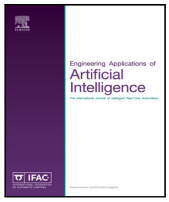
Engineering Applications of Artificial Intelligence 114 (2022) 105075

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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



Dandelion Optimizer: A nature-inspired metaheuristic algorithm for engineering applications

Shijie Zhao [a,b,*], Tianran Zhang [a], Shilin Ma [a], Miao Chen [a]

[a] Institute of Intelligence Science and Optimization, Liaoning Technical University, Fuxin, 123000, China
[b] Institute for Optimization and Decision Analytics, Liaoning Technical University, Fuxin, 123000, China

ARTICLE INFO

Keywords:
Nature-inspired metaheuristic algorithm
Swarm intelligence
Dandelion optimizer
Real-world optimization problems

ABSTRACT

This paper proposes a novel swarm intelligence bioinspired optimization algorithm, called the Dandelion Optimizer (DO), for solving continuous optimization problems. DO simulates the process of dandelion seed long-distance flight relying on wind, which is divided into three stages. In the rising stage, seeds raise in a spiral manner due to the eddies from above or drift locally in communities according to different weather conditions. In the descending stage, flying seeds steadily descend by constantly adjusting their direction in global space. In the landing stage, seeds land in randomly selected positions so that they grow. The moving trajectory of a seed in the descending stage and landing stage are described by Brownian motion and a Levy random walk. CEC2017 benchmark functions are utilized to evaluate the performance of DO, including the optimization accuracy, stability, convergence, and scalability, through a comparison with 9 well-known nature-inspired metaheuristic algorithms. Finally, the applicability of DO is verified by solving 4 real-world optimization problems. The experimental results indicate that the proposed DO method is a higher performing optimizer with outstanding iterative optimization and strong robustness compared with well-established algorithms. Source codes of DO are publicly available at https://ww2.mathworks.cn/matlabcentral/fileexchange/114680-dandelion-optimizer.

1. Introduction

Optimization is the process of seeking the optimal value of an objective function under a series of constraints. Presently, the complexity and diversity of engineering applications require optimization problems that are increasingly characterized by large-scale variable dimensions and high-dimensional function objectives. However, some canonical algorithms, such as Newton's method (Galli and Lin, 2021), the steepest descent method (Pu et al., 2013), and integer programming (Cornuéjols, 2008), tend to sink into local optima due to the lack of random operators when tackling such complex optimization problems. Moreover, canonical algorithms have high requirements on the continuity and differentiability of the objective function, so it is generally acknowledged that canonical algorithms are not conducive to the development of modern universal optimization technology (Jain et al., 2019). Therefore, to handle modern optimization problems effectively, it is crucial to introduce stochastic technology into the optimization method, which leads to the emergence of metaheuristic algorithms. In essence, metaheuristics are general algorithmic frameworks, often nature-inspired (Bianchi et al., 2009; Blum and Roli, 2003). Randomness is the main characteristic of nature-inspired metaheuristic algorithms (Back, 1996). Such randomness is in a more intelligent way, which moves or generates new solutions through learning strategies to effectively approach the optimal solution (Dorigo and Stützle, 2019). Thus, a nature-inspired metaheuristic algorithm has a greater capacity to escape local extremes than canonical optimization techniques. Accordingly, nature-inspired metaheuristic algorithms do not require too much prior information about the problem, but focus on several heuristic paradigms to describe the candidate solutions (Halim et al., 2021). Hence, they are broadly applied to black box problems. Nature-inspired metaheuristic algorithms, which are not dependent on gradient information and benefit from an uncomplicated core idea, are successful substitutes for canonical algorithms and can tackle challenging problems more effectively. They have been widely used in many fields, such as mechanical design (Gupta et al., 2021), image segmentation (Kurban et al., 2021), parameter optimization (Zhou et al., 2021), deep learning (Chan-Ley and Olague, 2020), and video tracking (Soubervielle-Montalvo et al., 2022).

In general, nature-inspired metaheuristic algorithms can be roughly divided into four categories based on different heuristic mechanisms (see Fig. 1). The first category is evolutionary algorithms (EAs) based on genetic and mutated organisms. Such algorithms generate new individuals by evolutionary operators, the individuals gradually develop into better individuals compared with the last generation, and finally, they search for the vicinity of the optimal solution (Fonseca and Fleming, 1995). Therefore, EAs possess strong global optimization ability and local extremum avoidance. The main representative

* Corresponding author at: Institute of Intelligence Science and Optimization, Liaoning Technical University, Fuxin, 123000, China.
E-mail address: zhaoshijie@lntu.edu.cn (S. Zhao).

https://doi.org/10.1016/j.engappai.2022.105075
Received 13 January 2022; Received in revised form 28 May 2022; Accepted 11 June 2022
Available online 16 July 2022
0952-1976/

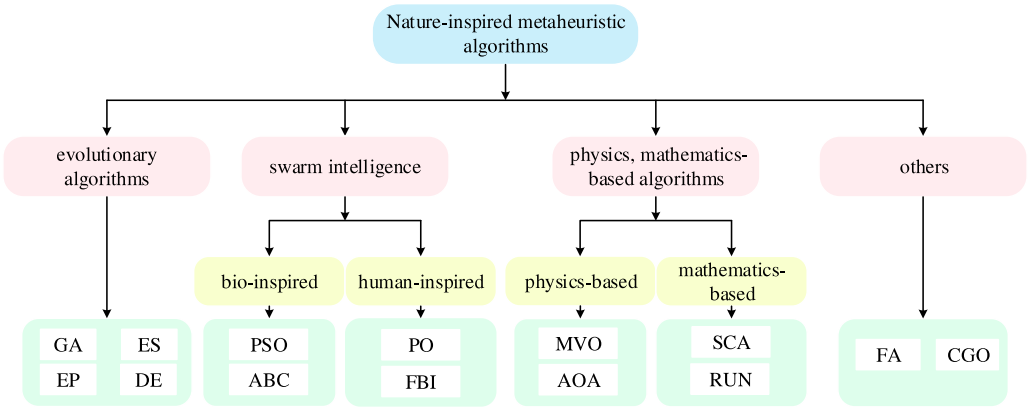


Fig. 1. Classification of the nature-inspired metaheuristic algorithms.

algorithm of this kind is the genetic algorithm (GA) (Holland, 1992), which simulates the natural selection and genetic mechanism of Darwinian evolution. Heredity and variation are the core operations of the GA. Other popular examples of EAs include evolutionary programming (EP) (Fogel, 1998), evolution strategy (ES) (Nand et al., 2021), and differential evolution (DE) (Storn and Price, 1997). The second category is the swarm-intelligence (SI) optimization algorithm, which is inspired by the intelligent behaviour of animals, plants or other organisms and belongs to a more mature class of nature-inspired metaheuristic algorithms. Predation, reproduction, and hunting are common social behaviours. SI is activated by a group of randomly produced solutions in the search space, and constructs several heuristic methods to mimic such social behaviours to achieve global and local iterative optimization of the algorithm. It is possible to make multiple search agents communicate with each other and show complex and ordered swarm intelligence behaviour through cooperation among individuals (Zahedi et al., 2016). Many well-known algorithms in this class have been proposed. Particle swarm optimization (PSO) (Kennedy and Eberhart, 1995) was developed to mimic the predation behaviour of birds, in which the particle utilizes speed and position to collect important information in the search process, and the particles are continuously updated to realize iterative optimization. The artificial bee colony (ABC) (Karaboga and Basturk, 2007) approach simulates the honey-gathering behaviour of bees. The motivation of ant colony optimization (ACO) (Dorigo and Blum, 2005) comes from the path found by ants in the process of looking for food. In recent years, several new SI algorithms have been proposed and include the whale optimization algorithm (WOA) (Mirjalili and Lewis, 2016), the moth swarm algorithm (MSA) (Mohamed et al., 2017), Harris hawks optimization (HHO) (Heidari et al., 2019), the seagull optimization algorithm (SOA) (Dhiman and Kumar, 2019), the tunicate swarm algorithm (TSA) (Kaur et al., 2020), the chimp optimization algorithm (ChOA) (Khishe and Mosavi, 2020), levy flight distribution (LFD) (Houssein et al., 2020), the horse herd optimization algorithm (HOA) (MiarNaeimi et al., 2021), the aquila optimizer (AO) (Abualigah et al., 2021), the dwarf mongoose optimization algorithm (DMO) (Agushaka et al., 2022), and the snake optimizer (SO) (Hashim and Hussien, 2022). In addition, some algorithms based on humans-specific behaviour have been developed, including the forensic-based investigation algorithm (FBI) (Chou and Nguyen, 2020), and political optimizer (PO) (Askari et al., 2020). The third category is based on physics or mathematics. This kind of algorithm can describe some physical phenomena to achieve individual iterative optimization by borrowing physical theories or mathematical formulas (Mohammadi-Balani et al., 2021). Benefiting from the fact that such algorithms are usually able to logically explain the resulting heuristic paradigm theory, an increasing number of mathematical-based or physics-based techniques are being studied. Physics-based algorithms include the multiverse optimizer (MVO) (Mirjalili et al., 2016), yin-yang-pair Optimization (YYPO) (Punnathanam and Kotecha, 2016), the Lichtenberg algorithm (LA) (Pereira et al., 2021), the Archimedes optimization algorithm (AOA) (Hashim et al., 2021), and atomic orbital search (AOS) (Azizi, 2021). Methods based on mathematics include the sine cosine algorithm (SCA) (Mirjalili, 2016), golden ratio optimization method (GROM) (Nematollahi et al., 2020), and Runge Kutta method (RUN) (Ahmadianfar et al., 2021). The last category includes other methods, which are based on real-life situations. Such algorithms include the fireworks algorithm (FA) (Tan and Zhu, 2010) and chaos game optimization (CGO) (Talatahari and Azizi, 2021).

However, nature-inspired metaheuristic algorithms also have some shortcomings. For example, the sine cosine algorithm and whale optimization algorithm are sensitive to dimension expansions. Although they have good optimization performance in low-dimensional problems, they have slow convergence speeds and easily fall into local optima in high-dimensional problems. Harris hawks optimization only uses linear decline factors, resulting in a poor balance between exploitation and exploration. Therefore, it is weak in avoiding local extreme values of complex functions. Too many parameters need to be optimized for the horse herd optimization algorithm. Levy flight distribution has higher convergence accuracy but a longer running time.

Proverbially, exploration and exploitation are two of the most important aspects of nature-inspired metaheuristic algorithms (Hussain et al., 2019). Excessive exploration ultimately leads to difficulties for the convergence of the algorithm, while focusing only on exploitation causes the model to easily to fall into local optima. Hence, how to strike a balance between exploration and exploitation is still a problem to be solved. The *no free lunch* (NFL) theorem (Wolpert and Macready, 1997) states that for any algorithm, an improvement in performance on one class of problems will be offset by a decrease in performance on another. No algorithm can be completely suitable for dealing with all kinds of problems. Optimization theory encourages more algorithms to solve complex problems.

In this paper, a novel swarm-intelligence bioinspired optimization algorithm, called Dandelion Optimizer (DO), is proposed to tackle continuous optimization problems. Comparatively, the previous studies about dandelion plants were mainly inspired by the sowing behaviour. In detail, the dandelion algorithm (Gong et al., 2018) updated the next generation of individuals by dynamically regulating the seeding radius of dandelions and their autonomous learning. And based on it, a variant dandelion algorithm (Li et al., 2017) divided the population of dandelions into two subgroups: core dandelion and assistant dandelions, and employed different sowing behaviours for both different subgroups. On the other hand, the long-distance flight of a dandelion seed is another important behaviour for the biological evolution, and there are no reports based on this behaviour in previous study. Under the influence of wind and weather, different flight strategies cause dandelions to land

(a) Dandelions growing (b) Dandelions floating in the wind

Fig. 2. Dandelions in nature.

in different locations. DO simulates a lifetime journey of dandelion seeds flying in the wind as they mature. The contributions of this work are presented as follows.

