E−04 < 0.05	1.01E−03 < 0.05	9.11E−03 < 0.05	0.162	0.678
CF8	0.427	0.273	0.307	0.241	0.97	0.064
CF9	1.83E−04 < 0.05	1.83E−04 < 0.05	1.01E−03 < 0.05	1.83E−04 < 0.05	1.40E−02 < 0.05	1.31E−03 < 0.05
CF10	0.273	1.83E−04 < 0.05	1.83E−04 < 0.05	1.83E−04 < 0.05	1.13E−02 < 0.05	0.212

Table 9
Friedman test.

Test function	Algorithm and Friedman							
CEC2017 30D	QHDBO	SSA	GA	DBO	SCA	WOA	BOA	Algorithm
	1.2500	2.8214	3.0000	4.0714	5.0000	5.2143	6.6429	Friedman
CEC2017 100D	QHDBO	SSA	GA	DBO	SCA	WOA	BOA	Algorithm
	1.4643	2.5000	2.8571	4.5000	4.5357	5.6071	6.5357	Friedman
CEC2020 20D	QHDBO	SSA	GA	DBO	WOA	SCA	BOA	Algorithm
	1.3000	2.3000	2.7000	4.0000	4.4000	5.000	6.200	Friedman

whether the distance between this point and the sensor is less than the communication radius of the sensor. Suppose the coordinates of any point in the space are $X(x, y)$ and the distance between the sensor node and this point is

$$disp\left(X, P_i\right) = \left(\sqrt{(x - x_i)^2 + (y - y_i)^2}\right) \tag{20}$$

Assuming a sensor communication radius of r, the following equation can be used to determine whether a point in the area is covered by a particular sensor:

$$c_X\left(p_i\right) = \{\begin{matrix} 1, & disp\left(X, p_i\right) < r \\ 0, & disp\left(X, p_i\right) \geq r \end{matrix} \tag{21}$$

The coverage of point p by N sensor nodes in the two-dimensional region is $c_X\left(p_1\right), c_X\left(p_2\right), \ldots, c_X\left(p_N\right)$, respectively, and the overall coverage of point X in the two-dimensional region is:

$$c_X(A) = 1 - \prod_{N}^{1}\left(1 - c_X\left(p_i\right)\right) \tag{22}$$

Point X is called the exploration point. Suppose that K exploration points are taken from the two-dimensional space, and then the probability of exploration for these explorations points are taken as the mean value, and the statistical value can be used instead of the theoretical value when K is large enough, so the overall coverage of the network

Table 10
Parameter setting.

Parameters	Takes values
Regional scope	50 m ∗ 50 m
Number of nodes	30, 35, 40
Perception radius	5 m
Communication radius	10 m

is

$$C(A) = \frac{1}{K}\sum_{i=1}^{K}\left(c_i(A)\right) \tag{23}$$

It can be seen that maximizing the coverage is solving the maximum value of Eq. (23), i.e., the coverage problem is transformed into an optimization problem. Therefore, the swarm intelligence optimization algorithm can be applied to find the optimal solution and obtain its optimal solution. The settings for the sensor parameters in this paper are shown in Table 10.

The results of each algorithm after thirty independent runs are shown in Table 11 below, from which it can be seen that the average value of the QHDBO algorithm for the optimization of sensor coverage at 30, 35, and 40 nodes are all first, indicating that the QHDBO the algorithm does not make the capability decrease due to the increase in problem complexity.

5.2. Three pole truss design issues

The problem is posed by Nowacki, which is to satisfy the stress constraint on each side of the truss member while minimizing the volume, and is mathematically described as follows:

Variables:

$$x = (x_1, x_2)$$

Objective function:

$$Minf(x) = (2\sqrt{2}x_1 + x_2) \times l \tag{24}$$

Binding Conditions:

$$g_1 = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \le 0$$
$$g_2 = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \le 0$$
$$g_3 = \frac{1}{x_1 + \sqrt{2}x_2}P - \sigma \le 0$$
$$0 \le x_1, x_2 \le 2, P = 2, l = 100, \sigma = 2$$

In order to facilitate the algorithm to solve the objective function with constraints, this paper adopts the method of penalty function to convert the objective function with constraints into the objective function without constraints, and the transformation is as follows Eq. (24):

$$Penalty\,Factor = [140, 7, 15] \tag{25}$$

$$Minf(x) = (2\sqrt{2}x_1 + x_2) \times l + \sum_{i=1}^{3} Penalty\,Factor(i) \times max(g_i, 0) \tag{26}$$

The experimental results of the six algorithms for the three-rod truss design problem are shown in Table 12, from which it can be seen that QHDBO achieves the optimal value with respect to the comparison algorithm.

