The Journal of Supercomputing (2023) 79:7305–7336
https://doi.org/10.1007/s11227-022-04959-6

Dung beetle optimizer: a new meta-heuristic algorithm for global optimization

Jiankai Xue[1,2] · **Bo Shen**[1,2]

Accepted: 16 November 2022 / Published online: 27 November 2022

Abstract

In this paper, a novel population-based technique called dung beetle optimizer (DBO) algorithm is presented, which is inspired by the ball-rolling, dancing, foraging, stealing, and reproduction behaviors of dung beetles. The newly proposed DBO algorithm takes into account both the global exploration and the local exploitation, thereby having the characteristics of the fast convergence rate and the satisfactory solution accuracy. A series of well-known mathematical test functions (including both 23 benchmark functions and 29 CEC-BC-2017 test functions) are employed to evaluate the search capability of the DBO algorithm. From the simulation results, it is observed that the DBO algorithm presents substantially competitive performance with the state-of-the-art optimization approaches in terms of the convergence rate, solution accuracy, and stability. In addition, the Wilcoxon signed-rank test and the Friedman test are used to evaluate the experimental results of the algorithms, which proves the superiority of the DBO algorithm against other currently popular optimization techniques. In order to further illustrate the practical application potential, the DBO algorithm is successfully applied in three engineering design problems. The experimental results demonstrate that the proposed DBO algorithm can effectively deal with real-world application problems.

Keywords Convergence rate · Dung beetle optimizer (DBO) · Engineering applications · Particle swarm optimization (PSO) · Swarm intelligence

✉ Bo Shen
bo.shen@dhu.edu.cn

[1] College of Information Science and Technology, Donghua University, Shanghai, China

[2] Engineering Research Center of Digitalized Textile and Fashion Technology, Ministry of Education, Shanghai, China

1 Introduction

The optimization problems have been the focus of research for a long time and has existed in a variety of real-world systems, including fault diagnosis systems [1, 2], energy management systems [3, 4], forecasting systems [5, 6], and so on [7, 8]. It should be noticed that a large number of complex optimization problems (e.g., the NP-complete problems) are particularly difficult to be settled using the conventional mathematical programming techniques such as the conjugate gradient and quasi-Newton methods [9]. In this regard, a great number of swarm intelligence (SI) optimization algorithms have been introduced with the merits of the easy implementation, self-learning ability, and simple framework. Specifically, the SI system can be viewed as a swarm where each individual denotes a candidate solution in the entire search space. In addition, the characteristic of the SI system is that the individual interactions promote the appearance of the intelligence behavior. It is worth pointing out that the realization of the optimization process mainly includes the following two steps: 1) creating a group of random individuals within the scope of the search space and 2) combining, moving, or evolving these random individuals during the iteration process. This is also the main framework of almost all SI-based techniques in dealing with optimization problems. Note that the difference for each optimization algorithm is how to design new strategies (especially combining, moving, or evolving) during the optimization process.

For example, a well-known population-based technique, namely the particle swarm optimization (PSO) technique, has gained much research attention with the advantages of the fast convergence rate, few parameters, and satisfactory solution accuracy [10, 11]. To be specific, in a standard PSO algorithm, all particles thoroughly explore and exploit the search space of the optimization problem according to their velocity and position. Then, the movement of the whole group evolves from the disorder to the order, and eventually all particles are clustered at the best position. It should be mentioned that the position of each individual can gradually converge to the global optimal solution during the evolution process, which mainly depends on following two important positions: 1) one is the personal best position of each individual and 2) the other is the global best position of the entire population. In addition, the ant colony optimization (ACO) algorithm has become as a famous SI-based technique as the PSO approach as well [12, 13]. More specifically, in a ACO algorithm, the path of each ant indicates a feasible solution and the paths of all ants constitute the solution space. Note that the selection of the pheromone is of crucial importance for the ant colony to search the globally optimal path in the whole optimization problem space. In such a case, the shortest path is obtained in the direction of the stronger level of pheromone.

Up to now, many novel SI-based techniques have been proposed with the purpose of obtaining more optimizers with good search ability that can solve various practical optimization problems more efficiently, and their performance has been verified by a large number of experiments in the existing literature. Similarly,

these new SI-based algorithms also mainly mimic the social behaviors of living creatures (e.g., fish, insects and birds) in nature [14]. For example, the grey wolf optimizer (GWO) algorithm has been presented in [15], which simulates the leadership hierarchy (including alpha, beta, delta, and omega) and hunting behavior of grey wolves. Note that the hunting mechanism of the GWO algorithm mainly consists of searching, encircling, and attacking prey. Recently, the whale optimization algorithm (WOA) has been put forward in [16] where the bubble-net hunting strategy has been first proposed that mimics the social behavior of humpback whales. More recently, a new SI-based technique, the Harris hawks optimizer (HHO) algorithm, has been proposed in [17], which simulates the living behavior of Harris' hawks. The HHO algorithm has attracted the attention of researchers from both industrial and academic societies due to its strong capability of searching around the global optimum [18–20]. Moreover, there are several promising SI-based algorithms as well, see [21–25] for more discussions.

It is well known that a great number of SI-based algorithms have played a vitally important role in many practical applications. For example, it has been shown in [26, 27] that the ACO algorithm has the satisfactory search capability in dealing with the traveling salesman problems. Similarly, during the past few decades, the PSO algorithm has been widely implemented in solving the complex optimization problems such as large-scaled complex networks [28], fuzzy systems [29], complicated multimodal optimization systems [30] and other fields [31]. Furthermore, in the past few years, the newly developed SI-based approaches have also been successfully applied to various applications with the purpose of providing more suitable solutions for different optimization problems. For instance, in [32], by introducing the two new strategies such as the reverse learning and the Levy flight disturbance, an improved the WOA (IWOA) has been proposed and then is used to identify the unknown parameters with regard to a static var compensator (SVC) model in order to verify the optimization performance of the IWOA. In [33], a hybrid HHO algorithm, namely comprehensive learning Harris hawks-equilibrium optimization (CLHHEO) algorithm, has been designed by utilizing the comprehensive learning strategy, the operator of equilibrium optimizer, and the terminal replacement mechanism. The CLHHEO algorithm is successfully used in the optimization of several engineering design problems (e.g., the car side impact design and the multiple disc clutch brake design). The sparrow search algorithm (SSA) has been designed in [23] and a random walk SSA (RWSSA) has been proposed in [34] to optimize the distribution and signal coverage of 5G networks in open-pit mines. In summary, the emerging SI-based techniques have received much research attention in dealing with various practical applications.

On the other hand, according to No Free Lunch (NFL) theorem, we know that no single algorithm can deal with all optimization problems. In other words, the optimization performance of an algorithm may perform well in one set of problems and poorly in another. Therefore, the NFL theorem encourages finding and developing more optimizers with satisfactory performance. Motivated by the above discussions, a new SI-based technique called dung beetle optimizer (DBO) is developed with the aim of providing a more efficient optimizer to solve complex optimization problems. Correspondingly, the main contributions of this paper can be highlighted

into the following three aspects: *1) the DBO algorithm is put forward where five different updating rules that are inspired by the ball-rolling, dancing, foraging, stealing, and reproduction behaviors of dung beetles are designed to help find high-quality solutions; 2) the DBO algorithm is comprehensively evaluated through a series of mathematical test functions (including both 23 benchmark functions and 29 CEC-BC-2017 test functions) and the experimental results indicate that the DBO algorithm more competitive performance compared with the state-of-the-art optimization techniques; and 3) the DBO algorithm is successfully employed in several practical engineering design problems, which illustrates that this algorithm is promising for solving real-world application problems.*

The remainder of this paper is structured as follows. The main inspiration and biological foundations of this paper and the proposed DBO algorithm are introduced in Sect. 2. Benchmark test functions, parameter settings, optimization results and discussions are shown in Sect. 3. Section 4 introduces the applications of the DBO algorithm with regard to three engineering design problems in detail. Finally, the conclusions of on the paper are given in Sect. 5.

2 Dung beetle optimizer

In this section, a novel SI-based optimization technique called the DBO algorithm is discussed in detail, including the following two aspects: 1) the inspiration and 2) the mathematical model.

2.1 Inspiration

There are various species for the dung beetles, such as Copris ochus Motschulsky, Onthophagus gibbulus, Caccobius jessoensis Harold and so on. It is well known that, the dung beetle, as a common insect in nature, feeds on the dung of animals. Note that dung beetles are found in most parts of the world and act as decomposers in nature, which means that they are of vital importance in the ecosystem. Research has shown that dung beetles have an interesting habit of making dung into a ball and then rolling it out, as captured in Fig. 1. It is worth mentioning that the purpose of dung beetles is able

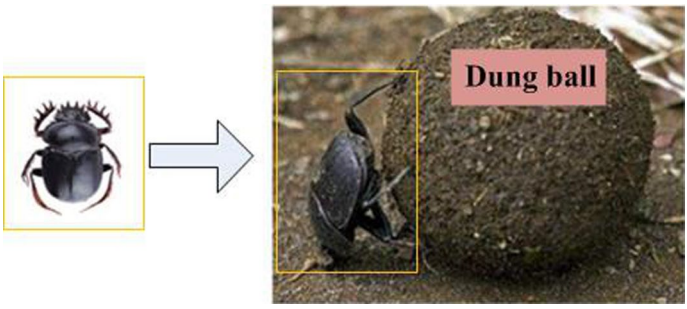


Fig. 1 The rolling dung ball behavior of the dung beetle

to move their dung ball as quickly and efficiently as possible, which can prevent them from being competed by other dung beetles [35].

As shown in Fig. 1, a dung beetle is rolling backward a dung ball that is larger than itself. On the other hand, a fascinating behavior of dung beetle can use celestial cues (especially the sun, the moon and polarized light) to navigate and make the dung ball roll along a straight line [36, 37]. However, if there is no light source at all (that is, in total darkness), the dung beetle's path is no longer straight, but curved and sometimes even slightly round [35]. Note that a number of natural factors (such as wind and uneven ground) can cause dung beetles to deviate from their original direction. In addition, the dung beetle is likely to encounter obstacles and be unable to move forward in the rolling process. In this regard, dung beetles usually climb on top of the dung ball to dance (including a series of rotations and pauses), which determines their direction of the movement [38].

Another interesting behavior observed from the dung beetle lifestyle is that dung balls obtained have the following two main purposes : 1) some dung balls are used to lay eggs and raise the next generation, and 2) the rest are used as food. Specifically, dung beetles bury the dung balls and the females lay their eggs in these dung balls. It should be noted that the dung ball not only serves as a developmental site for the larva, but also provides the larva with food that is essential for life. Therefore, dung balls play an irreplaceable role in the survival of dung beetles.

A new SI optimization algorithm, the DBO technique, is mainly inspired by the ball-rolling, dancing, foraging, stealing, and reproduction behaviors of dung beetles. In the next subsection, the behavior of the dung beetle will be modeled mathematically with the hope of developing a optimizer with the satisfactory search performance.

2.2 The structure and the algorithm

According to the above discussion, we know that dung beetles need to navigate through the celestial cues in the rolling process to keep the dung ball rolling in a straight path. To simulate the ball rolling behavior, dung beetles are required to move in a given direction throughout the entire search space. The trajectory of a dung beetle is given in Fig. 2. In this figure, we can see that a dung beetle use the sun to navigate, where the red arrow indicates the rolling direction. In this paper, we assume that the intensity of the light source also affects the dung beetle's path. During the rolling process, the position of the ball-rolling dung beetle is updated and can be expressed as

$$\begin{aligned} x_i(t+1) &= x_i(t) + \alpha \times k \times x_i(t-1) + b \times \Delta x, \\ \Delta x &= \mid x_i(t) - X^w \mid \end{aligned} \tag{1}$$

where t represents the current iteration number, $x_i(t)$ denotes the position information of the ith dung beetle at the tth iteration, $k \in (0, 0.2]$ denotes a constant value which indicates the deflection coefficient, b indicates a constant value belonging to (0, 1), α is a natural coefficient which is assigned -1 or 1, X^w indicates the global worst position, Δx is used to simulate changes of light intensity.