- According to the characteristics of the long-distance flight of dandelion seeds, mathematical models of dandelion seeds from the rising stage, descending stage and landing stage are constructed under different weather conditions, and the design expressions are explained in detail.
- The international standard CEC2017 benchmark functions include unimodal, multimodal and composition functions, and function optimization is challenging. The proposed algorithm is tested on the CEC2017 suite.
- Statistical analyses, convergence analyses, expansibility analyses, Wilcoxon tests, and Friedman tests are employed to evaluate the proposed DO algorithm, and the results are strictly compared with 9 well-known nature-inspired metaheuristic algorithms.
- Four classic real-world optimization problems, namely, speed reduction design, tension/compression spring design, welded beam design, and pressure vessel design, are utilized to evaluate the constrained optimization of the proposed DO algorithm. Furthermore, fewer evolutionary iterations than used by existing algorithms are used to verify the efficiency of the proposed DO algorithm.

The rest of this paper is organized as follows. Section 2 introduces the inspiration of DO and the detailed mathematical model building process. In Section 3, the performance of DO is evaluated on CEC2017 benchmark functions and compared with 9 well-known nature-inspired metaheuristic algorithms. Section 4 describes the application of DO in real-world optimization problems. Finally, Section 5 summarizes the conclusions and gives prospects for future work.

2. Dandelion optimizer

This section describes DO in detail. First, the biological mechanism and motivation of the proposed DO are presented. Then, the mathematical model of DO is formulated, and its expressions are given. Finally, the complexity of DO is analysed and compared with that of other algorithms.

2.1. Inspiration

A dandelion (see Fig. 2), scientifically known as *Herba taraxaci*, is a perennial herb in the Asteraceae family. These plants can reach more than 20 cm high. Dandelion heads are shaped like inflorescences. The seeds are usually composed of hundreds of crested hairs, a beak, and an achene (Meng et al., 2016). When the seeds mature, the wind carries them to new locations to breed life. Crown hairs play an important role in the dispersal of Asteraceae seeds. Because they prolong the descent of the seeds, these hairs enable the seeds to be blown farther by the wind (Sheldon and Burrows, 1973). A dandelion is one of the most representative plants that relies on wind for seed propagation. Its seed can fly dozens of kilometres in the wind under the right conditions. As a dandelion seed flies, it forms two vortices, which create upwards drag. When seeds fall at lower speeds, the two vortices above become larger and symmetrical (Cummins et al., 2018). A symmetrical vortex ensures the steady descent of the seed; namely, the filament is level with the ground, and the fruit points downwards. For dandelion seeds to fly long distances, they need to be kept at a relatively stable altitude. The separated vortex is maintained at a fixed distance below the dandelion crown (Cavieres et al., 2008). Curiously, the porosity of the dandelion crown appears to be precisely regulated to stabilize the vortex ring. Crested hairs are made of slender filaments that radiate from a central handle, similar to spokes on a bicycle wheel. Such seeds always have between 90 and 110 spokes. This consistency is the key to the stability of the separation vortex above a dandelion seed, thus helping the seed to remain stable during long-distance flight (Casseau et al., 2015).

Wind speeds and weather are the two primary factors that affect the spread of dandelion seeds. The wind speed is used to determine whether a seed flies long or short distances (Soons et al., 2004). Weather controls whether dandelion seeds can fly and influences the dandelion's ability to grow in nearby or far away spaces. Dandelion seeds travel through three stages, which are reported below.

- In the rising stage, a vortex is generated above the dandelion seed and it rises under the action of a dragging force with sunny and windy weather. Conversely, the weather is rainy, there are no eddies above the seeds. In this case, the search can only be performed locally.
- In the descending stage, after seeds rise to a certain height, they drop steadily.
- In the landing stage, dandelion seeds eventually randomly land in one place under the influence of wind and weather to grow new dandelions.

Dandelions evolve their population by passing their seeds to the next generation on a three-stage basis. The main inspiration of DO in this paper comes from the above three stages. The following subsections describe how to model these behaviours in DO.

2.2. Mathematical model

The mathematical expressions for DO are described specifically in this subsection. First, the mathematical expressions of the two kinds of weather conditions are given, and then the mathematical models for the descending stage and landing stage are analysed.

2.2.1. Initialization

Similar to other natured-inspired metaheuristic algorithms, DO fulfils population evolution and iterative optimization on the basis of population initialization. In the proposed DO algorithm, it is assumed that each dandelion seed represents a candidate solution, whose population is expressed as

$$population = \begin{bmatrix} x_1^1 & \cdots & x_1^{Dim} \\ \vdots & \ddots & \vdots \\ x_{pop}^1 & \cdots & x_{pop}^{Dim} \end{bmatrix} \tag{1}$$

where pop denotes the population size and Dim is the dimension of the variable. Each candidate solution is randomly generated between the upper bound (UB) and the lower bound (LB) of the given problem, and the expression of the ith individual X_i is

$$X_i = rand \times (UB - LB) + LB \tag{2}$$

where i is an integer between 1 and pop and $rand$ denotes a random number between 0 and 1. LB and UB are expressed as

$$\begin{aligned} LB &= [lb_1, \ldots, lb_{Dim}] \\ UB &= [ub_1, \ldots, ub_{Dim}] \end{aligned} \tag{3}$$

During initialization, DO regards the individual with the optimal fitness value as the initial elite, which is approximately considered the most suitable position for the dandelion seed to flourish. Taking the minimum value as an example, the mathematical expression of the initial elite X_{elite} is

$$\begin{aligned} f_{best} &= \min\left(f\left(X_i\right)\right) \\ X_{elite} &= X\left(find\left(f_{best} == f\left(X_i\right)\right)\right) \end{aligned} \tag{4}$$

where $find()$ denotes two indexes with equal values.

2.2.2. Rising stage

In the rising stage, dandelion seeds need to reach a certain height before they can float away from their parent. Under the influence of the wind speed, air humidity, etc., dandelion seeds rise to different heights. Here, the weather is divided into the following two situations.

Case 1 On a clear day, wind speeds can be regarded to have a lognormal distribution $\ln Y \sim N\left(\mu, \sigma^2\right)$. Under this distribution, random numbers are more distributed along the Y-axis, which increases the chance for dandelion seeds to travel to far regions. Therefore, DO emphasizes exploration in this case. In the search space, dandelion seeds are blown randomly to various locations by the wind. The rising height of a dandelion seed is determined by the wind speed. The stronger the wind is, the higher the dandelion flies and the farther the seeds scatter. Affected by the wind speed, the vortexes above the dandelion seeds are constantly adjusted to make them rise in a spiral form. The corresponding mathematical expression in this case is

$$X_{t+1} = X_t + \alpha * v_x * v_y * \ln Y * \left(X_s - X_t\right) \tag{5}$$

where X_t represents the position of the dandelion seed during iteration t. X_s represents the randomly selected position in the search space during iteration t. Eq. (6) provides the expression for the randomly generated position.

$$X_s = rand(1, Dim) * (UB - LB) + LB \tag{6}$$

$\ln Y$ denotes a lognormal distribution subject to $\mu = 0$ and $\sigma^2 = 1$, and its mathematical formula is

$$\ln Y = \begin{cases} \dfrac{1}{y\sqrt{2\pi}} \exp\left[-\dfrac{1}{2\sigma^2}(\ln y)^2\right] & y \geq 0 \\ 0 & y < 0 \end{cases} \tag{7}$$

In Eq. (7), y denotes the standard normal distribution $N(0, 1)$. α is an adaptive parameter used to adjust the search step length, and the mathematical expression is

$$\alpha = rand() * \left(\frac{1}{T^2}t^2 - \frac{2}{T}t + 1\right) \tag{8}$$

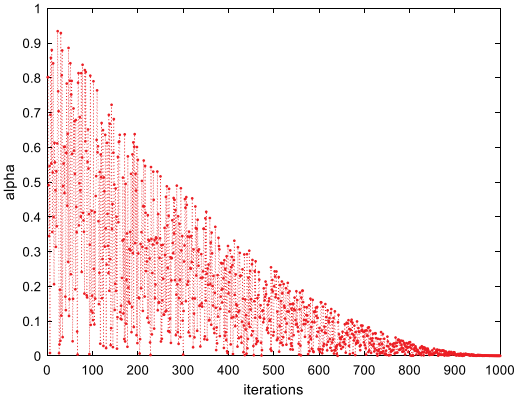


Fig. 3. Dynamic trend of α.

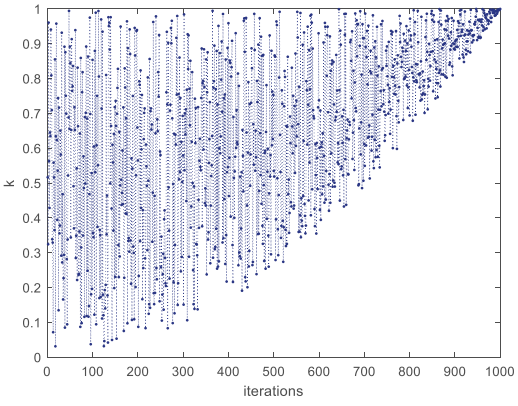


Fig. 4. Dynamic trend of k.

Fig. 3 visualizes the dynamic change in α with the number of iterations. According to Fig. 3, α is a random perturbation between $[0, 1]$ in the process of a nonlinear decrease that approaches 0. Such fluctuations make the algorithm pay much attention to the global search in the early stage and turn to a local search in the later stage, which is beneficial to ensure accurate convergence after a full global search. v_x and v_y represent the lift component coefficients of a dandelion due to the separated eddy action. Eq. (9) is utilized to calculate the force on the variable dimension.

$$\begin{aligned} r &= \frac{1}{e^{\theta}} \\ v_x &= r * \cos\theta \\ v_y &= r * \sin\theta \end{aligned} \tag{9}$$

where θ is a random number between $[-\pi, \pi]$.