5.3. Cantilever beam design issues

The problem is a structural engineering design problem with a cantilevered beam rigidly supported at one end. The beam consists of five hollow squares with fixed thicknesses. The optimization objective is to reduce the weight of the beam to a minimum. The mathematical model of the problem is:

Variables:

$$x = (x_1, x_2, x_3, x_4, x_5)$$

Objective function:

$$Minf(x) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5) \tag{27}$$

Binding Conditions:

$$g(x) = \frac{61}{x_1^3} + \frac{37}{x_2^3} + \frac{19}{x_3^3} + \frac{7}{x_4^3} + \frac{1}{x_5^3} - 1 \le 0$$
$$0.01 \le x_1, x_2, x_3, x_4, x_5 \le 100$$

The unconstrained function for this problem takes the form Eq. (27)

$$Penalty\,Factor = [10] \tag{28}$$

$$Minf(x) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5) + \sum_{i=1}^{1} Penalty\,Factor(i) \times max(g_i, 0); \tag{29}$$

From Table 13, we can see that the QHDBO algorithm achieves the best results relative to the other six compared algorithms, which again confirms that the performance of the improved dung beetle algorithm has been improved overall.

6. Conclusion

For the original dung beetle algorithm, this paper makes four main improvements: (1) Using the good point set strategy to initialize the initial population of dung beetles to make the initial population distribution more uniform, reducing the possibility of the algorithm falling into local optimal solutions. (2) Improving the Convergence factor, the convergence factor in the original algorithm is not a good balance between the global search ability of the algorithm in the early stage and the local exploration ability in the later stage, the improved convergence factor can make the algorithm pay more attention to the global search in the early stage to avoid premature maturity, and may accelerate the convergence speed in the later stage. (3) Combining spawning and foraging colonies, individuals are selected by dynamic probability to become either spawning or foraging dung beetles. Early in the algorithm, when there are more foraging dung beetles, the algorithm focuses more on global exploration, and later in the algorithm, when there are more spawning dung beetles, the algorithm focuses more on local exploration. Improvements to the spawning dung beetle location update formula apply to all situations. (4) Using a quantum computing-based t-distribution variation strategy to variate the optimal global solution, thus preventing the algorithm from falling into a locally optimal solution as much as possible.

To verify the performance of the improved dung beetle algorithm, this paper evaluates the improved dung beetle algorithm using the CEC2017 test function and the CEC2020 test function. From the evaluation results, the improved dung beetle algorithm increases the global search capability in the early stage so as to avoid falling into prematureness and also has a better iteration speed in the later stage i.e. the local exploration capability is enhanced. Thus, the effectiveness of the improvement is demonstrated. For the engineering problems, two engineering problems (three-rod truss design problem, three-rod truss design problem), which are often used to evaluate the capability of swarm intelligence optimization algorithms, were selected for wireless sensor coverage and the improved dung beetle algorithm has good engineering application capability from the experimental results.

Table 11
Sensor coverage.

		GA	BOA	WOA	SCA	SSA	DBO	QHDBO
30 nodes	mean	0.7163	0.69154	0.73545	0.011414	0.77224	0.77686	**0.80296**
	std	0.031689	0.011234	0.011817	0.65575	0.019154	0.016723	0.012448
35Nodes	mean	0.77374	0.73856	0.79085	0.71353	0.82814	0.84691	**0.86363**
	std	0.037064	0.011392	0.015637	0.009789	0.019971	0.02237	0.017232
40Nodes	mean	0.81084	0.77701	0.84802	0.75459	0.88128	0.88831	**0.913**
	std	0.033499	0.005313	0.020123	0.012006	0.018161	0.019549	0.021855

Table 12
Three pole truss design issues.

Algorithm	Most advantageous position	Best value
GA	X = (0.78726,0.40817)	263.9214
BOA	X = (0.76705,0.45301)	264.2257
WOA	X = (0.76578,0.4768)	264.3125
SCA	X = (0.7704,0.46517)	264.4205
SSA	X = (0.76273,0.47885)	264.3961
DBO	X = (0.77599,0.44236)	264.0209
QHDBO	X = (0.78796,0.41018)	**263.8968**

Table 13
Cantilever beam design issues.