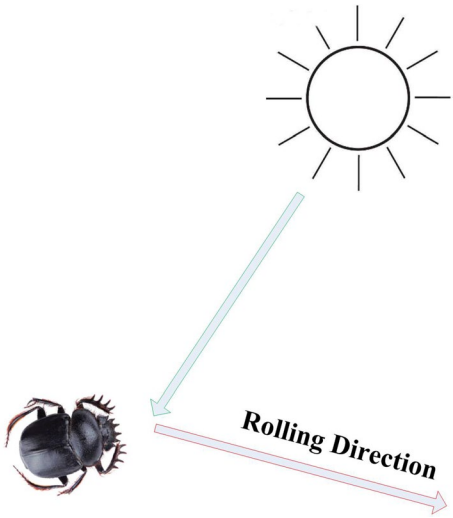


Fig. 2 Conceptual model of a dung beetle's trajectory

Remark 1 In (1), it is vitally crucial to select the appropriate values of the two parameters (k and b). Note that α means that many natural factors (such as wind and uneven ground) can deflect dung beetles from their original direction. Specifically, $\alpha = 1$ indicates no deviation, and $\alpha = -1$ indicates deviation from the original direction. In this paper, α is set to be 1 or -1 by the probability method in order to simulate the complex environment in real world (see Algorithm 1). Similarly, a higher value of Δx indicates a weaker light source. In addition, k and b are set to be 0.1 and 0.3, respectively. The Δx can promote the ball-rolling dung beetle to have the following two merits: 1) explore the whole problem space as thoroughly as possible during the optimization process; and 2) pursue stronger searching performance and reduce the possibility of falling into local optima. Hence, the X^w is more suitable control the value of the Δx to expand the search scope.

Algorithm 1 α selection strategy

Require: The probability value λ.
Ensure: The natural coefficient α.
1: $\eta = \text{rand}(1)$;
2: **if** $\eta > \lambda$ **then**
3: $\alpha = 1$;
4: **else**
5: α = -1;
6: **end if**

When the dung beetle encounters an obstacle and can't move forward, it needs to reorient itself through dancing with the purpose of obtaining a new route. Notably, the dance behavior plays an important role in ball-rolling dung beetles.

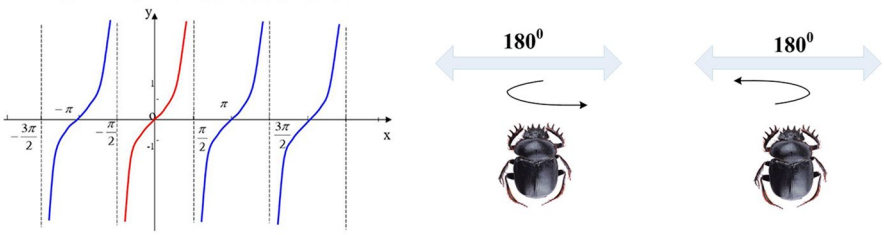


Fig. 3 Conceptual model of the tangent function and the dance behavior of a dung beetle

In order to mimic the dance behavior, we use the tangent function to get the new rolling direction. It should be mentioned that we only need to consider the values of the tangent function defined on the interval $[0, \pi]$, as shown in Fig. 3. Once the dung beetle has successfully determined a new orientation, it should continue to roll the ball backward. Therefore, the position of the ball-rolling dung beetle is updated and defined as follows:

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

where θ is the deflection angle belonging to $[0, \pi]$.

Remark 2 In (2), $\mid x_i(t) - x_i(t-1) \mid$ is the difference between the position of the ith dung beetle at the tth iteration and its position at the $t-1$th iteration. Thus, the position updating of the ball-rolling dung beetle is closely related to current and historical information. Note that, if θ equals 0, $\pi/2$ or π, the position of the dung beetle is not updated (see Algorithm 2).

Algorithm 2 θ selection strategy

Require: The deflection angle θ.
Ensure: The position of the ith dung beetle x_i.
1: **if** $\theta = 0$ or $\theta = \pi/2$ or $\theta = \pi$ **then**
2: x_i is not updated;
3: **else**
4: Update x_i according to (2);
5: **end if**

In nature, dung balls are rolled to safety and hidden by dung beetles (see Fig. 4). For the purpose of providing a safe environment for their offspring, the choice of the right place to lay their eggs is crucial for dung beetles. Inspired by the above discussions, a boundary selection strategy is proposed to simulate the areas where female dung beetles lay their eggs, which is defined by

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

where X^* denotes the current local best position, Lb^* and Ub^* mean the lower and upper bounds of the spawning area respectively, $R = 1 - t/T_{\max}$ and $T_{\max}$ indicates the maximum iteration number, Lb and Ub represent the lower and upper bounds of the optimization problem, respectively.

As shown in Fig. 5, the current local best position X^* is indicated by using a large brown circle, while the small black circles around X^* indicate the brood balls. Note that each brood ball contains an egg of the dung beetle (see Fig. 4). In addition, the small red circles represent the upper and lower bounds of the boundary.

Once the spawning area is identified, the female dung beetles choose the brood balls in this area to lay their eggs. It should be mentioned that for the DBO algorithm, each female dung beetle produces only one egg in each iteration. Furthermore, it can be clearly seen from (3) that the boundary range of the spawning area changes dynamically, which mainly determined by the R value. Thus, the position of the brood ball is also dynamic in the iteration process, which is defined by

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

where $B_i(t)$ is the position information of the ith brood ball at the tth iteration, b_1 and b_2 represent two independent random vectors by size $1 \times D$, D indicates the dimension of the optimization problem. Note that the position of the brood ball is strictly restricted to a certain range, i.e., the spawning area (see Algorithm 3).

Algorithm 3 The brood ball position updating strategy

Require: The maximum iteration number $T_{\max}$, the brood ball number n, and the current iteration number t.
Ensure: The position of the ith brood ball B_i.

```
R = 1 − t/T_max
for i ← 1 to n do
Update the brood ball's position by using (4);
    for j ← 1 to D do
      if B_ij > Ub* then
        B_ij ← Ub*;
      end if
      if B_ij < Lb* then
        B_ij ← Lb*;
      end if
    end for
end for
```

Some dung beetles that have grown into adults, emerge from the ground to find food (see Fig. 6). In this paper, we call them small dung beetles. In addition, we need to establish the optimal foraging area to guide the foraging beetles, which simulates

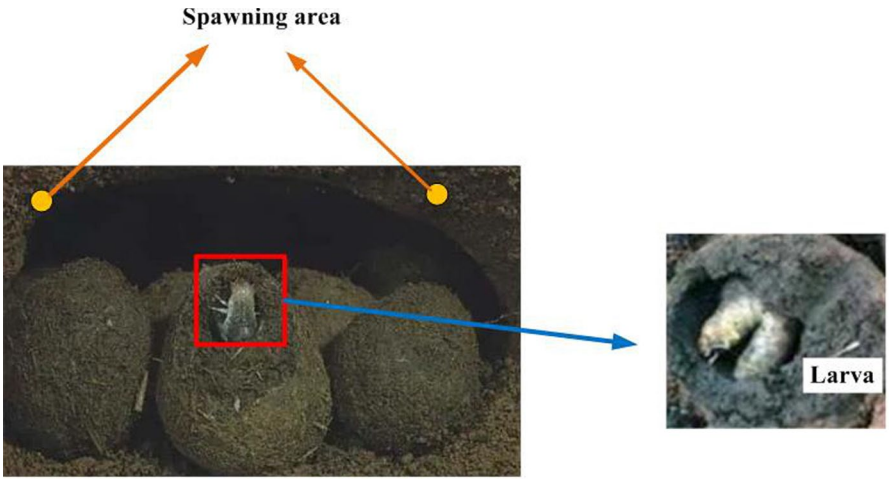


Fig. 4 The spawning area of dung beetles

the foraging process of these dung beetles in nature. Specifically, the boundary of the optimal foraging area is defined as follows:

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

where X^b denotes the global best position, Lb^b and Ub^b mean the lower and upper bounds of the optimal foraging area respectively, other parameters are defined in (3). Therefore, the position of the small dung beetle is updated as follows:

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

where $x_i(t)$ indicates the position information of the ith small dung beetle at the tth iteration, C_1 represents a random number that follows normally distributed, and C_2 denotes a random vector belonging to (0, 1).

On the other hand, some dung beetles, known as thieves, steal dung balls from other dung beetles (see Fig. 7). It should be pointed out that this is a very common phenomenon in nature. Furthermore, it can be seen from (5) that X^b is the optimal food source. Thus, we can assume that the around X^b indicates the best place to compete for food. During the iteration process, the position information of the thief is updated and can be described as follows:

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

where $x_i(t)$ denotes the position information of the ith thief at the tth iteration, and g is a random vector by size $1 \times D$ that follows normally distributed, S indicates a constant value.

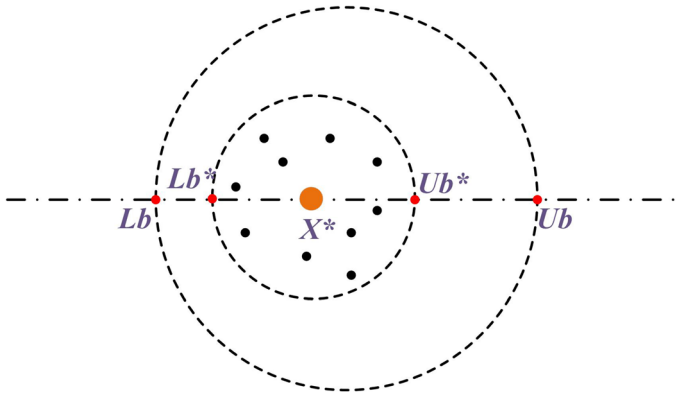


Fig. 5 Conceptual model of the boundary selection strategy

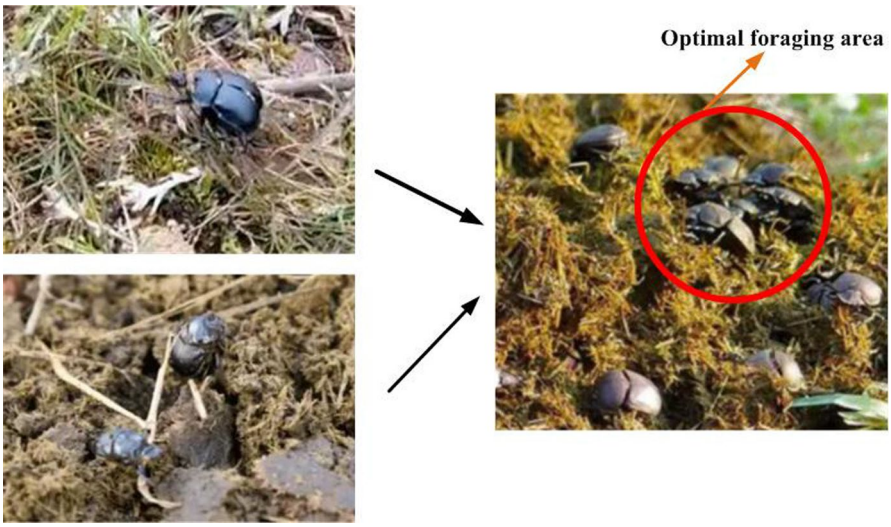


Fig. 6 The foraging process of dung beetles

Fig. 7 The stealing behavior of the dung beetle

Algorithm 4 The framework of the DBO algorithm

Require: The maximum iterations $T_{\max}$, 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 $i \leftarrow 1, 2, \ldots, N$ and define its relevant parameters
2: **while** ($t \leq T_{\max}$) **do**
3: **for** $i \leftarrow 1$ to N **do**
4: **if** $i ==$ ball-rolling dung beetle **then**
5: $\delta = \text{rand}(1)$;
6: **if** $\delta < 0.9$ **then**
7: Select α value by Algorithm 1
8: Update the ball-rolling dung beetle's position by using (1);
9: **else**
10: Update the ball-rolling dung beetle's position by using (2);
11: **end if**
12: **end if**
13: **if** $i ==$ brood ball **then**
14: Update the brood ball's position by using Algorithm 3;
15: **end if**
16: **if** $i ==$ small dung beetle **then**
17: Update the small dung beetle's position by using (6);
18: **end if**
19: **if** $i ==$ thief **then**
20: Update the position of the thief by using (7);
21: **end if**
22: **end for**
23: **if** the newly generated position is better than before **then**
24: Update it;
25: **end if**
26: $t = t + 1$;
27: **end while**
28: **return** X^b and its fitness value f_b.