Case 2 On a rainy day, dandelion seeds cannot rise appropriately with the wind because of air resistance, humidity and other factors. In this case, dandelion seeds are exploited in their local neighbourhoods, and the corresponding mathematical expression is

$$X_{t+1} = X_t * k \tag{10}$$

where k is used to regulate the local search domain of a dandelion, and Eq. (11) is used to calculate the domain.

$$\begin{aligned} q &= \frac{1}{T^2 - 2T + 1}t^2 - \frac{2}{T^2 - 2T + 1}t + 1 + \frac{1}{T^2 - 2T + 1} \\ k &= 1 - rand() * q \end{aligned} \tag{11}$$

Fig. 4 shows the dynamic wave of k. Clearly, k exhibits a "downwards convex" oscillation, which is conducive to local exploitation of the algorithm with a long stride in the early stage and a small cloth

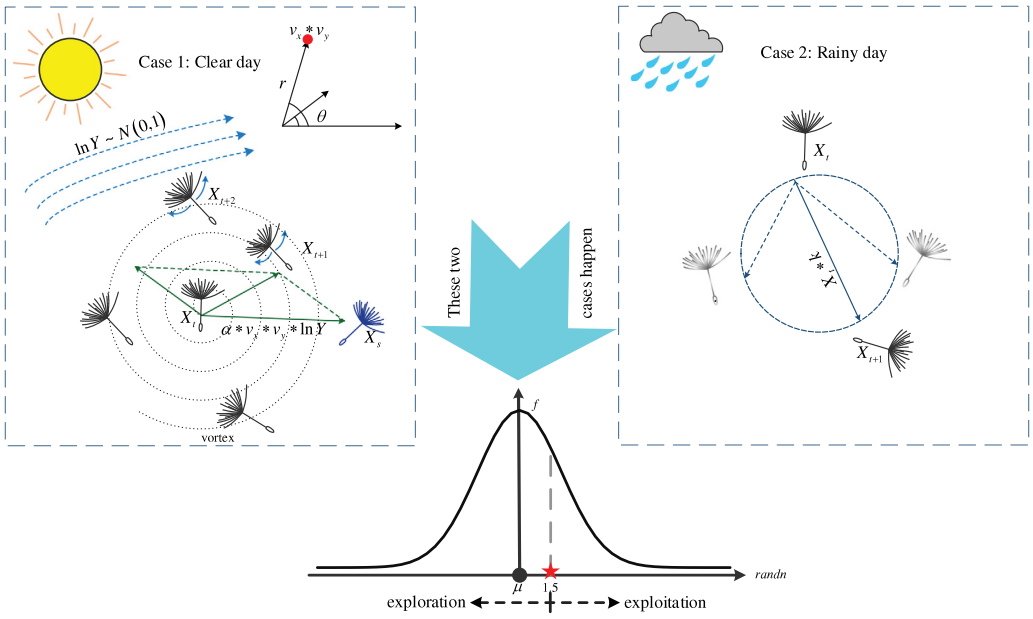


Fig. 5. Schematic diagram of the rising stage of dandelion seeds.

length in the later stage. At the end of the iteration, parameter k gradually approaches 1 to guarantee that the population eventually converges to the optimal search agent.

In conclusion, the mathematical expression of dandelion seeds in the rising stage is

$$X_{t+1} = \begin{cases} X_t + \alpha * v_x * v_y * \ln Y * \left(X_s - X_t\right) & randn < 1.5 \\ X_t * k & else \end{cases} \tag{12}$$

where $randn()$ is the random number that follows the standard normal distribution.

Fig. 5 shows the behaviour of dandelion seeds flying under different weather conditions. The approximate regeneration locations of dandelion seeds are given in the figure. First, when the weather is clear, dandelion seeds are updated based on randomly selected location information to emphasize the exploration process. The eddy above the seed acts on the moving vector by multiplying the x and y components to correct the direction of the dandelion's movement in a spiral. In the second case, dandelion seeds are exploited in all directions in the local community. The normal distribution of random numbers is used to dynamically control exploitation and exploration. To make the algorithm more global search oriented, the cut-off point is set to 1.5. This setting makes dandelion seeds traverse the entire search space as much as possible in the first stage to provide the correct direction for the next stage of iterative optimization.

2.2.3. Descending stage

In this stage, the proposed DO algorithm also emphasizes exploration. Dandelion seeds descend steadily after rising to a certain distance. In DO, Brown motion is used to simulate the moving trajectory of dandelions. It is easy for individuals to traverse more search communities in the process of iterative updating because Brownian motion obeys a normal distribution at each change. To reflect the stability of dandelion descent, the average position information after the rising stage is employed. This facilitates the development of the population as a whole towards promising communities. The corresponding mathematical expression is

$$X_{t+1} = X_t - \alpha * \beta_t * \left(X_{mean\_t} - \alpha * \beta_t * X_t\right) \tag{13}$$

where β_t denotes Brownian motion and is a random number from the standard normal distribution (Einstein, 1956). X_{mean_t} denotes the average position of the population in the ith iteration, and its mathematical expression is

$$X_{mean\_t} = \frac{1}{pop} \sum_{i=1}^{pop} X_i \tag{14}$$

Fig. 6 shows the regeneration process of dandelion seeds during descent. According to Fig. 6, the average position information of the population is essential for the iterative updating of individuals, which directly determines the evolution direction of individuals. The trajectory of Brownian motion, which is based on a global search, is also presented in the figure. The irregular movement causes the search agents to escape the local extremum with a high probability during the iterative update and then pushes the population to seek the region near the global optimum.

2.2.4. Landing stage

In this part, the DO algorithm focuses on exploitation. Based on the first two stages, the dandelion seed randomly chooses where to land. As the iterations gradually progress, the algorithm will hopefully converge to the global optimal solution. Therefore, the obtained optimal solution is the approximate position where dandelion seeds will most easily survive. To accurately converge to the global optimum, search agents borrow the eminent information of the current elite to exploit in their local neighbourhoods. With the evolution of the population, the global optimal solution can eventually be found. This behaviour is expressed in Eq. (15).

$$X_{t+1} = X_{elite} + levy(\lambda) * \alpha * \left(X_{elite} - X_t * \delta\right) \tag{15}$$

where X_{elite} represents the optimal position of the dandelion seed in the ith iteration. $levy(\lambda)$ represents the function of Levy flight and is calculated using Eq. (16) (Mantegna, 1994).

$$Levy(\lambda) = s \times \frac{w \times \sigma}{|t|^{\frac{1}{\beta}}} \tag{16}$$

In Eq. (16), β is a random number between $[0, 2]$ ($\beta = 1.5$ in this paper). s is a fixed constant 0.01. w and t are random numbers between $[0, 1]$. The mathematical expression of σ is

$$\sigma = \left(\frac{\Gamma(1+\beta) \times \sin\left(\frac{\pi\beta}{2}\right)}{\Gamma\left(\frac{1+\beta}{2}\right) \times \beta \times 2^{\left(\frac{\beta-1}{2}\right)}} \right) \tag{17}$$

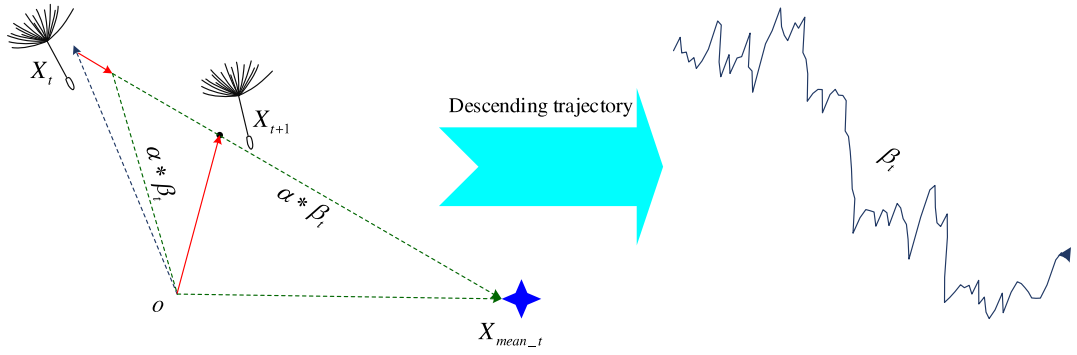


Fig. 6. Schematic diagram of the descending stage of dandelion seeds.

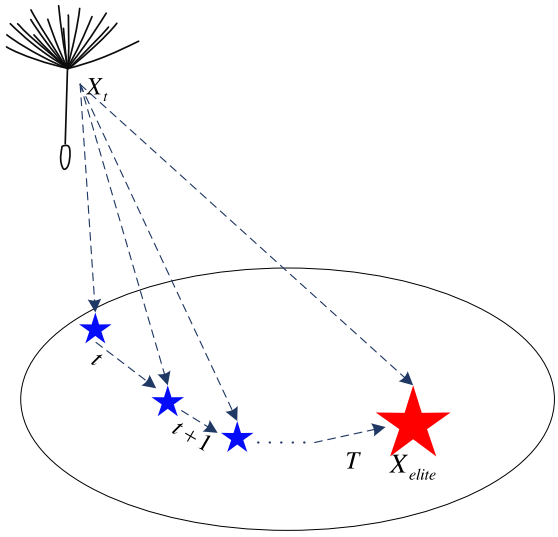


Fig. 7. Schematic diagram of the dandelion seed landing stage.

where β is fixed at 1.5. δ is a linear increasing function between $[0, 2]$ and is calculated by Eq. (18).

$$\delta = \frac{2t}{T} \tag{18}$$

Fig. 7 shows the process of the search agent gradually updating to the global optimal solution in the final phase. To accurately converge to the global optimum, the linear increasing function is applied to individuals to avoid excessive exploitation. In this stage, the Levy flight coefficient is used to simulate the individual movement step size. The reason is that the Levy flight coefficient can be used by agents to stride to other positions with a large probability under a Gaussian distribution, which develops more local search domains with a limited number of iterations.

2.2.5. DO algorithm execution

This subsection completely describes the specific execution flow of DO. For DO, groups of vectors are randomly generated in the search space to initiate the optimization process. Then, dandelion seeds perform iterative optimization through three stages of rising, descending, and landing. It is worth noting the rules for the generation of new populations before the next iteration. Taking the minimum value as an example, each dandelion seed is arranged in ascending order from top to bottom according to the fitness value. The individual with the minimum fitness value is the elite individual of the next generation of the population, and the sorted population is the initial population of the next iteration. This kind of sorting method is beneficial to the inheritance of good information and avoids the algorithm overwriting the accurate direction optimization due to the iteration process. The algorithm ends the optimization process by setting the maximum number of iterations. The pseudocode for the proposed DO algorithm is detailed in Algorithm 1.

Algorithm 1 Dandelion Optimizer

Input: The population size pop, maximum number of iterations T and variable dimension Dim

Output: The optimal dandelion seed X_{best} and its fitness value f_{best}

Initialize dandelion seeds X of DO

Calculate the fitness value f of each dandelion seeds

Select the optimum dandelion seed X_{elite} according to fitness values

while $(t < T)$ **do**

/*Rise stage*/

  if $randn() < 1.5$ **do**

    Generate adaptive parameters using Eq. (8)

    Update dandelion seeds using Eq. (5)

  else if do

    Generate adaptive parameters using Eq. (11)

    Update dandelion seeds using Eq. (10)

  end if

/*Decline stage*/

  Update dandelion seeds using Eq. (13)

/*Land stage*/

  Update dandelion seeds using Eq. (15)

  Arrange dandelion seeds from good to bad according to fitness values

  Update X_{elite}

  if $f(X_{elite}) < f(X_{best})$

    $X_{best} = X_{elite}$, $f_{best} = f(X_{elite})$

  end if

end while

Return X_{best} and f_{best}

2.3. Computational complexity

Computational complexity is an important indicator to measure algorithm efficiency. This subsection analyses the time complexity and space complexity of the proposed DO algorithm.

2.3.1. Time complexity

As with the sine cosine algorithm, the whale optimization algorithm, the moth swarm algorithm, Harris hawks optimization, the seagull optimization algorithm, the chimp optimization algorithm, Levy flight distribution, the horse herd optimization algorithm, and the aquila optimizer, the initialization population of the proposed DO takes $O(pop \times Dim)$ time, where pop represents the population size and Dim

Table 1
Summary of the CEC2017 benchmark functions (Awad et al., 2017).