Algorithm	Most advantageous position	Best value
GA	X = (10.0781,7.97346,9.53573,14.1029,19.6587)	3.8282
BOA	X = (6.8397,5.2197,5.4946,2.8017,3.5693)	1.4929
WOA	X = (5.7564,16.7699,3.08624,9.9781,3.89346)	2.4638
SCA	X = (9.18028,6.66374,20.9619,3.29882,7.32438)	2.9596
SSA	X = (5.7993,4.6321,6.2748,9.2556,6.8913)	2.05
DBO	X = (9.07309,15.7351,4.46144,2.58976,5.32152)	2.3201
QHDBO	X = (5.2543,4.736,6.6706,4.2815,2.3732)	**1.4549**

CRediT authorship contribution statement

Fang Zhu: Paper gate-keeping. **Guoshuai Li:** Thesis writing, Code writing, Innovation proposal. **Hao Tang:** Algorithm testing, Data curation. **Yingbo Li:** Algorithm testing, Data curation. **Xvmeng Lv:** Language touch-ups for the paper. **Xi Wang:** Language touch-ups for the paper.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

No data was used for the research described in the article.

Acknowledgements

This research is supported by the fund of the Hebei Provincial Higher Education Scientific Research Project (Science and Technology Research Key Project ZD2021403).

References

Arora, S. (2019). Butterfly optimization algorithm: a novel approach for global optimization. *Soft Computing*, *23*, http://dx.doi.org/10.1007/s00500-018-3102-4.

Bayzidi, H. (2021). Social network search for solving engineering optimization problems. *Computational Intelligence and Neuroscience*, http://dx.doi.org/10.1155/2021/8548639.

Chen, H., Li, W., & Yang, X. (2020). A whale optimization algorithm with chaos mechanism based on quasi-opposition for global optimization problems. *Expert Systems with Applications*, *158*, Article 113612. http://dx.doi.org/10.1016/j.eswa.2020.113612, URL: https://www.sciencedirect.com/science/article/pii/S095741742030436X.

Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3–18. http://dx.doi.org/10.1016/j.swevo.2011.02.002, URL: https://www.sciencedirect.com/science/article/pii/S2210650211000034.

Dorigo, M., Birattari, M., & Stutzle, T. (2006). Ant colony optimization. *IEEE Computational Intelligence Magazine*, *1*(4), 28–39. http://dx.doi.org/10.1109/MCI.2006.329691.

Duan, Y., & Yu, X. (2023). A collaboration-based hybrid GWO-SCA optimizer for engineering optimization problems. *Expert Systems with Applications*, *213*, Article 119017. http://dx.doi.org/10.1016/j.eswa.2022.119017, URL: https://www.sciencedirect.com/science/article/pii/S0957417422020358.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine Learning*, *3*(2), 95–99. http://dx.doi.org/10.1023/A:1022602019183.

Han, K.-H., & Kim, J.-H. (2000). Genetic quantum algorithm and its application to combinatorial optimization problem. In *Proceedings of the 2000 congress on evolutionary computation. CEC00 (Cat. No.00TH8512), Vol. 2* (pp. 1354–1360). http://dx.doi.org/10.1109/CEC.2000.870809, vol. 2.

Hua Luogeng, W. Y. (1978). *Applications of number theory to modern analysis*. Beijing: Science Press.

Jianhua, L., & Zhiheng, W. (2021). A hybrid sparrow search algorithm based on constructing similarity. *IEEE Access*, *9*, 117581–117595. http://dx.doi.org/10.1109/ACCESS.2021.3106269.

Kennedy, J., & Eberhart, R. (1995). Particle swarm optimization. *4*, In *Proceedings of ICNN'95 - International Conference on Neural Networks* (pp. 1942–1948 vol.4). http://dx.doi.org/10.1109/ICNN.1995.488968.

Liang, J., Suganthan, P., Qu, B., Gong, D., & Yue, C. (2019). Problem definitions and evaluation criteria for the CEC 2020 special session on multimodal multiobjective optimization. http://dx.doi.org/10.13140/RG.2.2.31746.02247.

Luo, X., Du, B., Gui, P., Zhang, D., & Hu, W. (2023). A hunger games search algorithm with opposition-based learning for solving multimodal medical image registration. *Neurocomputing*, *540*, Article 126204. http://dx.doi.org/10.1016/j.neucom.2023.03.065, URL: https://www.sciencedirect.com/science/article/pii/S092523122300317X.

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems*, *96*, 120–133. http://dx.doi.org/10.1016/j.knosys.2015.12.022, URL: https://www.sciencedirect.com/science/article/pii/S0950705115005043.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software*, *95*, 51–67. http://dx.doi.org/10.1016/j.advengsoft.2016.01.008, URL: https://www.sciencedirect.com/science/article/pii/S0965997816300163.

Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in Engineering Software*, *69*, 46–61. http://dx.doi.org/10.1016/j.advengsoft.2013.12.007, URL: https://www.sciencedirect.com/science/article/pii/S0965997813001853.

Seyyedabbasi, A. (2022). Sand Cat swarm optimization: a nature-inspired algorithm to solve global optimization problems. *Engineering with Computers*, http://dx.doi.org/10.1007/s00366-022-01604-x.

Shen, Y., Zhang, C., Soleimanian Gharehchopogh, F., & Mirjalili, S. (2023). An improved whale optimization algorithm based on multi-population evolution for global optimization and engineering design problems. *Expert Systems with Applications*, *215*, Article 119269. http://dx.doi.org/10.1016/j.eswa.2022.119269, URL: https://www.sciencedirect.com/science/article/pii/S0957417422022874.

Singh, S., & Bansal, J. C. (2022). Mutation-driven grey wolf optimizer with modified search mechanism. *Expert Systems with Applications*, *194*, Article 116450. http://dx.doi.org/10.1016/j.eswa.2021.116450, URL: https://www.sciencedirect.com/science/article/pii/S0957417421017346.

Sun, Y., Wang, X., Chen, Y., & Liu, Z. (2018). A modified whale optimization algorithm for large-scale global optimization problems. *Expert Systems with Applications*, *114*, 563–577. http://dx.doi.org/10.1016/j.eswa.2018.08.027, URL: https://www.sciencedirect.com/science/article/pii/S0957417418305360.

Wolpert, D., & Macready, W. (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, *1*(1), 67–82. http://dx.doi.org/10.1109/4235.585893.

Wu, R., Huang, H., Wei, J., Ma, C., Zhu, Y., Chen, Y., & Fan, Q. (2023). An improved sparrow search algorithm based on quantum computations and multi-strategy enhancement. *Expert Systems with Applications*, *215*, Article 119421. http://dx.doi.org/10.1016/j.eswa.2022.119421, URL: https://www.sciencedirect.com/science/article/pii/S095741742202440X.

Wu, G., Mallipeddi, R., & Suganthan, P. (2016). Problem definitions and evaluation criteria for the CEC 2017 competition and special session on constrained single objective real-parameter optimization.

Xiong, H., Wu, Z., Fan, H., Li, G., & Jiang, G. (2018). Quantum rotation gate in quantum-inspired evolutionary algorithm: A review, analysis and comparison study. *Swarm and Evolutionary Computation*, *42*, 43–57. http://dx.doi.org/10.1016/j.swevo.2018.02.020, URL: https://www.sciencedirect.com/science/article/pii/S2210650216303807.

Xue, J. (2023). Dung beetle optimizer: a new meta-heuristic algorithm for global optimization. *The Journal of Supercomputing*, *79*, http://dx.doi.org/10.1007/s11227-022-04959-6.

Xue, J., & Shen, B. (2020). A novel swarm intelligence optimization approach: sparrow search algorithm. *Systems Science & Control Engineering*, *8*(1), 22–34. http://dx.doi.org/10.1080/21642583.2019.1708830, arXiv:https://doi.org/10.1080/21642583.2019.1708830.

Yao, Y., Hu, S., Li, Y., & Wen, Q. (2022). A node deployment optimization algorithm of WSNs based on improved moth flame search. *IEEE Sensors Journal*, *22*(10), 10018–10030. http://dx.doi.org/10.1109/JSEN.2022.3166804.

Yildirim, G. (2022). A novel grid-based many-objective swarm intelligence approach for sentiment analysis in social media. *Neurocomputing*, *503*, 173–188. http://dx.doi.org/10.1016/j.neucom.2022.06.092, URL: https://www.sciencedirect.com/science/article/pii/S0925231222008220.

Yu, X., Jiang, N., Wang, X., & Li, M. (2023). A hybrid algorithm based on grey wolf optimizer and differential evolution for UAV path planning. *Expert Systems with Applications*, *215*, Article 119327. http://dx.doi.org/10.1016/j.eswa.2022.119327.

Zhao, W., Zhang, Z., & Wang, L. (2020). Manta ray foraging optimization: An effective bio-inspired optimizer for engineering applications. *Engineering Applications of Artificial Intelligence*, *87*, Article 103300. http://dx.doi.org/10.1016/j.engappai.2019.103300, URL: https://www.sciencedirect.com/science/article/pii/S0952197619302593.