Based on the previous discussions, the pseudo codes of the proposed DBO algorithm are shown in Algorithm 4. Firstly, let $T_{\max}$ be the number of maximum iteration, and N be the size of the particle's population. Then, all agents of the DBO algorithm are randomly initialized and their distribution settings are shown in Fig. 8. In this figure, the number of small rectangles denotes the population size. Specifically, as given in the Fig. 8, the population size is assumed to be 30. It is worth mentioning that the blue, yellow, green and red rectangles represent the ball-rolling dung beetle, the brood ball, the small dung beetle and the thief, respectively. After that, according to steps 2-27 of Algorithm 4, we know that the positions of the ball-rolling dung beetle, the brood ball, the small dung beetle and

the thief are constantly updated during the optimization process. Finally, the best position X^b and its fitness value are output. In summary, for the any optimization problem, the DBO algorithm, as a novel SI-based optimization technique, mainly has six steps, which can be outlined as follows: 1) initialize the dung beetle swarm and parameters of the DBO algorithm; 2) calculate the fitness values of all agents according to the objective function; 3) update the locations of the all dung beetles; 4) judge whether each agent is out of the boundary; 5) renew the current optimal solution and its fitness value; and 6) repeat the above steps till t meets the termination criterion and output the global optimal solution and its fitness value.

Remark 3 According to the developed DBO algorithm, each dung beetle swarm consists of four distinct agents, i.e., the ball-rolling dung beetle, the brood ball, the small dung beetle and the thief. More specifically, in the DBO algorithm, a dung beetle population includes N agents, where each agent i represents a candidate solution. The position vector of the ith agent is indicated by $x_i(t) = (x_{i1}(t), x_{i2}(t), \ldots, x_{iD}(t))$ at the tth iteration, where D is the dimension of the searching space. Their distribution ratio is not specified, and can be set according to the real-world application problems. For instance, as shown in Fig. 8, the size of the dung beetle's swarm is $N = 30$. The numbers of the ball-rolling dung beetle, the brood ball, the small dung beetle and the thief are 6, 6, 7, and 11, respectively. It should be noticed that the sum of their numbers should be the same as the entire population which is set to be 30.

3 Experimental results

In this section, the search performance of the developed DBO technique is verified via 23 classical test functions (including unimodal, multimodal, and fixed dimension multi modal functions), and another 29 competition functions named CEC-BC-2017 [39]. Meanwhile, the developed DBO algorithm is compared to seven well-studied optimization techniques (the HHO [17], PSO [10], WOA [16], MVO [40], SCA [41], SSA [22], and GWO [15] algorithms). It should be noticed that these algorithms cover almost all of the recently proposed techniques such as the HHO, WOA, MVO, SCA, SSA, and GWO algorithms and also, include the most classical optimization approach like the PSO algorithm. Fig. 9 demonstrates a two-dimensional shape of some test functions that this study employs to evaluate the DBO algorithm.

3.1 Experimental settings

The experiments should be conducted in the same environment with the purpose of ensuring the fairness of the experiment. Specifically, the evaluation criteria adopted in the experiments are to fairly compare the comprehensive search ability of different optimization approaches. Therefore, for the 23 classical functions, the size of

the particle's population is $N = 30$ and the number of maximum iteration is $T_{\max} = 500$ (15,000 maximum function evaluations). It should be noted that the dimension of the benchmark test functions (F1-F13) is $D = 30$. Furthermore, the dimension of the functions F14-F23 is the same as recommended in [16]. Similarly, for the CEC-BC-2017, the size of the particle's population is set to be 30 along with 10000 maximum iterations and the dimension is set to 10, which limits to 300,000 maximum function evaluations. It is worth pointing out that each simulation experiment is repeated for 30 times independently with the aim of avoiding random influence. The parameter setting for each optimization algorithm is reported in Table 1. In addition, all simulation environments are implemented under MATLAB platform with a Windows 10.

3.2 Performance indicators

Two statistical tools are used in this paper, namely, the mean value and the standard deviation (Std Dev). Their mathematical formulations are given as follows:

$$\begin{aligned} \mathrm{M} &= \frac{1}{P}\sum_{i=1}^{P} f_i, \\ \mathrm{Std} &= \sqrt{\frac{1}{P-1}\sum_{i=1}^{P}(f_i - \mathrm{M})^2} \end{aligned} \tag{8}$$

where P is the number of optimization experiments, and f_i represents the optimal value in each independent run.

3.3 DBO's exploitation capability (functions F1-F7)

It is worth mentioning that F1-F7 are typical unimodal test functions since they include only one global best solution, which has been widely used in the swarm intelligence optimization community. Note that these test functions are applied to evaluate the exploitation capability of the introduced DBO algorithm. The detailed information of simulation results is given in Table 2, where the Std Dev and the mean of the fitness value are shown to measure the search performance of the algorithms. It should be noticed that the Std Dev is presented in parentheses.

It is observable from Table 2 that, for the classical test functions F1-F4, the DBO algorithm demonstrates superiority over the other seven optimization algorithms in

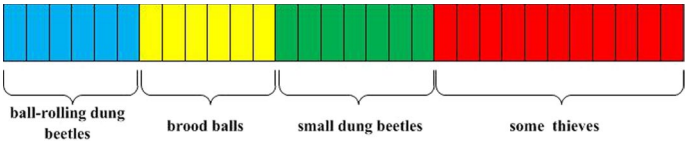


Fig. 8 The distribution of the search agent in the DBO algorithm

terms of evaluation indices including the mean and the Std Dev. Specifically, the mean fitness value of the DBO algorithm is closer to the theoretical optimum than other algorithms for F1-F4, indicating that the DBO algorithm has high exploitation ability. Moreover, although the DBO algorithm cannot obtain the best mean for F5 and F7, it still receives the second rank after HHO algorithm.

3.4 DBO's exploration capability (functions F8-F23)

Unlike these functions (F1-F7) with only one globally best solution, multimodal functions have the features of many local minima, which are hard to find the best solution in the searching space. Meanwhile, as the dimension increases, the number of local optima for multimodal functions will grow exponentially. Therefore, the functions F8-F23 (including high dimensional functions (F8-F13) and fixed-dimension test functions (F14-F23)) are very useful to verify the exploration ability of the DBO approach. The statistical results of the DBO algorithm on sixteen multimodal test functions are illustrated in Table 2.

In Table 2, for high dimensional multimodal functions F9-F11, the proposed DBO algorithm obtains better search ability than the PSO, GWO, SSA, SCA, and MVO algorithms. Moreover, it is apparent that on F9 and F11, the DBO algorithm is able to search the global optimum. For function F8, the developed DBO algorithm achieves the third rank after WOA and HHO algorithm, which demonstrates competitive result than that of the PSO, GWO, SSA, SCA, and MVO algorithms. For function F13, the HHO algorithm exhibits better optimization performance and obtains first rank, while the DBO algorithm still shows the competitive performance compared with other SI-based approaches. In fixed-dimension multimodal functions, the search capability of all algorithms is similar, and the optimization results obtained by the DBO algorithm are competitive. In functions F16-F18, the proposed algorithm can obtain the global optimum.

3.5 Sensitivity analysis of DBO's parameters

In this section, the sensitivity of three control parameters (k, b and S) employed in the DBO algorithm is studied in detail. This study is mainly used to demonstrate which parameters are robust and which ones affect the search performance of the algorithm and which ones are sensitive for different inputs. Three benchmark test functions (including the unimodal function F6, the multimodal function F9, and the fixed-dimension multimodal function F14) are used to meet a full factorial design for these control parameters. The values of parameters are expressed as $k = \{0.01, 0.15, 0.1, 0.15, 0.2\}$, $b = \{0.1, 0.3, 0.5, 0.7, 0.9\}$, and $S = \{0.1, 0.5, 1, 1.5, 2\}$. Thus, there are a total of $5 \times 5 \times 5 = 125$ combinations of design for the full factorial.

The sensitivity analysis is displayed in Fig. 10, where the horizontal coordinate indicates the mean fitness, and the vertical coordinate is the three control parameters. We can see from Fig. 10a that S demonstrates a high sensitive behavior, and S =0.5 has the best performance in F6. In addition, k and b show the similar sensitivity

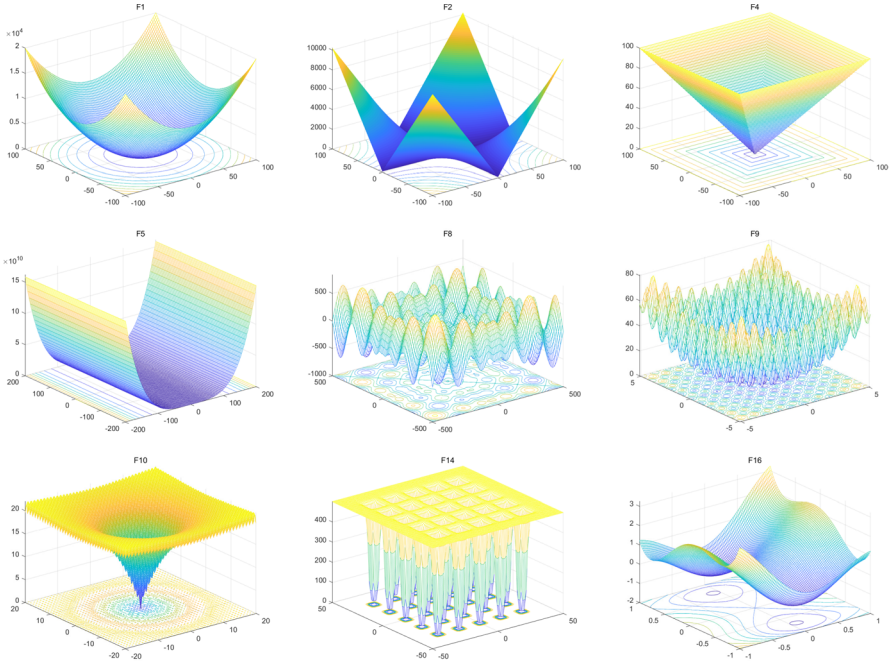


Fig. 9 2-D versions of some benchmark functions

to different inputs. Note that $k = 0.1$ and $b = 0.3$ indicate the optimal behavior. It is observable in Fig. 10b that b shows more sensitivity than k and S while S demonstrates the robust behavior to all values. Thus, $k = 0.1$ and $b = 0.3$ are the best selection for the multimodal function F9. In Fig. 10c, S indicates a sensitive behavior in its left boundary while showing a lower sensitivity after its value equals 0.5, and b demonstrates the robust behavior for different values. Moreover, we can see that $K = 0.1$ can obtain the best search performance in F14. Based on the above discussions, in this paper, $K = 0.1$, $b = 0.3$, and $S = 0.5$ are selected as the recommended parameter values for the DBO algorithm.