	No.	Function	Range	F_{best}
Unimodal functions	F_1	Shifted and Rotated Bent Cigar Function	[−100,100]	100
	F_3	Shifted and Rotated Zakharov Function	[−100,100]	300
Simple multimodal functions	F_4	Shifted and Rotated Rosenbrock's Function	[−100,100]	400
	F_5	Shifted and Rotated Rastrigin's Function	[−100,100]	500
	F_6	Shifted and Rotated Expanded Scaffer's F6 Function	[−100,100]	600
	F_7	Shifted and Rotated Lunacek Bi_Rastrigin Function	[−100,100]	700
	F_8	Shifted and Rotated Non-Continuous Rastrigin's Function	[−100,100]	800
	F_9	Shifted and Rotated Levy Function	[−100,100]	900
	F_{10}	Shifted and Rotated Schwefel's Function	[−100,100]	1000
Hybrid functions	F_{11}	Hybrid Function 1 ($N = 3$)	[−100,100]	1100
	F_{12}	Hybrid Function 2 ($N = 3$)	[−100,100]	1200
	F_{13}	Hybrid Function 3 ($N = 3$)	[−100,100]	1300
	F_{14}	Hybrid Function 4 ($N = 4$)	[−100,100]	1400
	F_{15}	Hybrid Function 5 ($N = 4$)	[−100,100]	1500
	F_{16}	Hybrid Function 6 ($N = 4$)	[−100,100]	1600
	F_{17}	Hybrid Function 6 ($N = 5$)	[−100,100]	1700
	F_{18}	Hybrid Function 6 ($N = 5$)	[−100,100]	1800
	F_{19}	Hybrid Function 6 ($N = 5$)	[−100,100]	1900
	F_{20}	Hybrid Function 6 ($N = 6$)	[−100,100]	2000
Composition functions	F_{21}	Composition Function 1 ($N = 3$)	[−100,100]	2100
	F_{22}	Composition Function 2 ($N = 3$)	[−100,100]	2200
	F_{23}	Composition Function 3 ($N = 4$)	[−100,100]	2300
	F_{24}	Composition Function 4 ($N = 4$)	[−100,100]	2400
	F_{25}	Composition Function 5 ($N = 5$)	[−100,100]	2500
	F_{26}	Composition Function 6 ($N = 5$)	[−100,100]	2600
	F_{27}	Composition Function 7 ($N = 6$)	[−100,100]	2700
	F_{28}	Composition Function 8 ($N = 6$)	[−100,100]	2800
	F_{29}	Composition Function 9 ($N = 3$)	[−100,100]	2900
	F_{30}	Composition Function 10 ($N = 3$)	[−100,100]	3000

represents the variable dimension. The calculated fitness value of the population takes $O(pop \times f)$ time, where f is the objective function that defines the problem.

The three stages of rising, descending and landing need a total of $O(T \times pop \times Dim)$ time, where T represents the maximum number of iterations. In each iteration, it takes $O(M)$ time to find the current optimal solution. The total time complexity of DO in the iteration stage is $O(M \times T \times pop \times Dim \times f)$. The total time complexity of the sine cosine algorithm, the whale optimization algorithm, the moth swarm algorithm, Harris hawks optimization, the seagull optimization algorithm, Levy flight distribution, and the aquila optimizer are also $O(M \times T \times pop \times Dim \times f)$ in the iteration stage. The chimp optimization algorithm requires $O(K)$ time to update the population and $O(K \times M \times T \times pop \times Dim \times f)$ time for the entire iteration. The time complexity of horse herd optimization is $O\left(\left(\frac{T \times pop^2}{2} + \frac{T \times D \times pop}{2}\right) \times f \times M\right)$. Hence, the calculation time of the DO algorithm is better than that of the chimp optimization algorithm and horse herd optimization algorithm, and the calculation efficiency of the DO algorithm is equal to that of the other algorithms, such as the sine cosine algorithm.

2.3.2. Space complexity

Initializing the population can be regarded as the maximum amount of space occupied by DO at any time. Therefore, the space complexity of the proposed DO is $O(pop \times Dim)$.

3. Experimental results and discussion

This section describes the DO simulation experiment on 29 benchmark functions. The optimization ability, convergence, scalability, and statistical tests of DO were evaluated. All experiments were run on the same operating system.

3.1. Benchmark functions and comparison algorithms

To verify the local mining, local extremum avoidance, global exploration, and other performance indicators of DO, CEC2017 unconstrained benchmark functions were selected (Awad et al., 2017). These functions are composed of basic functions through translations, rotations, combinations, etc. Table 1 shows the specific information of the CEC2017 suite. F_1 and F_3 are unimodal functions, which are used to test the convergence accuracy of the algorithms. The global optimization ability of the methods is measured on simple multimodal functions $F_4 - F_{10}$ and hybrid functions $F_{11} - F_{20}$. $F_{21} - F_{30}$ are composition functions used to test the local extremum avoidance of the algorithms. Fig. 8 shows the topology of the two-dimensional version of some of the benchmark functions.

The proposed DO algorithm was compared with 9 well-known algorithms, namely, the sine cosine algorithm (Mirjalili, 2016), the whale optimization algorithm (Mirjalili and Lewis, 2016), the moth swarm algorithm (Mohamed et al., 2017), Harris hawks optimization (Heidari et al., 2019), the seagull optimization algorithm (Dhiman and Kumar, 2019), the chimp optimization algorithm (Khishe and Mosavi, 2020), Levy flight distribution (Houssein et al., 2020), the horse herd optimization algorithm (MiarNaeimi et al. and the aquila optimizer (Abualigah et al., 2021).

Table 2 shows the parameter settings of all algorithms, among which the comparison algorithm parameters are set according to the original literature. To ensure a fair competition among the algorithms, each function was implemented independently 30 times. The mean fitness value (mean) and standard deviation (std) of 30 results were used as the final statistical indexes.

3.2. Qualitative analysis

To clearly observe the optimization behaviour of DO during the iterative process, this subsection conducts a qualitative analysis of the proposed DO algorithm. The experimental dimension is $Dim = 10$, the population size $pop = 30$, and the maximum number of iterations is $T = 1000$. Fig. 9 shows the qualitative measurement results of DO on the CEC2017 benchmark functions. The selected functions include two unimodal, three multimodal and four composition functions. In Fig. 9, the first column depicts the topological structure of the functions in the two-dimensional version. In addition, the last four columns represent the search history on the first two dimensions of an individual, the

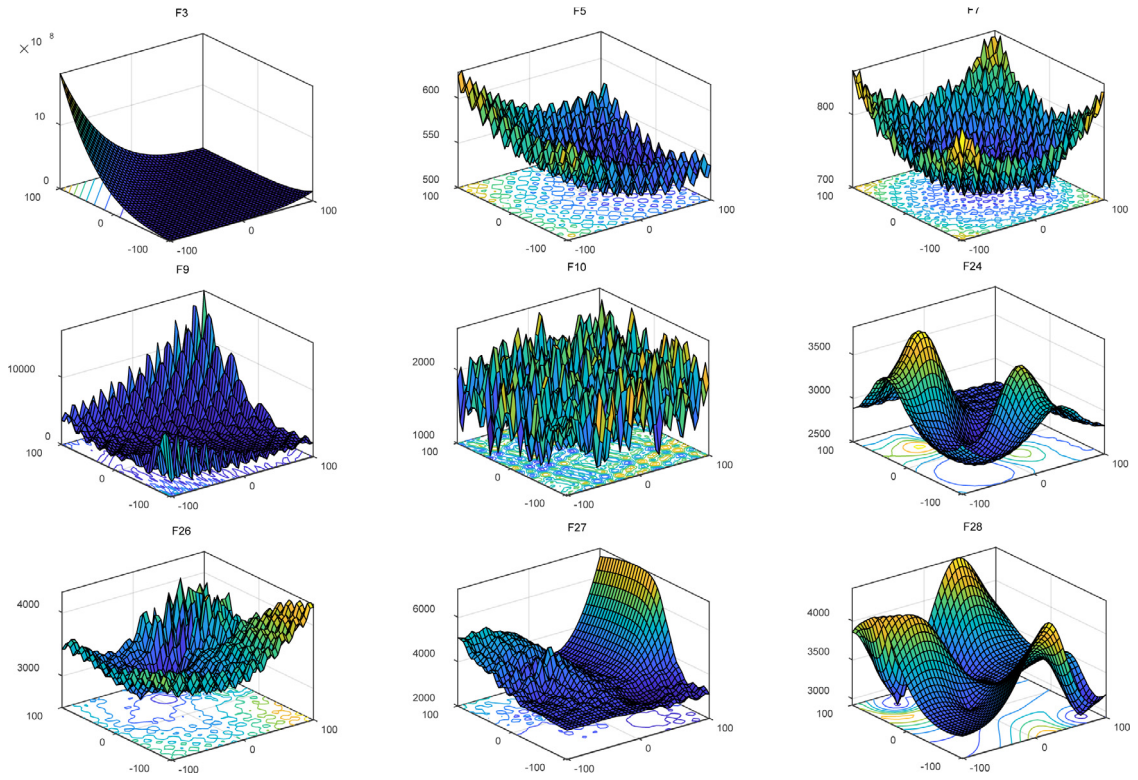


Fig. 8. Topological structures of the benchmark functions in the two-dimensional version.

Table 2
Parameter settings of the algorithms.

Algorithm name	Parameters	Value
All algorithms	Population size Maximum iterations	60 1000
SCA (Mirjalili, 2016)	Number of elites α	2
WOA (Mirjalili and Lewis, 2016)	α α_2	$[2, 0]$ $[-2, -1]$
MSA (Mohamed et al., 2017)	Number of Pathfinders	12
HHO (Heidari et al., 2019)	E_0	$[-1, 1]$
SOA (Dhiman and Kumar, 2019)	Control parameter (A) f_c	$[2, 0]$ 2
ChOA (Khishe and Mosavi, 2020)	r_1, r_2 m	Random Chaotic
LFD (Houssein et al., 2020)	$Threshold$ CSV β α_1 α_2 α_3 ∂_1 ∂_2	2 0.5 1.5 10 0.00005 0.005 0.9 0.1
HOA (MiarNaeimi et al., 2021)	h_β, h_γ s_β, s_γ i_γ d_α, d_β, d_γ r_δ, r_γ	0.9,0.5 0.2,0.1 0.3 0.5,0.2,0.1 0.1,0.05
DO	α k	$[0, 1]$ $[0, 1]$

search trajectory on the first dimension of the search agents, the average fitness value of the population, and the convergence curve.

The search history paints the points that dandelions pass through during the iterations to find the global optimal solution. It can be seen from the second column in the figure that DO searches globally. The red "dots" represent the best solution that DO gains at the preset maximum number of iterations. Numerous search agents are around the optimal solution, indicating that DO is more inclined to exploit promising solutions. Compared with the unimodal functions, the search agents are more scattered on the multimodal functions and the composition functions, which reflects the balance of DO among numerous local optimal solutions.

To visualize the dandelion seed behaviour, the iterative trajectory in the first variable is approximatively regarded as the movement trajectory of a search agent. From the third column in the figure, it can be seen that the curve exhibits a large oscillation state in the early stage of iteration. Whereafter, the curve gradually flattens out in the middle and late iterations. It is worth considering that the oscillation durations of the multimodal functions or the composition functions are longer than those of the unimodal functions. The reason for this phenomenon is that a function has many local extreme values, and it is more difficult to search for an optimal solution. Under the action of adaptive parameter α and Brownian motion, DO is more inclined to explore. At the end of the iterations, exploitation is performed to ensure that DO converges to a higher precision. In addition, the figure clearly reveals the switching time of DO from exploration to exploitation.

The average fitness value denotes the average target optimal value of all dimensions in each iteration and reflects the average tendency of population evolution. As seen from column 4 in Fig. 9, the mean fitness value has a distinct and frequent oscillation in the early iterations, while the oscillation gradually weakens and tends to be gentle in the late iterations. This process indicates that DO fully explores in the early stage and precisely exploits in the later stage. This implies the transition occurs from the rising stage to the descending stage of dandelion seed migration.

The convergence curve shows the optimal behaviour of a dandelion seed to obtain the optimal solution thus far. On unimodal functions, the curve drops rapidly at the beginning of the iteration, and later, the precision is refined. Different from the unimodal function, the curve on the multimodal function descends step by step, which is caused by jumping out local optimal solutions and gradually searching near the global optimal solution.

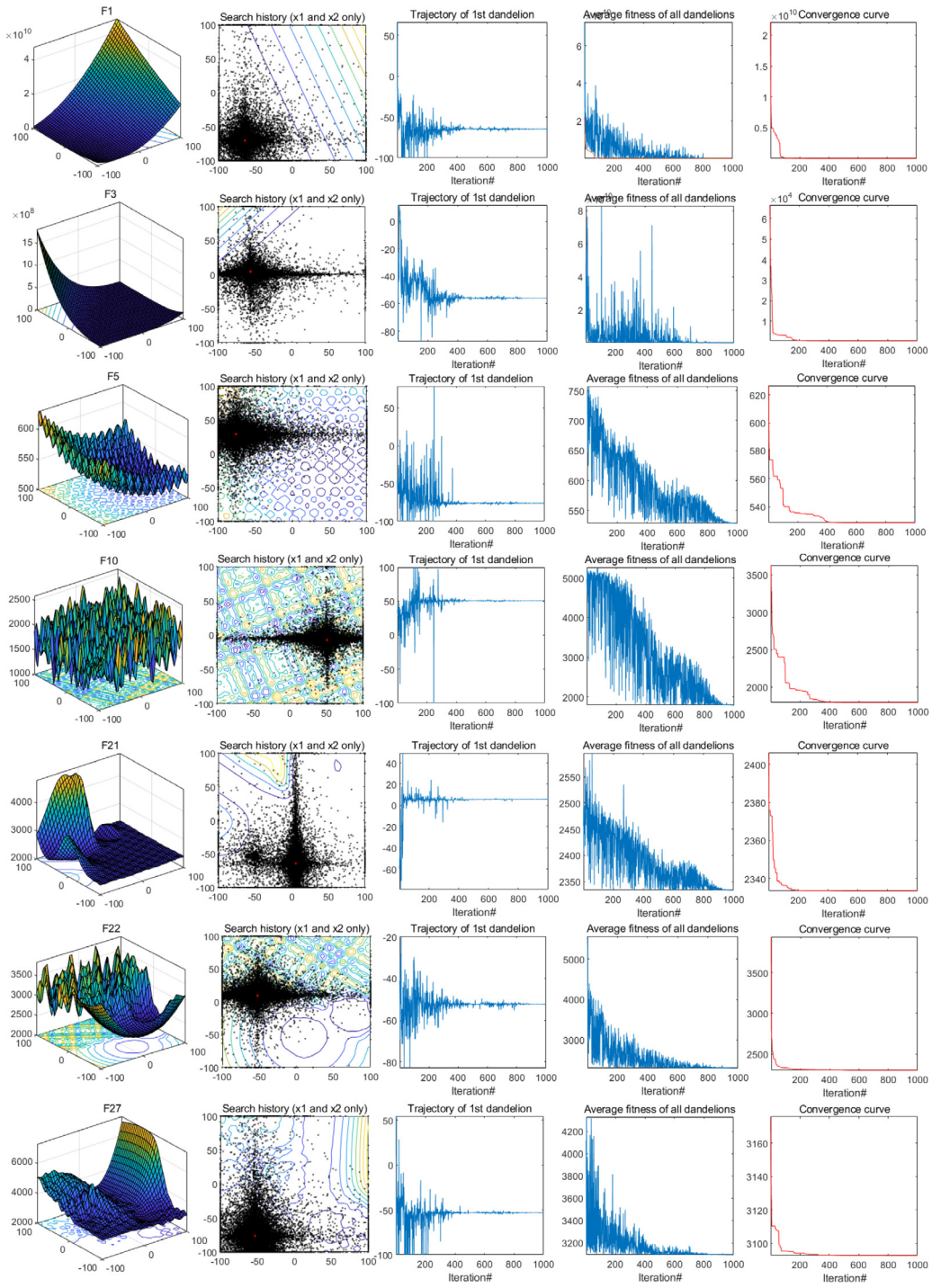


Fig. 9. Qualitative analysis results of DO for unimodal, multimodal, and composition functions.

3.3. Statistical results

To evaluate the general optimization performance of DO, this section tests the proposed DO algorithm and 9 comparison algorithms on 50-dimensional CEC'17 functions. The statistical results are shown in Table 3. Considering the results of the different algorithms on the 50-dimensional CEC'17 functions, DO obtained better results in 33/58 indicators, among which better mean fitness values were obtained for 22/29 functions. For unimodal functions F_1 and F_3, DO shows superior optimization accuracy and stability than the other algorithms. DO has the best optimization effect on the simple multimodal functions of F_4, F_5, F_6, F_7, F_8, and F_{10}. On function F_4, DO obtains the minimum std indicator. In addition, on function F_9, the optimization result of DO is second only to that of MSA and better than those of the other algorithms. For the F_{11}, F_{12}, F_{13}, F_{15}, F_{16}, F_{17}, and F_{20} hybrid functions, DO has excellent exploration performance. On the F_{14}, F_{18} and F_{19} functions, DO achieves better results than the other algorithms except MSA. DO has better std indexes on most complex hybrid functions. Among the composition functions, DO outperforms the other algorithms on the F_{21}, F_{22}, F_{23}, F_{25}, F_{27}, F_{28}, and F_{29} functions. In addition to the above composition functions, DO acquires suboptimal results on the other composition functions. In summary, DO shows accurate exploitation performance and superior exploration performance on the unimodal

Table 3
Statistical results of the different algorithms on the 50-dimensional CEC'17 functions. (The best result is marked in bold, and italics indicate that the algorithm achieves suboptimal results.)

Fun.		SCA	WOA	MSA	HHO	ChOA	SOA	LFD	HOA	AO	DO
F_1	mean	5.38E+10	2.21E+09	1.71E+06	1.06E+08	5.08E+10	3.62E+10	3.07E+08	1.92E+10	1.13E+09	**1.29E+06**
	std	6.97E+09	9.40E+08	1.06E+06	1.54E+07	5.66E+09	6.97E+09	1.89E+08	2.97E+09	4.03E+08	**8.42E+05**
F_3	mean	1.48E+05	2.13E+05	8.35E+04	8.70E+04	1.90E+05	1.19E+05	2.93E+05	1.54E+05	1.87E+05	**2.78E+04**
	std	2.07E+04	6.97E+04	2.34E+04	1.42E+04	1.49E+04	1.83E+04	2.79E+04	1.82E+04	3.57E+04	**9.93E+03**
F_4	mean	9.88E+03	1.38E+03	6.33E+02	7.29E+02	1.14E+04	4.00E+03	1.40E+03	4.36E+03	9.52E+02	**5.64E+02**
	std	2.37E+03	2.83E+02	6.61E+01	1.03E+02	1.56E+03	1.48E+03	1.21E+03	5.39E+02	1.48E+02	**5.69E+01**
F_5	mean	1.10E+03	1.01E+03	8.59E+02	8.91E+02	1.08E+03	9.31E+02	8.64E+02	1.10E+03	8.56E+02	**8.29E+02**
	std	3.89E+01	7.08E+01	7.02E+01	**2.66E+01**	3.07E+01	4.99E+01	3.03E+01	3.20E+01	3.67E+01	6.58E+01
F_6	mean	6.77E+02	6.87E+02	6.55E+02	6.73E+02	6.81E+02	6.61E+02	6.68E+02	6.87E+02	6.64E+02	**6.51E+02**
	std	6.04E+00	1.03E+01	6.59E+00	**4.89E+00**	5.60E+00	8.49E+00	1.18E+01	5.29E+00	8.68E+00	8.60E+00
F_7	mean	1.74E+03	1.78E+03	1.55E+03	1.83E+03	1.72E+03	1.55E+03	1.69E+03	1.53E+03	1.52E+03	**1.35E+03**
	std	9.12E+01	1.11E+02	1.20E+02	8.54E+01	8.86E+01	7.74E+01	8.47E+01	**4.72E+01**	1.17E+02	1.10E+02
F_8	mean	1.41E+03	1.29E+03	1.15E+03	1.18E+03	1.35E+03	1.24E+03	1.18E+03	1.41E+03	1.17E+03	**1.13E+03**
	std	3.70E+01	6.84E+01	6.95E+01	3.00E+01	3.26E+01	4.97E+01	4.26E+01	**2.89E+01**	3.86E+01	5.70E+01
F_9	mean	2.77E+04	2.86E+04	**1.25E+04**	2.54E+04	3.06E+04	2.07E+04	1.69E+04	2.89E+04	2.22E+04	*1.46E+04*
	std	3.43E+03	9.32E+03	3.05E+03	2.92E+03	**2.91E+03**	4.52E+03	4.96E+03	3.07E+03	3.59E+03	3.92E+03
F_{10}	mean	1.51E+04	1.14E+04	8.84E+03	9.18E+03	1.50E+04	1.22E+04	1.03E+04	1.51E+04	8.90E+03	**7.98E+03**
	std	**3.82E+02**	1.42E+03	1.26E+03	1.19E+03	4.29E+02	1.11E+03	1.38E+03	6.34E+02	1.23E+03	7.81E+02
F_{11}	mean	9.03E+03	2.93E+03	1.42E+03	1.54E+03	1.09E+04	6.58E+03	2.31E+03	9.32E+03	2.13E+03	**1.33E+03**
	std	2.02E+03	4.37E+02	9.39E+01	8.06E+01	1.68E+03	2.40E+03	8.40E+02	3.49E+03	2.79E+02	**5.77E+01**
F_{12}	mean	1.59E+10	8.32E+08	2.79E+07	1.47E+08	2.80E+10	5.68E+09	1.19E+09	5.83E+09	3.75E+08	**2.76E+07**
	std	3.32E+09	3.96E+08	1.50E+07	9.64E+07	8.16E+09	3.29E+09	3.27E+09	1.09E+09	2.75E+08	**1.41E+07**
F_{13}	mean	4.35E+09	2.53E+07	2.18E+05	3.86E+06	1.43E+10	1.32E+09	4.62E+07	1.36E+09	7.51E+06	**1.07E+05**
	std	1.33E+09	2.17E+07	1.33E+05	3.32E+06	7.60E+09	1.65E+09	2.49E+08	3.59E+08	5.62E+06	**6.24E+04**
F_{14}	mean	4.65E+06	2.91E+06	**1.65E+05**	1.45E+06	2.85E+06	1.45E+06	5.50E+06	3.11E+06	3.37E+06	*2.34E+05*
	std	2.62E+06	1.79E+06	**4.65E+04**	1.79E+06	8.33E+05	1.41E+06	4.93E+06	1.41E+06	2.97E+06	*1.19E+05*
F_{15}	mean	6.73E+08	1.19E+06	6.63E+04	5.47E+05	5.74E+08	9.10E+07	2.36E+08	3.46E+08	5.70E+05	**5.12E+04**
	std	3.96E+08	2.05E+06	4.30E+04	2.30E+05	1.29E+09	2.27E+08	5.25E+08	9.53E+07	2.36E+05	**2.78E+04**
F_{16}	mean	5.93E+03	5.69E+03	3.94E+03	4.26E+03	5.71E+03	4.05E+03	5.16E+03	6.09E+03	4.23E+03	**3.78E+03**
	std	3.75E+02	7.10E+02	4.69E+02	5.30E+02	3.99E+02	4.52E+02	1.07E+03	3.66E+02	6.32E+02	**3.39E+02**
F_{17}	mean	4.78E+03	4.26E+03	3.89E+03	3.69E+03	5.05E+03	3.61E+03	3.87E+03	4.42E+03	3.68E+03	**3.35E+03**
	std	3.67E+02	5.09E+02	4.36E+02	4.78E+02	5.68E+02	4.05E+02	4.40E+02	**2.21E+02**	4.75E+02	3.35E+02
F_{18}	mean	3.09E+07	2.25E+07	**1.19E+06**	4.97E+06	1.45E+07	6.90E+06	1.94E+07	2.59E+07	7.69E+06	*2.49E+06*
	std	1.59E+07	2.10E+07	**7.73E+05**	6.22E+06	6.71E+06	4.61E+06	1.62E+07	8.31E+06	4.33E+06	*1.04E+06*
F_{19}	mean	4.01E+08	6.51E+06	**6.74E+04**	1.17E+06	8.84E+08	8.74E+07	4.17E+06	1.26E+08	1.95E+06	*1.59E+05*
	std	1.29E+08	5.48E+06	**6.60E+04**	9.75E+05	1.01E+09	1.98E+08	7.81E+06	2.94E+07	1.66E+06	*1.05E+05*
F_{20}	mean	4.03E+03	3.78E+03	3.51E+03	3.43E+03	4.15E+03	3.53E+03	3.69E+03	4.04E+03	3.30E+03	**3.28E+03**
	std	2.54E+02	3.89E+02	3.70E+02	3.80E+02	**2.20E+02**	3.85E+02	3.22E+02	2.65E+02	2.40E+02	3.98E+02
F_{21}	mean	2.92E+03	2.96E+03	2.67E+03	2.87E+03	2.93E+03	2.73E+03	2.89E+03	2.91E+03	2.70E+03	**2.64E+03**
	std	3.99E+01	1.15E+02	6.79E+01	6.74E+01	4.70E+01	5.27E+01	1.04E+02	**3.91E+01**	5.85E+01	6.51E+01
F_{22}	mean	1.67E+04	1.31E+04	1.09E+04	1.13E+04	1.70E+04	1.35E+04	1.21E+04	1.67E+04	1.07E+04	**9.55E+03**
	std	4.48E+02	1.46E+03	1.23E+03	8.23E+02	**3.91E+02**	1.38E+03	1.53E+03	9.09E+02	1.63E+03	*8.15E+02*
F_{23}	mean	3.59E+03	3.70E+03	3.48E+03	3.85E+03	3.58E+03	**3.20E+03**	3.94E+03	4.09E+03	3.43E+03	**3.20E+03**
	std	**5.72E+01**	1.51E+02	1.27E+02	1.60E+02	6.31E+01	5.75E+01	1.85E+02	1.40E+02	1.06E+02	8.19E+01
F_{24}	mean	3.81E+03	3.77E+03	3.62E+03	4.19E+03	6.80E+03	**3.29E+03**	4.21E+03	4.00E+03	3.47E+03	*3.45E+03*
	std	7.50E+01	1.46E+02	1.39E+02	2.28E+02	3.29E+02	**5.60E+01**	2.02E+02	1.24E+02	8.79E+01	*1.19E+02*
F_{25}	mean	7.42E+03	3.58E+03	3.14E+03	3.21E+03	9.46E+03	5.52E+03	3.40E+03	5.39E+03	3.37E+03	**3.10E+03**
	std	8.15E+02	1.80E+02	4.38E+01	5.57E+01	1.23E+03	6.36E+02	1.91E+02	3.60E+02	1.29E+02	**3.27E+01**
F_{26}	mean	1.29E+04	1.39E+04	1.15E+04	1.07E+04	1.15E+04	8.32E+03	1.17E+04	8.70E+03	**7.93E+03**	*9.05E+03*
	std	6.26E+02	1.72E+03	1.43E+03	1.61E+03	5.45E+02	**6.12E+02**	1.79E+03	1.26E+03	2.28E+03	*1.01E+03*
F_{27}	mean	4.66E+03	4.54E+03	4.30E+03	4.41E+03	4.73E+03	3.79E+03	4.22E+03	6.01E+03	4.06E+03	**3.77E+03**
	std	1.62E+02	4.63E+02	3.77E+02	3.89E+02	1.87E+02	**1.43E+02**	8.66E+02	5.21E+02	2.43E+02	1.82E+02
F_{28}	mean	7.86E+03	4.46E+03	3.42E+03	3.59E+03	6.52E+03	8.53E+03	4.38E+03	5.73E+03	4.19E+03	**3.35E+03**
	std	8.29E+02	3.71E+02	5.40E+01	6.80E+01	3.70E+02	1.44E+03	1.24E+03	2.07E+02	2.96E+02	**3.07E+01**
F_{29}	mean	7.73E+03	8.47E+03	6.25E+03	6.09E+03	8.63E+03	6.46E+03	8.56E+03	7.51E+03	6.35E+03	**4.77E+03**
	std	6.20E+02	1.28E+03	5.27E+02	6.47E+02	1.63E+03	9.05E+02	2.96E+03	5.47E+02	5.33E+02	**3.36E+02**
F_{30}	mean	9.29E+08	2.09E+08	**7.74E+06**	4.15E+07	1.26E+09	2.74E+08	1.59E+08	4.44E+08	1.05E+08	*1.09E+07*
	std	2.80E+08	9.59E+07	3.62E+06	1.17E+07	1.23E+09	1.36E+08	2.33E+08	6.09E+07	3.49E+07	**2.90E+06**