3.6 Analysis of convergence behavior

The convergent curves of the developed DBO algorithm and other seven optimization approaches are displayed in Fig. 11, where the horizontal coordinate represents the iteration number, and the vertical coordinate indicates the average best so far. For the functions, nine benchmark test functions (including four unimodal test functions (F1-F3 and F7), three multimodal test functions (F10-F12), and two fixed-dimension test functions (F16 and F18)) are employed to evaluate the convergence performance of the DBO algorithm.

Table 1 Parameter values of algorithms

Algorithm	Parameter	Value
PSO	Topology	Fully connected
	c_1	2
	c_2	2
	Inertia weight	Linear reduction from 0.9 to 0.1
SSA	Leader position update probability	0.5
	v_0	0
WOA	a	Decreased from 2 to 0
MVO	WEP and TDR	[0.2 1] and [0.6 0]
SCA	a	2
GWO	$a_{\min}$ and $a_{\max}$	0 and 2
HHO	Interval of E_0	[− 1 1]
DBO	k and λ	0.1
	b	0.3
	S	0.5

From the Fig. 11, we can see that the compared with other algorithms, the DBO algorithm obtains satisfactory convergence rate for unimodal functions F1-F3. This is due to the DBO algorithm that can thoroughly search for promising regions in the early part of the iteration and converge towards the global best position as quickly as possible in the later. In F7, the convergence speed of the DBO approach is better than the PSO, GWO, WOA, SSA, SCA, and MVO. By observing some convergence curves in the multimodal functions environment, it can be seen in F10-f12 that, the DBO approach is able to avoid many local optima effectively and eventually converge to the best position in the searching space. In F16 and F18, the DBO algorithm is the rapid convergence in the early part of the iteration. These results indicate that the proposed DBO approach has a higher success ratio than the state-of-the-art optimization techniques.

3.7 Statistical analysis of DBO

In order to make more rigorous comparisons, the Wilcoxon signed-rank test is employed to whether the DBO algorithm has a significant performance difference compared with other optimization approaches at a significance level of 5% confidence. Meanwhile, the Friedman test is applied to calculate the ranking of each optimization algorithm.

Table 3 provides the p-values of the Wilcoxon signed-rank test in the 23 classical test functions. Based on the p-values from Table 3, the statistics of $\}\}+''$, $\}\}-''$ and $\}\}=''$ are displayed at the bottom of Table 2. Specifically, symbol $\}\}+''$ indicates that the DBO algorithm significantly outperforms other algorithms, while symbol $\}\}-''$ means the contrary. Meanwhile, symbol $\}\}=''$ represents no significant difference,

Table 2 Results of DBO, PSO, GWO, WOA, SSA, SCA, MVO and HHO on benchmark test functions

Function	DBO	HHO	GWO	WOA	SSA	SCA	MVO	PSO
F1	2.02E−104 (1.11E−103)	4.95E−97 (2.48E−96)	1.74E−27 (4.31E−27)	2.13E−69 (1.15E−68)	3.19E−07 (5.98E−07)	9.17197 (18.41890)	1.18812 (0.33134)	1.17E−05 (3.05E−05)
F2	1.51E−55 (8.28E−55)	3.95E−51 (1.64E−50)	7.17E−17 (4.21E−17)	4.29E−51 (1.38E−50)	2.10572 (1.65985)	0.01134 (0.02551)	0.82440 (0.29227)	0.03754 (0.04520)
F3	1.37E−80 (7.48E−79)	7.91E−66 (4.33E−65)	8.39E−06 (2.85E−05)	4.40E+04 (1.89E+04)	1.46E+03 (9.86E+02)	8.40E+03 (4.65E+03)	2.10E+02 (92.82078)	77.11498 (33.63623)
F4	7.54E−56 (2.26E−55)	1.04E−47 (5.34E−47)	8.36E−07 (8.28E−07)	45.11843 (26.76003)	11.50279 (3.67431)	34.24609 (14.54381)	1.90629 (0.57681)	2.68680 (0.52359)
F5	25.73226 (0.19979)	0.01898 (0.01726)	27.07397 (0.68312)	27.98438 (0.41592)	270.697 (397.694)	5.62E+04 (9.16E+04)	337.761 (685.570)	89.21365 (68.84788)
F6	3.90E−04 (6.47E−04)	2.66E−04 (5.69E−04)	0.79625 (0.35208)	0.45753 (0.25512)	1.67E−07 (2.54E−07)	18.49275 (22.74452)	1.15520 (0.34313)	1.75E−05 (5.91E−05)
F7	0.001398 (0.001186)	1.43E−04 (1.66E−04)	0.002090 (0.001070)	0.003232 (0.003994)	0.19409 (0.09395)	0.12563 (0.14775)	0.03197 (0.01132)	0.080440 (0.02635)
F8	-8859.05 (869.71)	-12569.01 (0.73335)	-6148.35 (733.33)	-10921.29 (1722.35)	-7607.38 (640.75)	-3821.95 (238.58)	-7702.25 (453.70)	-6203.04 (1399.94)
F9	0.0000 (0.0000)	0.0000 (0.0000)	0.73963 (1.78153)	0.0000 (0.0000)	56.24825 (19.78855)	38.30915 (35.97585)	1.24E+02 (28.47116)	54.59095 (18.37504)
F10	8.88E−16 (0.0000)	8.88E−16 (0.0000)	9.91E−14 (1.55E−14)	4.32E−15 (2.71E−15)	2.55485 (0.70528)	13.15044 (8.77485)	2.34916 (3.34170)	1.21E−04 (2.15E−04)
F11	0.0000 (0.0000)	0.0000 (0.0000)	0.00207 (0.00568)	0.0000 (0.0000)	0.01919 (0.01188)	0.86621 (0.35138)	0.84749 (0.07674)	0.01051 (0.01202)
F12	5.53E−07 (3.02E−06)	1.03E−05 (1.56E−05)	0.05084 (0.02425)	0.02249 (0.01818)	6.65730 (2.35274)	1.41E+02 (6.42E+02)	2.40444 (1.64963)	0.02764 (0.04662)
F13	0.59374 (0.53477)	1.07E−04 (1.64E−04)	0.64652 (0.24045)	0.58156 (0.25261)	17.78582 (15.24761)	1.72E+04 (6.63E+04)	0.18029 (0.07573)	0.00669 (0.01138)
F14	1.06427 (0.25219)	1.29490 (0.93993)	4.29418 (4.19564)	3.61084 (3.84964)	1.13054 (0.34368)	1.65658 (1.87571)	0.99800 (3.81E−11)	3.62102 (3.10758)

Table 2 (continued)

Function	DBO	HHO	GWO	WOA	SSA	SCA	MVO	PSO
F15	6.89E−04 (3.14E−04)	3.69E−04 (1.95E−04)	0.003803 (0.007538)	0.001240 (0.002267)	0.001530 (0.003572)	0.001042 (3.37E−04)	0.004062 (0.007420)	8.83E−04 (2.79E−04)
F16	-1.03162 (5.97E−16)	-1.03162 (1.68E−09)	-1.03162 (3.03E−08)	-1.03162 (8.65E−10)	-1.03162 (3.22E−14)	-1.03158 (3.01E−05)	-1.03162 (3.79E−07)	-1.03162 (6.58E−16)
F17	0.397887 (0.0000)	0.397892 (1.189E−05)	0.397889 (3.7E−06)	0.397892 (9.74E−06)	0.397886 (6.48E−16)	0.400546 (0.00271)	0.397888 (1.27E−06)	0.397887 (0.0000)
F18	3.00000 (4.96E−15)	3.00000 (4.17E−07)	3.00003 (3.40E−05)	3.00014 (3.65E−04)	3.00000 (2.57E−13)	3.00000 (1.56E−04)	3.00000 (3.11E−06)	3.00000 (2.57E−15)
F19	-3.86146 (0.002987)	-3.85836 (0.00550)	-3.86177 (0.002109)	-3.85445 (0.01253)	-3.86278 (3.62E−11)	-3.85489 (0.00355)	-3.86278 (1.12E−06)	-3.86278 (2.68E−15)
F20	-3.23117 (0.08642)	-3.11713 (0.10387)	-3.28175 (0.07149)	-3.22736 (0.10708)	-3.23454 (0.06349)	-2.88365 (0.41253)	-3.27375 (0.06009)	-3.25462 (0.05992)
F21	-8.00253 (2.51305)	-5.21137 (0.87671)	-9.30944 (1.91518)	-7.84871 (2.68595)	-8.23099 (3.06179)	-2.14380 (1.75588)	-7.37213 (2.91485)	-6.47169 (3.20279)
F22	-8.51348 (2.75257)	-5.25759 (0.95917)	-10.40132 (0.00102)	-7.95908 (3.11684)	-9.71800 (2.12043)	-3.34127 (1.74926)	-8.48390 (2.81493)	-7.53426 (3.44292)
F23	-9.01011 (2.61416)	-5.18520 (0.99404)	-10.53463 (0.00105)	-6.90157 (3.30492)	-8.33435 (3.23138)	-3.75404 (1.65319)	-8.51946 (3.20394)	-8.49766 (3.25049)
+/-/=	~	14/5/4	16/3/4	14/1/8	15/1/7	23/0/0	16/2/5	13/4/6
Mean	2.36956	3.10869	4.08695	4.76086	4.93478	7.06521	4.97826	4.56521
Rank	1	2	3	5	6	8	7	4

From Table 3, we can see that the developed DBO technique is superior to the PSO approach in 13 functions, superior to the GWO algorithm in 16 functions, superior to the WOA in 14 functions, superior to the SSA in 15 functions, superior to the SCA in 23 functions, superior to the MVO algorithm in 16 functions, and superior to the HHO algorithm in 14 functions. Based on the Friedman test results, the mean ranking value (Mean) of the DBO algorithm is 2.36956, which is lower than other optimization approaches. In general, the developed DBO algorithm can keep an adequate balance between the local and global search abilities.

3.8 DBO's performance on CEC-BC-2017 test functions

To further verify the search performance of the DBO technique, a set of challenging test functions, CEC-BC-2017, is employed. It is worth pointing out that the f2 has been removed from the CEC-BC-2017. The experimental results of the DBO algorithm and other optimizers are illustrated in Table 4, where the mean and the Std Dev of the fitness value are used to evaluate the search accuracy and stability of the algorithm (the Std Dev is presented in parentheses).

In Table 4, the developed DBO approach achieves a smaller mean fitness value than GWO, WOA, SCA, MVO, and HHO algorithms for function f1. In addition, the DBO algorithm is able to find he global optimum for function f3, which effectively proves that this technique has the satisfactory search performance. The results of f4-f10 show that the DBO algorithm exhibits more competitive performance than the most of the compared algorithms because it reduces the possibility of falling into local optima and efficiently controls between the exploitation and exploration. For the hybrid functions (f13, f17, and f20), the DBO approach ranks first among these optimization techniques. For function f14, the DBO algorithm obtains third rank after PSO and MVO, but outperforms other five optimization algorithms (including GWO, WOA, SCA, SSA, and HHO algorithms). For functions f18 and f19, the developed DBO algorithm achieves second rank after PSO and MVO, respectively. Furthermore, in the f21-f30, the search capability of the DBO method still outperforms several well-known optimizers with high performance algorithms.

Figure 12 gives the convergence curves of the DBO, PSO, HHO, WOA, MVO, SCA, SSA, and GWO algorithms in the CEC-BC-2017 suite. For the functions, six CEC-BC-2017 functions that include unimodal (f3), multimodal (f8), hybrid (f13 and f17) and composition (f24 and f27) functions are applied to evaluate the convergence performance of the proposed DBO algorithm. Obviously, the DBO algorithm can find relatively satisfactory mean fitness values with a speed convergence than other optimization approaches in the iteration process. Similarly, the Wilcoxon signed-rank test and the Friedman test are also used to evaluate the experimental results of the algorithms. According to the results of the p-values shown in Table 5, the DBO algorithm is superior to the PSO algorithm in 14 functions, superior to the GWO algorithm in 13 functions, superior to the WOA in 23 functions, superior to the SSA in 10 functions, superior to the SCA in 21 functions, superior to the MVO algorithm in 13 functions, and superior to the HHO algorithm in 20 functions. Moreover, from the results of the Friedman test listed in Table 4, the Mean of the

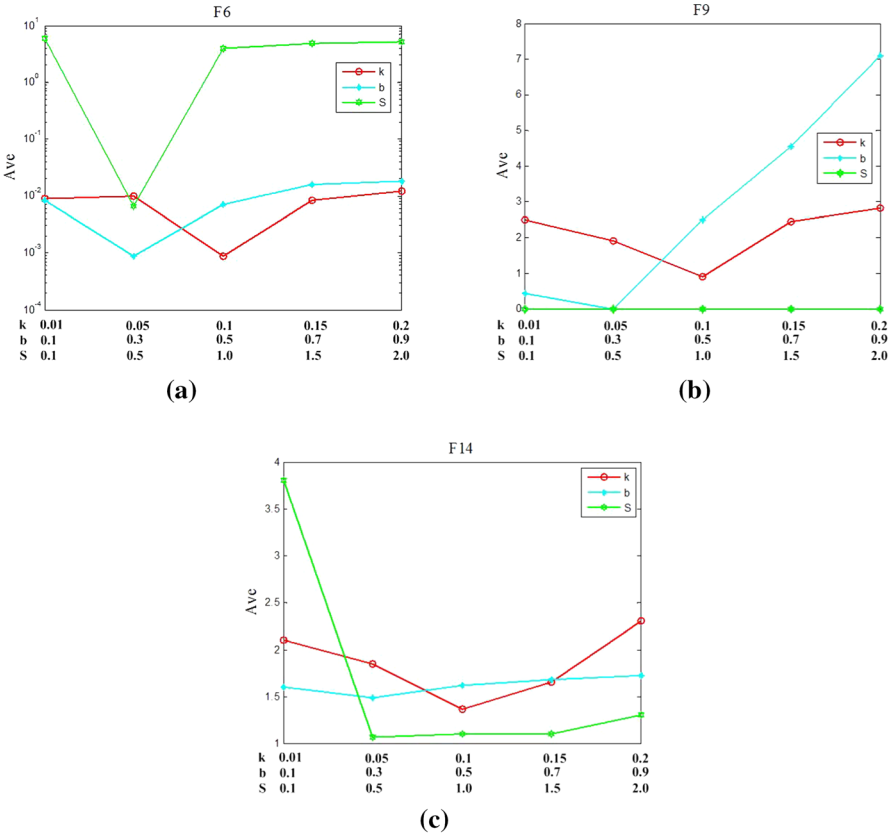


Fig. 10 Sensitivity analysis of the DBO's parameters for **a** F6, **b** F9, and **c** F14

DBO algorithm is 2.67241 and significantly outperforms the PSO algorithm, GWO algorithm, WOA, SSA, SCA, MVO algorithm, and HHO algorithm.

It is well known that the balance between the global exploration and local exploitation is crucial to finding the optimal solution successfully. In order to better quantify the process of the local and global searches for the DBO algorithm in solving the CEC-BC-2017 suite, the percentage of the exploration (Xpl) and exploitation (Xpt) is calculated by

$$\begin{aligned} \mathrm{Xpl}\% &= \frac{Div}{Div_{\max}} \times 100, \\ \mathrm{Xpt}\% &= \frac{|\, Div - Div_{\max} \,|}{Div_{\max}} \times 100, \\ Div_j &= \frac{1}{N} \sum_{i=1}^{N} median(x_j) - x_{ij}, \\ Div &= \frac{1}{D} \sum_{j=1}^{D} Div_j \end{aligned} \tag{9}$$

Table 3 p-values of Wilcoxon's signed-rank test at 5% level of significance in benchmark test functions

Function	HHO	GWO	WOA	SSA	SCA	MVO	PSO
F1	8.4660E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06
F2	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06
F3	4.0715E−05	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06
F4	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06
F5	1.7344E−06	1.7344E−06	1.7344E−06	1.9209E−06	1.7344E−06	1.7344E−06	2.8434E−05
F6	2.1336E−01	1.7344E−06	1.7344E−06	1.9209E−06	1.7344E−06	1.7344E−06	2.1266E−06
F7	1.7344E−06	1.4795E−02	5.1930E−02	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06
F8	1.7344E−06	1.9209E−06	1.0569E−04	6.1564E−04	1.7344E−06	1.1138E−03	1.4936E−05
F9	1.0000E+00	2.6002E−04	1.0000E+00	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06
F10	1.0000E+00	1.6364E−06	3.2159E−05	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06
F11	1.0000E+00	6.7889E−02	1.0000E+00	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06
F12	1.9729E−05	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.7344E−06	1.3601E−05
F13	1.7344E−06	5.1704E−01	9.2625E−01	3.5152E−06	1.7344E−06	1.4839E−03	1.7344E−06
F14	1.6046E−04	1.7333E−06	6.9838E−06	3.9614E−01	1.3595E−04	3.5888E−04	8.8911E−05
F15	6.3198E−05	7.6552E−01	5.5774E−01	8.2167E−03	7.7121E−04	8.2167E−03	1.0444E−02
F16	3.7896E−06	1.7344E−06	1.7344E−06	9.1724E−05	1.7344E−06	1.7344E−06	1.0000E+00
F17	2.5631E−06	1.7344E−06	1.7344E−06	3.1731E−01	1.7344E−06	1.7344E−06	1.0000E+00
F18	5.6061E−06	1.7344E−06	1.7344E−06	1.7267E−06	1.7344E−06	1.7344E−06	2.6998E−03
F19	8.9187E−05	4.7162E−02	8.1877E−05	5.7031E−02	7.6909E−06	5.7096E−02	2.5347E−02
F20	4.1955E−04	3.8723E−02	7.6551E−01	7.9709E−01	2.3534E−06	1.3590E−01	1.8352E−01
F21	1.7344E−06	1.8518E−02	1.7790E−01	3.4934E−01	1.7344E−06	4.9079E−01	1.6541E−01
F22	1.3601E−05	9.8421E−03	6.2683E−02	7.5213E−02	2.6033E−06	8.7740E−01	3.4381E−01
F23	4.2856E−06	2.4519E−01	4.5335E−04	3.8203E−01	4.7292E−06	3.2857E−01	8.8941E−01

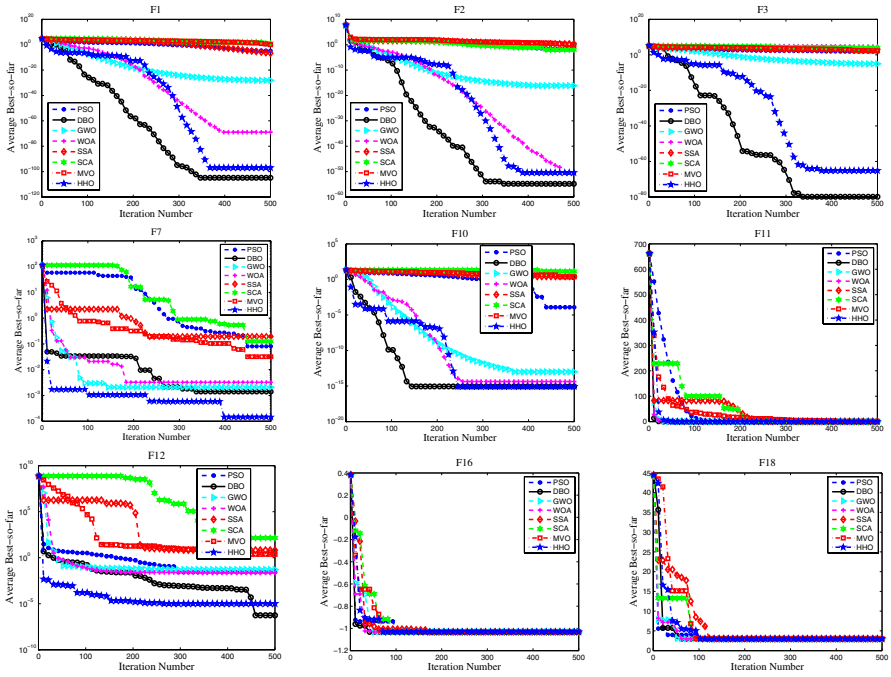


Fig. 11 The convergence curves by the DBO algorithm and other optimizers on benchmark test functions

where N is the size of the dung beetle's population, $median(x_j)$ denotes median of dimension j over all search agents, x_{ij} indicates the jth dimension of the ith agent, and D represents the dimension of the optimization problem.

During the evolution process, the exploration and exploitation ratios can be graphically evidenced in Fig. 13, where the X axis is the iteration number, and the Y axis is the percentage of the exploration and exploitation. It can be clearly seen from Fig. 13 that the DBO algorithm maintains a powerful exploration performance in the early stage of iteration process. Then, the percentage of the exploitation increases as the iteration number increases. Therefore, the DBO approach in this paper explores search space more extensively in the early part of the optimization process, and has better local exploitation in the later part of the optimization process. To summarize, our proposed DBO algorithm has a good balance the local and global searches.

4 Engineering application problems

In this section, three well-known engineering design problems are applied to test the optimization performance of the developed DBO algorithm in solving various practical applications. It should be mentioned that the penalty function is employed to deal with inequality constraints in this paper. Similar to the mathematical benchmark functions described previously, for the DBO algorithm, the population size is $N = 30$ and the maximum iteration number is $T_{\max} = 500$.

Table 4 Results of DBO, PSO, GWO, WOA, SSA, SCA, MVO and HHO on CEC-BC-2017 test functions