and multimodal functions and has strong local extremum avoidance on composite functions.

To intuitively present the distribution of the 30 results, Fig. 10 provides the boxplots of the different algorithms on the 50-dimensional CEC'17 functions. If '+' appears outside the upper edge, it represents an extreme situation in which the algorithm runs 30 experiments. In contrast, if it is beyond the lower edge, it means that algorithm adequately exploits and explores in the search space. According to

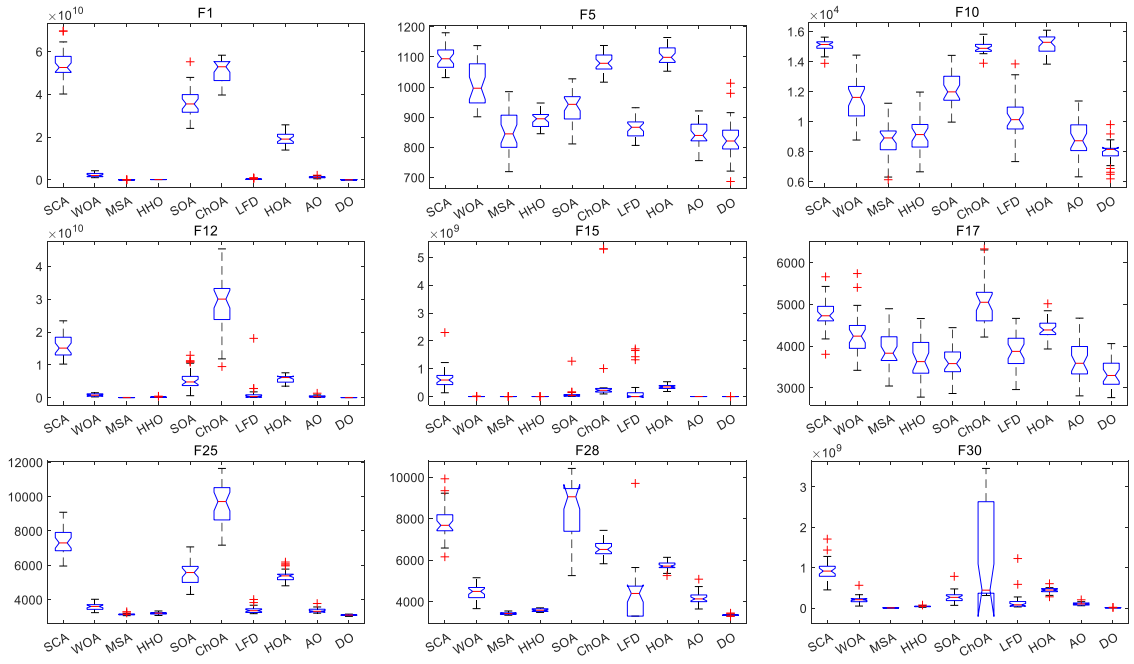


Fig. 10. Boxplots of the different algorithms on the 50-dimensional CEC'17 functions.

Fig. 10, the intermediate results of DO optimization are better than the results of the 9 comparison algorithms. The amplitudes of the upper and lower limits of DO do not change significantly compared to those of the other competitors, which is particularly prominent on the F_{25}, F_{28}, and F_{30} functions. The stability and robustness of DO are verified by these functions. On the F_5 and F_{10} functions, DO accounts for the majority of the upper and lower '+' indicators. This reflects the randomness and uncertainty of DO in solving similar multimodal problems. For other functions, DO has fewer '+' cases, and the difference between the upper bounds and lower bounds is not obvious. This demonstrates that DO still has good optimization results in extreme situations. Based on the above analysis, the boxplot further verifies the effectiveness and strong robustness of DO.

3.4. Scalability analysis

Two aspects mainly affect the complexity of engineering optimization problems: numerous local extrema and higher dimensional variables. This subsection focuses on the performance of DO in high dimensions. Table 4 shows the statistical results of all methods on 100-dimensional CEC'17 functions. According to Table 4, DO achieves better results on 36/58 indicators, among which the best mean fitness value is obtained on 23/30 functions. Comparing the results in Tables 3 and 4, it can be seen that DO shows outstanding optimization performance in the case of high dimensions (i.e., $F_1, F_4, F_5, F_6, F_7, F_8, F_{10}, F_{11}, F_{12}$). Except for F_9 and F_{18}, DO can obtain suboptimal or optimal mean values. On F_{19}, F_{24}, and F_{30} functions, even though the mean accuracy of DO is inferior to that of MSA on the 50-dimensional functions, the mean optimization accuracy of DO is superior to that of MSA on the 100-dimensional functions. In addition, HHO obtains better mean and std indicators than the other competitors on the F_3 function. MSA ranked first and DO ranked second on the F_{13} function. For the case of 100 dimensions, DO continues to maintain the positive optimization performance it experienced in the case of 50 dimensions on the rest of the functions. The statistical results show that DO still has good applicability in high-dimensional optimization problems.

Fig. 11 displays the boxplots of the different algorithms on the 100-dimensional CEC'17 functions. As the dimension increases, function optimization becomes more difficult. Therefore, compared with 50 dimensions, the extreme cases appear more frequently, and the instances of full exploration and exploitation on the functions decrease in the case of 100 dimensions. However, the DO algorithm can still outperform the other algorithms on these functions. On the F_1, F_{12}, F_{15}, F_{25}, F_{28}, and F_{30} functions, the upper and lower limits of DO, MSA, and HHO vary slightly and are similar to the distributions of fitness values in the case of 50 dimensions, as shown in Fig. 9. The boxplot results in the case of 100 dimensions show that the proposed DO algorithm still maintains good local extremum avoidance in high dimensions, which proves that DO is more applicable than the other algorithms.

Fig. 12 shows the log-mean fitness values of the different algorithms in the cases of 10, 30, 50 and 100 dimensions. To show a fair comparison of the scalability of the different algorithms, each function was independently run 30 times under the premise that all methods kept the same population size and maximum iteration number. In Fig. 12, there is no significant change in the mean optimization situation of the algorithm from 10 to 30 dimensions. However, the slope of the DO curve changes more gently than that of the other algorithms at 50 and 100 dimensions. DO shows the best search performance except for some dimensions of the F_{23} function. On the F_{23} function, although the optimization performance of DO is slightly lower than that of MSA at 30 and 50 dimensions, it is still better than that of MSA at 100 dimensions. This test shows that DO can synchronously track the difficulty of optimization caused by an increase in function dimension and obtain better convergence accuracy.