Function	DBO	HHO	GWO	WOA	SSA	SCA	MVO	PSO
f1	3.15E+03 (3.40E+03)	1.56E+05 (5.99E+04)	1.05E+06 (5.74E+06)	1.41E+04 (4.64E+04)	2.42E+03 (2.77E+03)	3.79E+08 (1.64E+08)	4.52E+03 (3.67E+03)	1.81E+03 (2.51E+03)
f3	3.00E+02 (8.64E−14)	3.00E+02 (1.74E−01)	5.47E+02 (6.79E+02)	3.39E+02 (5.76E+01)	3.00E+02 (1.98E−10)	8.10E+02 (2.56E+02)	3.00E+02 (2.48E−04)	3.00E+02 (5.02E−13)
f4	4.01E+02 (3.25E+00)	4.08E+02 (1.61E+01)	4.13E+02 (1.27E+01)	4.12E+02 (2.23E+01)	4.02E+02 (1.29E+01)	4.23E+02 (6.94E+00)	4.00E+02 (1.73E−01)	4.01E+02 (2.38E−01)
f5	5.29E+02 (1.37E+01)	5.38E+02 (9.74E+00)	5.13E+02 (6.35E+00)	5.43E+02 (1.49E+01)	5.16E+02 (7.72E+00)	5.39E+02 (7.84E+00)	5.31E+02 (1.08E+00)	5.36E+02 (1.24E+01)
f6	6.04E+02 (4.49E+00)	6.19E+02 (1.31E+01)	6.00E+02 (9.96E−01)	6.26E+02 (1.38E+01)	6.03E+02 (3.62E+00)	6.14E+02 (2.83E+00)	6.00E+02 (7.23E−01)	6.06E+02 (8.25E+00)
f7	7.38E+02 (1.38E+01)	7.63E+02 (1.76E+01)	7.27E+02 (9.97E+00)	7.67E+02 (2.11E+01)	7.27E+02 (8.31E+00)	7.64E+02 (8.72E+00)	7.22E+02 (6.64E+00)	7.18E+02 (4.19E+00)
f8	8.20E+02 (9.12E+00)	8.27E+02 (6.28E+00)	8.12E+02 (4.47E+00)	8.38E+02 (1.63E+01)	8.21E+02 (7.47E+00)	8.30E+02 (5.87E+00)	8.28E+02 (1.08E+01)	8.23E+02 (7.72E+00)
f9	9.21E+02 (4.50E+01)	1.20E+03 (2.35E+02)	9.24E+02 (3.39E+00)	1.16E+03 (2.37E+02)	9.01E+02 (9.41E−01)	9.58E+02 (2.32E+01)	9.00E+02 (1.16E−01)	9.00E+02 (2.11E−14)
f10	1.77E+03 (3.07E+02)	1.83E+03 (2.40E+02)	1.42E+03 (2.11E+02)	1.91E+03 (3.63E+02)	1.78E+03 (1.81E+02)	1.96E+03 (1.86E+02)	1.49E+03 (2.52E+02)	1.88E+03 (2.57E+02)
f11	1.14E+03 (4.28E+01)	1.14E+03 (5.19E+01)	1.13E+03 (3.41E+01)	1.18E+03 (9.07E+01)	1.13E+03 (1.98E+01)	1.15E+03 (1.39E+01)	1.11E+03 (4.77E+01)	1.12E+03 (1.60E+01)
f12	2.17E+05 (7.36E+05)	3.14E+05 (2.69E+05)	3.91E+05 (6.41E+05)	1.76E+06 (1.82E+06)	2.34E+04 (1.76E+04)	4.93E+06 (4.66E+06)	2.14E+04 (1.71E+04)	1.27E+04 (1.21E+04)
f13	7.37E+03 (8.19E+03)	1.55E+04 (9.28E+03)	7.52E+03 (3.44E+03)	1.42E+04 (1.21E+04)	1.38E+04 (9.06E+03)	1.15E+04 (8.25E+03)	1.12E+04 (1.16E+04)	7.74E+03 (5.58E+03)
f14	1.47E+03 (4.17E+01)	1.49E+03 (1.29E+01)	1.69E+03 (8.91E+02)	1.51E+03 (3.59E+01)	1.48E+03 (2.70E+01)	1.51E+03 (3.00E+01)	1.42E+03 (9.88E+00)	1.45E+03 (1.81E+01)
f15	1.63E+03 (7.85E+01)	1.58E+03 (5.18E+01)	2.07E+03 (1.18E+02)	2.12E+03 (7.16E+02)	1.73E+03 (1.53E+02)	1.76E+03 (4.32E+02)	1.51E+03 (5.11E+00)	1.55E+03 (2.96E+01)
f16	1.70E+03 (8.24E+01)	1.84E+03 (1.54E+02)	1.69E+03 (9.95E+01)	1.78E+03 (1.05E+02)	1.67E+03 (6.98E+01)	1.66E+03 (2.57E+01)	1.76E+03 (1.14E+02)	1.86E+03 (1.47E+02)
f17	1.74E+03 (1.56E+01)	1.75E+03 (2.55E+02)	1.75E+03 (2.27E+01)	1.77E+03 (2.85E+01)	1.76E+03 (4.63E+01)	1.76E+03 (1.08E+01)	1.77E+03 (2.52E+01)	1.76E+03 (4.38E+01)
f18	1.07E+04 (1.28E+04)	1.54E+04 (1.26E+04)	2.27E+04 (1.33E+04)	1.75E+04 (1.32E+04)	1.49E+04 (8.02E+03)	4.28E+04 (1.46E+04)	1.59E+04 (7.24E+03)	4.39E+03 (2.67E+03)
f19	1.99E+03 (1.74E+02)	3.98E+03 (2.57E+03)	3.83E+03 (4.33E+02)	1.32E+04 (1.29E+04)	2.15E+03 (5.91E+02)	2.05E+03 (9.69E+01)	1.91E+03 (2.63E+00)	2.11E+03 (2.76E+02)
f20	2.05E+03 (3.76E+01)	2.10E+03 (5.64E+01)	2.07E+03 (4.69E+01)	2.11E+03 (5.74E+01)	2.05E+03 (3.12E+01)	2.06E+03 (9.12E+00)	2.11E+03 (7.08E+01)	2.12E+03 (7.63E+01)
f21	2.20E+03 (1.46E+00)	2.28E+03 (7.47E+01)	2.31E+03 (2.85E+01)	2.30E+03 (6.98E+01)	2.22E+03 (4.91E+01)	2.21E+03 (2.22E+01)	2.31E+03 (2.99E+01)	2.31E+03 (5.57E+01)
f22	2.29E+03 (2.08E+01)	2.31E+03 (4.15E+00)	2.30E+03 (1.84E+01)	2.33E+03 (1.36E+02)	2.30E+03 (2.36E+01)	2.34E+03 (3.06E+01)	2.30E+03 (1.37E+00)	2.38E+03 (3.23E+01)
f23	2.62E+03 (5.94E+01)	2.66E+03 (2.79E+01)	2.61E+03 (7.28E+00)	2.64E+03 (1.91E+01)	2.62E+03 (7.32E+01)	2.65E+03 (4.99E+00)	2.62E+03 (4.45E+01)	2.70E+03 (3.42E+01)
f24	2.58E+03 (1.21E+02)	2.79E+03 (8.76E+01)	2.74E+03 (8.74E+00)	2.74E+03 (8.48E+01)	2.74E+03 (8.89E+01)	2.73E+03 (8.76E+01)	2.71E+03 (8.54E+01)	2.80E+03 (1.11E+02)
f25	2.91E+03 (2.58E+01)	2.92E+03 (6.27E+01)	2.94E+03 (1.46E+01)	2.93E+03 (6.53E+01)	2.92E+03 (2.34E+01)	2.94E+03 (1.39E+01)	2.93E+03 (2.31E+01)	2.92E+03 (2.21E+01)
f26	2.98E+03 (1.33E+02)	3.29E+03 (4.24E+02)	3.14E+03 (3.91E+02)	3.26E+03 (4.81E+02)	2.89E+03 (5.48E+01)	3.03E+03 (2.44E+01)	3.23E+03 (1.75E+02)	3.30E+03 (5.05E+02)
f27	3.09E+03 (2.46E+00)	3.13E+03 (3.77E+01)	3.09E+03 (6.38E+01)	3.12E+03 (2.69E+01)	3.09E+03 (3.39E+00)	3.10E+03 (1.31E+00)	3.10E+03 (1.78E+00)	3.14E+03 (4.54E+01)
f28	3.25E+03 (1.05E+02)	3.24E+03 (2.82E+01)	3.36E+03 (1.04E+02)	3.21E+03 (7.13E+01)	3.29E+03 (1.48E+01)	3.22E+03 (2.12E+01)	3.35E+03 (1.88E+02)	3.16E+03 (4.06E+01)

Table 4 (continued)

Function	DBO	HHO	GWO	WOA	SSA	SCA	MVO	PSO
f29	3.18E+03 (3.14E+01)	3.26E+03 (4.29E+02)	3.17E+03 (2.41E+01)	3.28E+03 (7.34E+01)	3.20E+03 (5.73E+01)	3.20E+03 (2.14E+01)	3.20E+03 (6.53E+01)	3.24E+03 (5.94E+01)
f30	3.29E+05 (5.09E+05)	4.07E+05 (7.47E+05)	5.34E+05 (5.87E+05)	1.87E+05 (2.66E+05)	3.68E+05 (5.47E+05)	2.81E+05 (2.27E+05)	4.36E+05 (4.80E+05)	5.27E+03 (2.69E+03)
+/−/=	~	20/1/8	13/7/9	23/0/6	10/5/14	21/1/7	13/9/7	14/9/6
Mean	2.67241	5.58620	4.46552	6.34483	3.46552	5.53448	3.72413	4.20689
Rank	1	7	5	8	2	6	3	4

4.1 Three-bar truss design problem

The design of three-bar truss is a classical engineering application problem, which has become one of the most studied cases. This problem is to achieve the minimum weight by adjusting two parameters (s_1 and s_2) on the basis of satisfying three constraints (g_1, g_2, and g_3). The objective function of this problem can be described as follows:

$$\begin{aligned} &\min f(\mathbf{s}) = (2\sqrt{2}s_1 + s_2) \times V \\ &\text{s.t. } g_1 = \frac{\sqrt{2}s_1 + s_2}{\sqrt{2}s_1^2 + 2s_1 s_2} Q - \sigma \le 0, \\ &\quad g_2 = \frac{z_2}{\sqrt{2}s_1^2 + 2s_1 s_2} Q - \sigma \le 0, \\ &\quad g_3 = \frac{1}{\sqrt{2}s_2 + s_1} Q - \sigma \le 0 \end{aligned} \tag{10}$$

where $s_i \in [0, 1]$ $(i = 1, 2)$, $V = 100\,\text{cm}$, $Q = 2\,\text{KN/cm}^2$, $\sigma = 2\,\text{KN/cm}^2$.

The results of the DBO algorithm are compared with other optimization approaches which include HHO algorithm [17], SSA [22], MVO algorithm [40], PSO-DE [42], MBA [43], and CS algorithm [44] in previous literature. As shown in Table 6, the proposed DBO algorithm demonstrates competitive result compared to the SSA, HHO algorithm, and PSO-DE algorithm. In addition, the DBO algorithm significantly outperforms the MVO algorithm, MBA, and CS algorithm. Therefore, the proposed algorithm can effectively deal with constrained problems as well.

4.2 Pressure vessel design problem

The design of the pressure vessel is to minimize the fabrication cost of a vessel based on four parameters (s_1, s_2, s_3 and s_4) and four constraints (g_1, g_2, g_3 and g_4). The fabrication cost can be formulated as follows:

$$\begin{aligned}
\min\ & f(\mathbf{s}) = 0.6224 s_1 s_3 s_4 + 1.7781 s_2 s_3^3 \\
& + 3.1661 s_1^2 s_4 + 19.84 s_1^2 s_3 \\
\text{s.t.}\ & g_1 = -s_1 + 0.0193 s_3 \le 0, \\
& g_2 = -s_2 + 0.00954 s_3 \le 0, \\
& g_3 = -\pi s_3^2 s_4 - \frac{4}{3}\pi s_3^3 + 1,296,000 \le 0, \\
& g_4 = s_4 - 240 \le 0
\end{aligned} \tag{11}$$

where $s_i \in[0,99]$ ($i = 1,2$), and $s_i \in[0,200]$ ($i = 3,4$).

The optimization results achieved by the DBO algorithm are compared with other techniques including WOA [16], HHO algorithm [17], GWO algorithm [15], CPSO algorithm [45], GSA [46], PSO algorithm [45], MVO algorithm [40], HPSO algorithm [47], and PSO-SCA [42]. In Table 7, it is clear that the DBO method ranks first and obtains the minimization of the fabrication cost among these algorithms, which demonstrates competitive result compared to the GWO and HHO algorithms. To be specific, from this table, the developed DBO algorithm obtains the best parameters with $\mathbf{s} = (0.81254, 0.402322, 42.098439, 176.6367)$ and the optimal cost $f(\mathbf{s}) = 5949.13541$. Hence, the simulation results demonstrate the advantages of the developed DBO algorithm in solving such design problems.

4.3 Welded beam design problem

The design of the welded beam is presented as the third problem in practical applications of this paper. The objective function of this problem is to discover the minimum of the manufacturing cost of the welded beam design. It should be mentioned that the welded beam design problem has four parameters (s_1, s_2, s_3 and s_4) and seven constraints (g_1, g_2, g_3, g_4, g_5, g_6, and g_7). The manufacturing cost can be formulated as follows:

$$\begin{aligned}
\min\ & f(\mathbf{s}) = 1.10471 s_1^2 s_2 + 0.04811 s_3 s_4 (14.0 + s_2) \\
\text{s.t.}\ & g_1(\mathbf{s}) = \tau(\mathbf{s}) - 13,600 \le 0, \\
& g_2(\mathbf{s}) = \sigma(\mathbf{s}) - 30,000 \le 0, \\
& g_3(\mathbf{s}) = \delta(\mathbf{s}) - 0.25 \le 0, \\
& g_4 = s_1 - s_4 \le 0, \\
& g_5 = p - p_c \le 0, \\
& g_6 = 0.125 - s_1 \le 0, \\
g_7 =& 1.10471 s_1^2 + 0.04811 s_3 s_4 (14.0 + s_2) \\
& - 5.0, \le 0, \\
& 0.1 \le s_1, s_4 \le 2.0, \\
& 0.1 \le s_2, s_3 \le 10.0
\end{aligned} \tag{12}$$

where

Table 5 p-values obtained from Wilcoxon's signed-rank test in CEC-BC-2017 test functions

Function	HHO	GWO	WOA	SSA	SCA	MVO	PSO
f1	1.7300E−06	1.0200E−01	1.1700E−02	6.8840E−01	1.7344E−06	1.4140E−01	5.4500E−02
f3	1.7322E−06	1.7235E−06	1.7344E−06	1.7344E−06	1.7322E−06	1.7344E−06	3.1730E−01
f4	3.7225E−05	1.9197E−06	3.7225E−05	5.3070E−05	1.7333E−06	8.9187E−05	5.8570E−01
f5	3.8700E−02	2.5942E−05	2.8000E−03	7.5104E−05	2.4000E−03	4.1200E−02	1.7500E−02
f6	3.1798E−06	8.9034E−05	2.3520E−06	4.7790E−01	2.1266E−06	2.1266E−06	4.0480E−01
f7	3.7225E−05	1.3000E−03	6.3391E−06	2.0515E−04	6.9838E−06	1.6378E−05	5.7451E−06
f8	1.4000E−03	8.9415E−04	2.1630E−05	6.4350E−01	4.8969E−04	3.4000E−03	3.7640E−01
f9	8.4614E−06	3.9330E−02	2.6017E−06	5.2135E−06	3.0650E−04	1.9197E−06	2.5562E−06
f10	6.8840E−01	2.4109E−04	2.7120E−01	9.9180E−01	3.8700E−02	1.0000E−03	1.3689E−02
f11	8.6120E−01	1.5880E−01	7.1900E−02	1.3590E−01	2.7120E−01	4.2832E−06	2.3000E−02
f12	4.5272E−04	2.3000E−03	2.1587E−05	3.0860E−01	1.7333E−06	2.3690E−01	2.4300E−02
f13	8.9443E−04	3.0860E−01	3.8700E−02	2.0000E−03	1.3200E−02	4.0218E−02	4.7542E−02
f14	7.8600E−02	1.3289E−02	3.2000E−03	7.0360E−01	9.6236E−06	3.1817E−06	9.8000E−03
f15	1.8500E−02	8.3463E−04	6.3169E−05	6.0000E−03	1.4700E−01	1.7344E−06	2.3693E−05
f16	1.2000E−03	3.0860E−01	3.6000E−03	1.4800E−02	2.8500E−02	9.7932E−03	1.1499E−04
f17	3.1850E−01	2.1330E−01	1.2000E−03	2.9760E−02	6.8892E−05	1.0725E−03	2.4300E−02
f18	8.2200E−02	2.0000E−03	3.5000E−02	3.0000E−02	3.5132E−06	2.6700E−02	5.3000E−03
f19	2.3534E−06	1.8364E−05	2.5970E−06	1.2500E−02	1.4000E−03	1.7344E−06	2.0700E−02
f20	4.8935E−04	3.0860E−01	2.6124E−04	1.9150E−01	1.1090E−01	3.9000E−03	3.3161E−04
f21	2.4000E−03	1.7333E−06	1.9197E−06	1.7140E−01	1.7344E−06	2.3534E−06	1.1939E−05
f22	1.1241E−05	1.1560E−01	1.4000E−03	2.6220E−01	8.4614E−06	7.971E−01	6.6417E−06
f23	6.3249E−06	6.8892E−05	3.1850E−01	8.1842E−01	1.1494E−04	8.7846E−01	1.7300E−06
f24	3.5152E−06	5.2828E−05	1.0561E−04	5.3046E−05	4.4452E−05	9.6266E−04	4.2163E−06
f25	7.1890E−01	3.2000E−03	7.1900E−02	5.9990E−01	2.8000E−03	8.7740E−01	6.7320E−01
f26	1.3000E−03	3.8200E−01	8.9415E−04	8.9415E−04	1.7790E−01	6.6452E−03	1.6600E−02
f27	1.7333E−06	1.2381E−05	1.7333E−06	8.9443E−04	1.7322E−06	6.1826E−05	1.4920E−05
f28	9.0990E−01	7.1547E−04	3.1460E−01	3.1936E−02	5.3040E−01	3.3300E−02	4.1681E−05
f29	7.6738E−06	1.3000E−03	6.9799E−06	3.0860E−01	1.2040E−01	5.1700E−01	3.5876E−04
f30	6.8840E−01	1.5880E−01	6.2880E−01	7.3430E−01	6.5830E−01	5.8570E−01	5.7484E−06

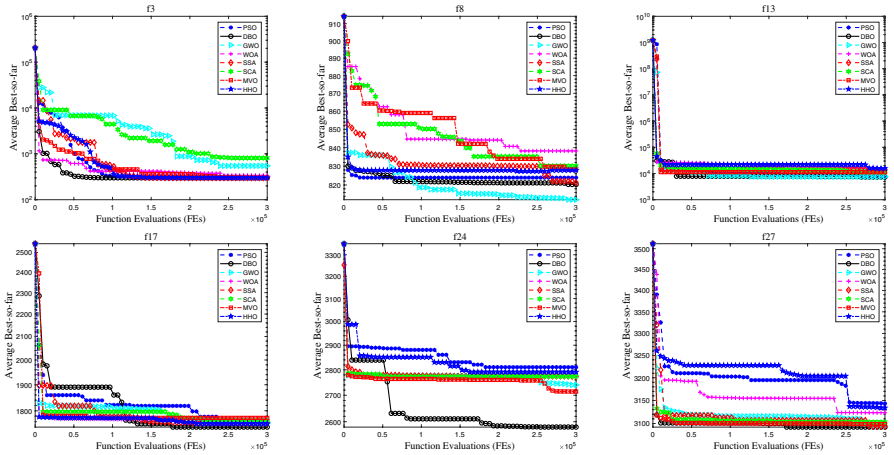
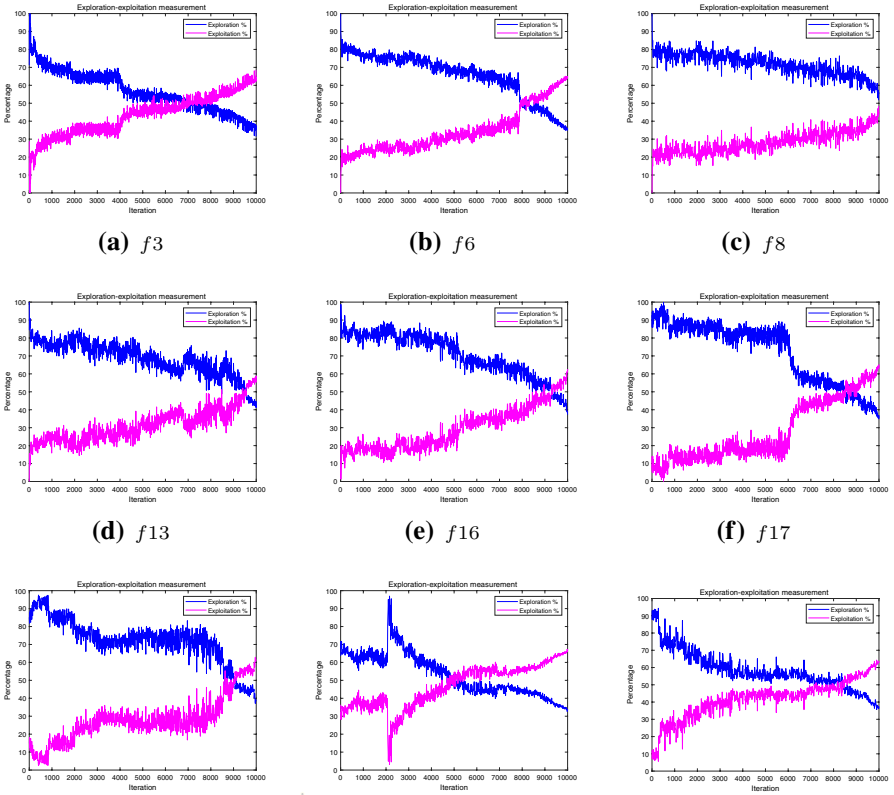


Fig. 12 Convergence curves of the DBO, PSO, GWO, WOA, SSA, SCA, MVO and HHO on CEC-BC-2017 test functions



(a) $f3$ **(b)** $f6$ **(c)** $f8$

(d) $f13$ **(e)** $f16$ **(f)** $f17$

(g) $f20$ **(h)** $f24$ **(i)** $f27$

Fig. 13 Exploration and exploitation of the DBO algorithm with different functions on the CEC 2017

$$\tau = \sqrt{\tau_1^2 + 2\tau_1\tau_2(\frac{s_2}{2r}) + \tau_2^2}, \tau_1 = \frac{p}{s_1 s_2\sqrt{2}},$$

$$m = p(l + \frac{s_2}{2}), j = 2\{\sqrt{2}s_1 s_2[\frac{s_2^2}{12} + (\frac{s_1 + s_3}{2})^2]\},$$

$$r = \sqrt{\frac{s_2^2}{4} + (\frac{s_1 + s_3}{2}^2)}, \sigma = \frac{6pl}{s_4 s_3^2}, \delta = \frac{6pl^3}{E s_3^2 s_4},$$

$$p_c = \frac{4.013E\sqrt{\frac{s_3^2 s_4^6}{36}}}{l^2}(1 - \frac{s_3}{2l}\sqrt{\frac{E}{4G}}),$$

$$G = 12 \times 10^6 \text{ psi}, E = 30 \times 10^6 \text{ psi},$$

$$p = 6000 \text{ lb}, l = 14 \text{ in}, \tau_2 = \frac{mr}{j}.$$

There are several optimization algorithms previously applied to the welded beam design problem such as the HHO algorithm [17], GSA [46], WOA [16], MVO algorithm [40], MPA [48], CPSO algorithm [49], HS algorithm [50], and SSA [22]. The optimal results with regard to the design process of each algorithm are illustrated in Table 8. It can be seen from Table 8 that the DBO algorithm obtains the optimal solution $\mathbf{s} = (0.20162, 3.3477, 9.0405, 0.20606)$ with the minimum of the manufacturing cost equal to 1.7050958. It is worth mentioning that the DBO technique has a substantial progress compared with GSA and HS algorithm.

Once more, all these simulation results can prove the advantages of the developed DBO approach in solving practical engineering applications with many inequality constraints. Therefore, we can conclude that the developed DBO algorithm shows the competitive performance compared to the state-of-the-art optimization algorithms.