3.5. Convergence analysis

Convergence analysis is conducted on DO and 9 comparison algorithms to reveal the exploration and exploitation of the different algorithms. Fig. 13 describes the convergence curves of the CEC2017 benchmark functions of the 10 algorithms at 100 dimensions. The selected benchmark functions include unimodal, multimodal, hybrid and composition functions to reflect the changing tendency with iterative optimization of the algorithms on different functions.

According to Fig. 13, the dynamic iteration process of different algorithms on each test function is varied. On the unimodal function F_1, the convergence speed of the whale optimization algorithm is slightly faster than that of DO in the early stage of iteration, but as the iterations gradually progress, DO converges faster than the other algorithms and

Table 4
Statistical results of the different algorithms on the 100-dimensional CEC'17 functions. (The best result is marked in bold, and italics indicate that the algorithm achieves suboptimal results.)

Fun.		SCA	WOA	MSA	HHO	SOA	ChOA	LFD	HOA	AO	DO
F_1	mean	1.84E+11	3.06E+10	3.47E+07	1.96E+09	1.40E+11	1.70E+11	1.12E+10	8.20E+10	1.87E+10	**3.01E+07**
	std	1.11E+10	4.94E+09	1.13E+07	3.70E+08	1.40E+10	1.04E+10	3.25E+09	9.55E+09	3.97E+09	**1.05E+07**
F_3	mean	4.33E+05	8.70E+05	4.78E+05	**2.72E+05**	3.27E+05	5.02E+05	7.12E+05	3.73E+05	3.30E+05	*2.95E+05*
	std	5.51E+04	1.35E+05	8.76E+04	**2.05E+04**	2.82E+04	9.50E+04	7.46E+04	3.40E+04	1.27E+04	4.98E+04
F_4	mean	3.91E+04	5.74E+03	1.08E+03	1.59E+03	1.80E+04	3.73E+04	3.48E+03	1.64E+04	3.79E+03	**8.73E+02**
	std	5.22E+03	1.29E+03	9.05E+01	2.34E+02	3.60E+03	6.81E+03	8.36E+02	2.07E+03	7.34E+02	**6.41E+01**
F_5	mean	1.98E+03	1.74E+03	1.37E+03	1.55E+03	1.62E+03	1.89E+03	1.39E+03	1.95E+03	1.51E+03	**1.29E+03**
	std	6.57E+01	9.92E+01	1.32E+02	**3.72E+01**	9.84E+01	4.86E+01	6.95E+01	7.29E+01	6.16E+01	1.11E+02
F_6	mean	6.98E+02	6.97E+02	6.70E+02	6.84E+02	6.82E+02	6.96E+02	6.72E+02	7.02E+02	6.82E+02	**6.65E+02**
	std	4.60E+00	9.44E+00	6.47E+00	**3.11E+00**	5.63E+00	4.68E+00	8.17E+00	4.83E+00	4.06E+00	7.74E+00
F_7	mean	3.79E+03	3.57E+03	3.06E+03	3.70E+03	3.26E+03	3.54E+03	3.31E+03	3.38E+03	3.15E+03	**2.66E+03**
	std	1.63E+02	1.92E+02	1.32E+02	1.14E+02	1.60E+02	**1.08E+02**	1.27E+02	2.54E+02	1.83E+02	2.30E+02
F_8	mean	2.34E+03	2.19E+03	1.76E+03	1.99E+03	1.97E+03	2.24E+03	1.84E+03	2.34E+03	1.94E+03	**1.64E+03**
	std	**6.18E+01**	1.18E+02	1.33E+02	6.55E+01	7.48E+01	4.39E+01	6.36E+01	7.58E+01	9.23E+01	7.75E+01
F_9	mean	8.15E+04	6.41E+04	**3.03E+04**	5.67E+04	5.62E+04	7.42E+04	3.37E+04	7.32E+04	5.58E+04	3.82E+04
	std	8.57E+03	1.64E+04	5.97E+03	4.72E+03	8.18E+03	4.79E+03	**4.34E+03**	5.76E+03	6.39E+03	9.09E+03
F_{10}	mean	3.25E+04	2.62E+04	2.07E+04	2.19E+04	2.75E+04	3.23E+04	2.32E+04	3.24E+04	2.11E+04	**1.67E+04**
	std	5.59E+02	2.23E+03	2.81E+03	1.67E+03	2.14E+03	**4.79E+02**	3.69E+03	6.84E+02	2.04E+03	1.63E+03
F_{11}	mean	1.21E+05	1.60E+05	1.21E+04	3.21E+04	8.61E+04	1.42E+05	2.78E+05	1.08E+05	1.76E+05	**3.30E+03**
	std	1.80E+04	5.97E+04	3.66E+03	1.23E+04	1.77E+04	1.60E+04	3.66E+04	1.25E+04	3.27E+04	**4.72E+02**
F_{12}	mean	8.06E+10	5.17E+09	2.56E+08	8.16E+08	4.00E+10	9.14E+10	5.35E+09	3.12E+10	3.41E+09	**2.34E+08**
	std	1.21E+10	1.41E+09	**9.40E+07**	2.77E+08	1.17E+10	1.11E+10	5.14E+09	5.36E+09	9.45E+08	*9.46E+07*
F_{13}	mean	1.32E+10	8.84E+07	**2.47E+05**	1.09E+07	5.74E+09	2.32E+10	9.90E+06	4.44E+09	3.59E+07	*5.20E+05*
	std	2.41E+09	4.21E+07	**3.80E+05**	2.45E+06	2.13E+09	4.46E+09	5.08E+06	6.99E+08	1.53E+07	*2.05E+06*
F_{14}	mean	4.39E+07	1.10E+07	**1.10E+06**	3.96E+06	9.35E+06	1.43E+07	1.57E+07	1.68E+07	1.04E+07	*2.09E+06*
	std	1.79E+07	4.39E+06	**3.50E+05**	1.16E+06	4.62E+06	4.26E+06	1.75E+07	4.80E+06	3.72E+06	*9.01E+05*
F_{15}	mean	4.28E+09	1.41E+07	1.12E+05	3.02E+06	1.90E+09	9.08E+09	1.25E+06	1.31E+09	4.65E+06	**5.12E+04**
	std	1.47E+09	1.43E+07	6.17E+04	2.03E+06	1.14E+09	3.61E+09	5.78E+06	1.99E+08	1.98E+06	**2.13E+04**
F_{16}	mean	1.41E+04	1.38E+04	7.63E+03	7.94E+03	8.85E+03	1.41E+04	9.03E+03	1.39E+04	8.87E+03	**6.52E+03**
	std	7.91E+02	1.76E+03	1.27E+03	9.17E+02	1.18E+03	1.20E+03	1.33E+03	9.50E+02	1.07E+03	**7.74E+02**
F_{17}	mean	2.18E+04	9.32E+03	6.82E+03	6.64E+03	7.83E+03	2.11E+04	8.06E+03	1.06E+04	7.39E+03	**5.61E+03**
	std	1.49E+04	1.49E+03	8.93E+02	7.69E+02	1.99E+03	1.17E+04	4.94E+03	1.02E+03	6.30E+02	**5.79E+02**
F_{18}	mean	8.11E+07	9.87E+06	**1.40E+06**	4.59E+06	9.81E+06	2.41E+07	1.06E+07	2.28E+07	9.02E+06	3.43E+06
	std	3.07E+07	4.33E+06	**4.84E+05**	2.00E+06	5.85E+06	7.70E+06	7.63E+06	8.66E+06	3.59E+06	1.66E+06
F_{19}	mean	3.60E+09	5.65E+07	1.90E+06	1.13E+07	1.41E+09	6.17E+09	9.92E+06	1.30E+09	1.59E+07	**1.00E+06**
	std	9.89E+08	4.42E+07	1.50E+06	4.00E+06	8.56E+08	4.20E+09	7.46E+06	3.09E+08	1.32E+07	**4.89E+05**
F_{20}	mean	7.77E+03	6.69E+03	5.87E+03	5.91E+03	6.28E+03	7.64E+03	6.42E+03	7.59E+03	5.71E+03	**5.61E+03**
	std	**2.65E+02**	6.87E+02	5.59E+02	5.15E+02	8.25E+02	4.69E+02	6.06E+02	3.41E+02	3.81E+02	5.50E+02
F_{21}	mean	4.07E+03	4.25E+03	3.56E+03	4.13E+03	3.56E+03	4.20E+03	4.22E+03	4.06E+03	3.82E+03	**3.27E+03**
	std	**8.12E+01**	2.25E+02	1.59E+02	1.71E+02	1.03E+02	1.11E+02	2.15E+02	1.14E+02	2.09E+02	1.12E+02
F_{22}	mean	3.48E+04	2.92E+04	2.32E+04	2.53E+04	2.97E+04	3.49E+04	2.65E+04	3.51E+04	2.46E+04	**1.96E+04**
	std	**4.79E+02**	1.93E+03	3.08E+03	1.17E+03	1.58E+03	6.64E+02	3.35E+03	8.20E+02	1.31E+03	1.52E+03
F_{23}	mean	5.07E+03	5.09E+03	4.68E+03	5.34E+03	4.04E+03	5.11E+03	5.94E+03	7.64E+03	4.55E+03	**3.82E+03**
	std	1.34E+02	2.32E+02	3.29E+02	3.66E+02	**1.18E+02**	1.87E+02	3.24E+02	5.16E+02	2.62E+02	1.36E+02
F_{24}	mean	6.97E+03	6.29E+03	6.25E+03	7.15E+03	4.79E+03	6.69E+03	8.87E+03	7.29E+03	5.72E+03	**4.76E+03**
	std	2.46E+02	5.30E+02	4.44E+02	5.05E+02	1.76E+02	3.00E+02	6.48E+02	4.40E+02	3.91E+02	**1.91E+02**
F_{25}	mean	1.88E+04	5.98E+03	3.71E+03	4.13E+03	1.31E+04	1.55E+04	5.07E+03	1.09E+04	5.18E+03	**3.54E+03**
	std	2.50E+03	4.19E+02	7.29E+01	1.38E+02	1.58E+03	1.11E+03	7.26E+02	6.85E+02	3.49E+02	**6.71E+01**
F_{26}	mean	3.76E+04	3.53E+04	3.07E+04	2.75E+04	**2.08E+04**	2.83E+04	3.10E+04	2.79E+04	2.65E+04	*2.18E+04*
	std	1.85E+03	3.33E+03	3.99E+03	1.37E+03	**1.48E+03**	1.13E+03	5.55E+03	2.36E+03	3.99E+03	*2.20E+03*
F_{27}	mean	7.85E+03	5.41E+03	5.34E+03	4.96E+03	4.56E+03	6.75E+03	7.72E+03	8.36E+03	5.19E+03	**4.10E+03**
	std	2.96E+02	6.56E+02	6.17E+02	5.04E+02	2.71E+02	3.42E+02	2.72E+03	7.01E+02	3.74E+02	**2.25E+02**
F_{28}	mean	2.39E+04	8.29E+03	3.90E+03	4.54E+03	2.31E+04	1.51E+04	7.42E+03	1.45E+04	7.07E+03	**3.62E+03**
	std	2.60E+03	1.09E+03	1.14E+02	3.95E+02	3.84E+03	1.31E+03	2.26E+03	1.19E+03	7.30E+02	**4.33E+01**
F_{29}	mean	2.30E+04	1.68E+04	1.06E+04	1.05E+04	1.40E+04	3.70E+04	1.54E+04	1.51E+04	1.23E+04	**8.15E+03**
	std	6.82E+03	1.97E+03	1.48E+03	7.39E+02	1.61E+03	1.79E+04	7.15E+03	1.12E+03	1.22E+03	**7.02E+02**
F_{30}	mean	9.60E+09	9.80E+08	1.87E+07	8.55E+07	3.52E+09	1.60E+10	7.48E+08	4.65E+09	3.47E+08	**1.55E+07**
	std	1.52E+09	3.72E+08	1.10E+07	3.05E+07	1.91E+09	3.01E+09	1.29E+09	9.79E+08	1.49E+08	**5.24E+06**

continues to exploit the global optimal value to obtain higher convergence accuracy. The reason for this result is that under the effect of Levy flight, search agents are drawn to other communities with larger strides and longer spans. The elite information inherited from each iteration also promotes the population's development towards promising regions. For multimodal functions, the different algorithms show different global search performances, among which DO has the best search performance. On the F_{10}, F_{15}, and F_{20} functions, even though