Table 6 Optimum results and comparison for the three-bar truss design problem

Algorithm	s_1	s_2	Optimal weight
DBO	0.788662816000217	0.4082831338318	263.8958434
HHO	0.788662816	0.408283133832900	263.8958434
SSA	0.788665414258065	0.408275784444547	263.8958434
MVO	0.78860276	0.40845307	263.8958499
PSO-DE	0.7886751	0.4082482	263.8958433
MBA	0.7885650	0.4085597	263.8958522
CS	0.78867	0.40902	263.9716

Table 7 Optimum results and comparison for the pressure vessel design problem

Algorithm	s_1	s_2	s_3	s_4	Optimal cost
DBO	0.81254	0.402322	42.098439	176.6367	5949.13541
HHO	0.81758383	0.4072927	42.09174576	176.7196352	6000.46259
PSO	0.8125	0.4375	42.091266	176.746500	6061.0777
WOA	0.812500	0.437500	42.0982699	176.638998	6059.7410
GWO	0.812500	0.434500	42.089181	176.758731	6051.5639
CPSO	0.812500	0.437500	42.091266	176.746500	6061.0777
GSA	1.125000	0.625000	55.9886598	84.4542025	8538.8359
MVO	0.8125	0.4375	42.0907382	176.738690	6060.8066
HPSO	0.812500	0.437500	42.0984	176.6366	6059.7143
PSO-SCA	0.8125	0.4375	42.098446	176.6366	6059.71433

5 Conclusion

A novel SI-based technique called the DBO algorithm has been put forward in this paper with the hope of providing a more efficient optimizer to deal with complex optimization problems. It is worth mentioning that five different updating rules inspired by the ball-rolling, dancing, foraging, stealing, and reproduction behaviors of dung beetles in nature have been designed to help find high-quality solutions. The DBO algorithm has demonstrated competitive search performance in terms of the convergence rate, solution accuracy, and stability by comparing with seven well-studied optimization techniques (the HHO, PSO, WOA, MVO, SCA, DA, and GWO algorithms) on a total of 52 test functions including both 23 classical functions and 29 CEC-BC-2017 test functions. In addition, the DBO algorithm has been

Table 8 Optimum results and comparison for the welded beam design problem

Algorithm	s_1	s_2	s_3	s_4	Optimal cost
DBO	0.20162	3.3477	9.0405	0.20606	1.7050958
HHO	0.204039	3.531061	9.027463	0.206147	1.73199057
GSA	0.182129	3.856979	10.0000	0.202376	1.879952
WOA	0.205396	3.484293	9.037426	0.206276	1.730499
MVO	0.205463	3.473193	9.044502	0.205695	1.72645
MPA	0.205728	3.470509	9.036624	0.205730	1.724853
CPSO	0.202369	3.544214	9.048210	0.205723	1.72802
HS	0.2442	6.2231	8.2915	0.2443	2.3807
RO	0.203687	3.528467	9.004233	0.207241	1.735344
SSA	0.2057	3.4714	9.0366	0.2057	1.72491

successfully employed in solving three engineering design problems. Experimental results have also been conducted to verify the superiority of the DBO algorithm compared to other optimizers.

It should be noticed that the developed DBO algorithm is a SI-based gradient-free optimizer. The following characteristics can theoretically help us to realize why the DBO algorithm is more competitive than other algorithms in terms of exploring or exploiting:

1) A novel search mechanism of the ball-rolling dung beetle is proposed, where the different searching patterns make it possible for us to: a) thoroughly explore the searching space by using the information of different time periods; and b) pursue stronger searching ability in order to avoid trapping into local optimum.

2) R parameter has the feature of dynamic change, which can further stimulate the exploration and exploitation states of the DBO algorithm.

3) Different regional search strategies (including the spawning area and the optimal foraging area) can promote the exploitative behaviors of the DBO algorithm.

4) Different updating rules can ensure that the developed DBO algorithm maintains an adequate balance between the local and global search abilities.

Author contributions JX contributed to conceptualization, methodology, software, investigation, writing-original draft. BS contributed to conceptualization, writing-review, editing, supervision, funding acquisition.

Funding This work was supported in part by the National Natural Science Foundation of China under Grants 61873059 and 61922024, and the Program of Shanghai Academic/Technology Research Leader under Grant 20XD1420100.

Data availibility All data generated or analysed during this study are included in this published article.

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Qin Y, Jin L, Zhang A, He B (2020) Rolling bearing fault diagnosis with adaptive harmonic kurtosis and improved bat algorithm. IEEE Trans Instrum Meas 70:1–12
2. Li M, Yan C, Liu W, Liu X, Zhang M, Xue J (2022) Fault diagnosis model of rolling bearing based on parameter adaptive AVMD algorithm. Appl Intell. https://doi.org/10.1007/s10489-022-03562-9
3. Karami H, Ehteram M, Mousavi S-F, Farzin S, Kisi O, El-Shafie A (2019) Optimization of energy management and conversion in the water systems based on evolutionary algorithms. Neural Comput Appl 31(10):5951–5964
4. Singh AR, Ding L, Raju DK, Raghav LP, Kumar RS (2022) A swarm intelligence approach for energy management of grid-connected microgrids with flexible load demand response. Int J Energy Res 46(4):301–4319
5. Li J, Lei Y, Yang S (2022) Mid-long term load forecasting model based on support vector machine optimized by improved sparrow search algorithm. Energy Rep 8:491–497
6. Wei D, Wang J, Li Z, Wang R (2021) Wind power curve modeling with hybrid copula and grey wolf optimization. IEEE Trans Sustain Energy 13(1):265–276
7. Zhang Y, Mo Y (2022) Chaotic adaptive sailfish optimizer with genetic characteristics for global optimization. J Supercomput 78:10950–10996. https://doi.org/10.1007/s11227-021-04255-9

8. Abdulhammed O (2022) Load balancing of IoT tasks in the cloud computing by using sparrow search algorithm. J Supercomput 78:3266–3287. https://doi.org/10.1007/s11227-021-03989-w
9. Wu G (2016) Across neighborhood search for numerical optimization. Inf Sci 329:597–618
10. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks, 1942–1948
11. Liu W, Wang Z, Yuan Y, Zeng N, Hone K, Liu X (2021) A novel sigmoid-function-based adaptive weighted particle swarm optimizer. IEEE Trans Cybern 51(2):1085–1093
12. Liu J, Yang J, Liu H, Tian X, Gao M (2017) An improved ant colony algorithm for robot path planning. Soft Comput 21(19):5829–5839
13. Drigo M (1996) The ant system: optimization by a colony of cooperating agents. IEEE Trans Syst Man Cybern Part B Cybern 26(1):1–13
14. Li M, Xu G, Fu B, Zhao X (2022) Whale optimization algorithm based on dynamic pinhole imaging and adaptive strategy. J Supercomput 78:6090–6120. https://doi.org/10.1007/s11227-021-04116-5
15. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey wolf optimizer. Adv Eng Softw 69:46–61
16. Mirjalili S, Lewis A (2016) The whale optimization algorithm. Adv Eng Softw 95:51–67
17. Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H (2019) Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 97:849–872
18. Abbasi A, Firouzi B, Sendur P (2021) On the application of Harris Hawks Optimization (HHO) algorithm to the design of microchannel heat sinks. Eng Comput 37(2):1409–1428
19. Cai J, Luo T, Xu G, Tang Y (2022) A novel biologically inspired approach for clustering and multilevel image thresholding: modified harris hawks optimizer. Cogn Comput. https://doi.org/10.1007/s12559-022-09998-y
20. Liu C (2021) An improved Harris Hawks Optimizer for job-shop scheduling problem. J Supercomput 77:14090–14129. https://doi.org/10.1007/s11227-021-03834-0
21. Gandomi AH, Alavi AH (2012) Krill herd: a new bio-inspired optimization algorithm. Commun Nonlinear Sci 17(12):4831–4845
22. Mirjalili S, Gandomi AH, Mirjalili SZ, Saremi S, Faris H, Mirjalili SM (2017) Salp swarm algorithm: a bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
23. Xue J, Shen B (2020) A novel swarm intelligence optimization approach: sparrow search algorithm. Syst Sci Control Eng 8(1):22–34
24. Yang X-S (2010) Firefly algorithm, stochastic test functions and design optimisation. Int J Bio-Inspired Comput 2(2):78–84
25. Yang X-S (2010) A new metaheuristic bat-inspired algorithm. In: Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), pp 65–74
26. Ebadinezhad S (2020) DEACO: adopting dynamic evaporation strategy to enhance ACO algorithm for the traveling salesman problem. Eng Appl Artif Intel 92:103649
27. Yang K, You X, Liu S, Pan H (2020) A novel ant colony optimization based on game for traveling salesman problem. Appl Intell 50(12):4529–4542
28. Liu Y, Chen S, Guan B, Xu P (2019) Layout optimization of large-scale oil-gas gathering system based on combined optimization strategy. Neurocomputing 332:159–183
29. Huang M, Lin H, Yunkai H, Jin P, Guo Y (2012) Fuzzy control for flux weakening of hybrid exciting synchronous motor based on particle swarm optimization algorithm. IEEE Trans Magn 48(11):2989–2992
30. Zeng N, Wang Z, Liu W, Zhang H, Hone K, Liu X (2020) A dynamic neighborhood-based switching particle swarm optimization algorithm. IEEE Trans Cybern. https://doi.org/10.1109/TCYB.2020.3029748
31. Liu W, Wang Z, Liu X, Zeng N, Bell D (2018) A novel particle swarm optimization approach for patient clustering from emergency departments. IEEE Trans Evol Comput 23(4):632–644
32. Guo Q, Gao L, Chu X, Sun H (2022) Parameter identification of static var compensator model using sensitivity analysis and improved whale optimization algorithm. CSEE J Power Energy 8(2):535–547
33. Zhong C, Li G (2022) Comprehensive learning Harris Hawks-Equilibrium optimization with terminal replacement mechanism for constrained optimization problems. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2021.116432
34. Chang Z, Gu Q, Lu C, Zhang Y, Ruan S, Jiang S (2021) 5G private network deployment optimization based on RWSSA in open-pit mine. IEEE Trans Ind Inform. https://doi.org/10.1109/TII.2021.3132041
35. Dacke M, Baird E, El JB, Warrant EJ, Byrne M (2021) How dung beetles steer straight. Annu Rev Entomol 66:243–256

36. Byrne M, Dacke M, Nordström P, Scholtz C, Warrant E (2003) Visual cues used by ball-rolling dung beetles for orientation. J Comp Physiol A 189(6):411–418
37. Dacke M, Nilsson D-E, Scholtz CH, Byrne M, Warrant EJ (2003) Insect orientation to polarized moonlight. Nature 424(6944):33–33
38. Yin Z, Zinn-Björkman L (2020) Simulating rolling paths and reorientation behavior of ball-rolling dung beetles. J Theor Biol 486:110106
39. Awad NH, Ali MZ, Suganthan PN (2017) Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems. In: 2017 IEEE Congress on Evolutionary Computation (CEC), pp 372–379
40. Mirjalili S, Mirjalili SM, Hatamlou A (2016) Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput Appl 27(2):495–513
41. Mirjalili M (2016) SCA: a sine cosine algorithm for solving optimization problems. Knowl Based Syst 96:120–133
42. Liu H, Cai Z, Wang Y (2010) Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. Appl Soft Comput 10(2):629–640
43. Sadollah A, Bahreininejad A, Eskandar H, Hamdi M (2013) Mine blast algorithm: a new population based algorithm for solving constrained engineering optimization problems. Appl Soft Comput 13(5):2592–2612
44. Gandomi AH, Yang X-S, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35
45. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intel 20(1):89–99
46. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2009) GSA: a gravitational search algorithm. Inf Sci 179(13):2232–2248
47. He Q, Wang L (2007) A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Appl Math Comput 186(2):1407–1422
48. Faramarzi A, Heidarinejad M, Mirjalili S, Gandomi AH (2020) Marine predators algorithm: A nature-inspired metaheuristic. Expert Syst Appl. https://doi.org/10.1016/j.eswa.2020.113377
49. Krohling RA, Coelho LS (2006) Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. IEEE Trans Syst Man Cybern Part B Cybern 36(6):1407–1416
50. Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Method Appl Mech Eng 194(36–38):3902–3933