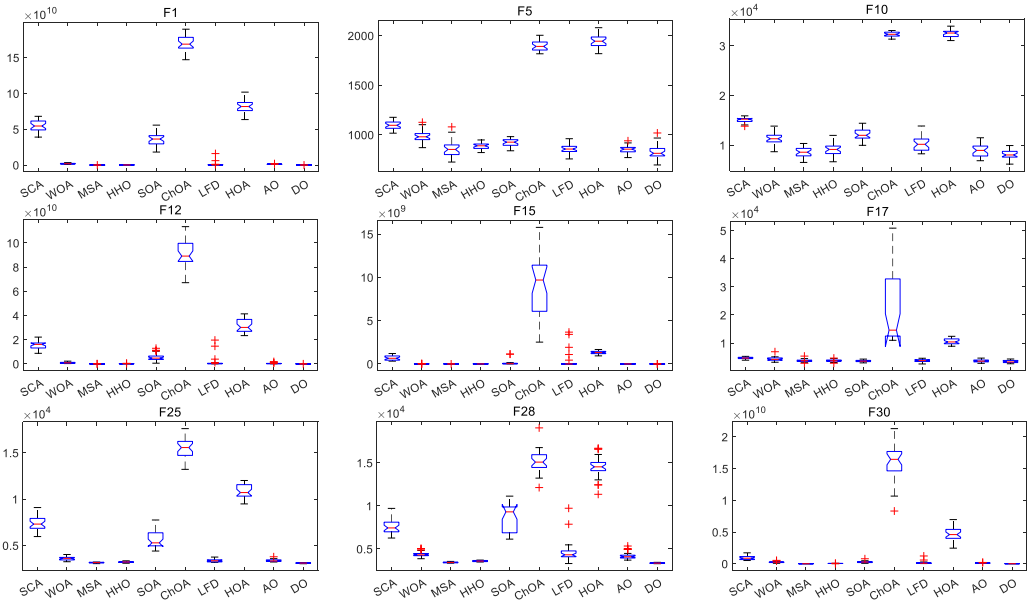


Fig. 11. Boxplots of the different algorithms on the 100-dimensional CEC'17 functions.

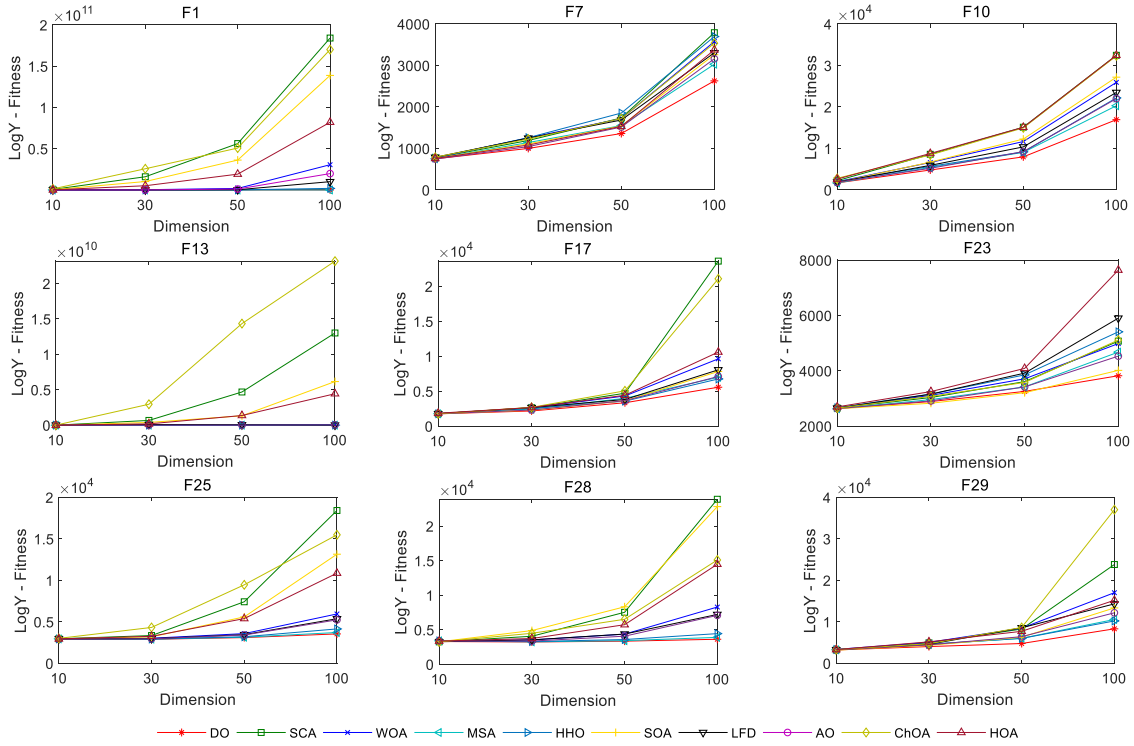


Fig. 12. Scalability results of the different algorithms when dealing with different dimensions.

DO does not converge the fastest during the early iteration stages, it can successfully jump out of local extrema and search near the optimal value as the iterations progress. Nevertheless, other algorithms cannot adequately achieve better accuracy or stop at local extrema without effective improvement. The curves of the composition functions show that DO has strong local extrema avoidance and high convergence accuracy in the case of a complicated function. For example, DO converges to the optimal value and then refines exploitation on the F_{27} and F_{29} functions. The curves of the multimodal and composition functions clearly show the balance between the DO's full exploration of the search space in the first two stages and the precise exploitation of the local neighbourhood in the last stage. The convergence analysis proves that the proposed DO is effective to a certain extent.

3.6. Statistical tests

Because the test results are occasional, a comparison between the algorithms cannot guarantee the superiority and validity completely. Therefore, this subsection uses a variety of statistical tests to demonstrate the statistical superiority of DO. Statistical tests are carried out with the results of the algorithms on the 100-dimensional CEC'17 functions.

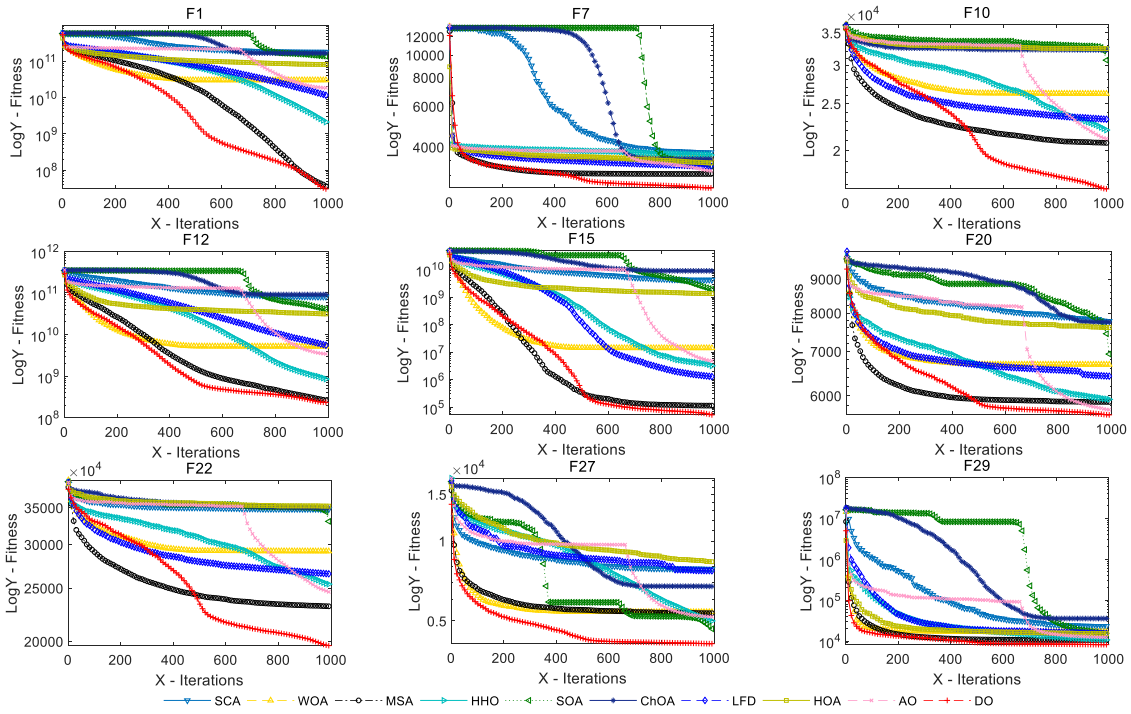


Fig. 13. Convergence curves of the different algorithms on the 100-dimensional CEC'17 functions.

First, the Wilcoxon rank sum test (Wilcoxon, 1992) is performed at a significance level of 5%. DO is taken as the control algorithm and compared with each algorithm in pairs to generate p values. Table 5 shows the statistical test results of the Mann-Whitney U test for the proposed DO algorithm at a significance level of 5%. '+' and '−' indicate that the algorithm can be considered to have significant advantages and no obvious competitiveness in statistical significance, respectively. Among the 261 competitive indicators of DO, 156 have significant superiority. The Wilcoxon rank sum test results show that DO outperforms the other comparison algorithms.

In addition, the Friedman test is a nonparametric method that uses rank implementations to determine whether there is a significant difference between multiple population distributions. The Friedman test shows the overall picture of algorithm performance. Hence, the Friedman test is used to evaluate the comprehensive optimization performance of DO on the 100-dimensional CEC'17 functions. The DO Friedman test results ranked first out of the results of the 10 algorithms in Table 6. The Friedman test results reveal that the exploration and exploitation strategies of DO are effective.

3.7. Discussions of existing works

From the experimental results of Sections 3.3 to 3.6, it can be concluded that the existing methods are usually sensitive to dimensional changes or have poor optimization performance on CEC2017 benchmark functions, and the proposed dandelion algorithm can better improve these deficiencies. The main reasons for such experimental results are analysed below.

SCA moves solutions inward or towards the best search agent with the help of the sines and cosines formula. The heuristic method is simple and fast. However, the optimization results of SCA are not satisfactory in high-dimensional optimization problems. Both WOA and SOA use spiral expressions and move current individuals by borrowing random search agents or global optimal solutions. However, they are easy to fall into local optimum for solving complex optimization problems. This is because the scope of spiral expression used is limited, so the global optimization of search agents in the solution space is not sufficient. MSA shows good performance on the 50-dimensional CEC'17 functions, but poor performance on the 100-dimensional. The optimization performance of AO, HHO, CHOA and HOA on CEC2017 benchmark functions, i.e., F_{12}, F_{19}, F_{30}, are poor. In AO, HHO, and MSA, levy-distributed random numbers are used, which can improve the optimization performance to a certain extent. However, for the complex multimodal functions, since they do not conduct a wider range of the random search, the proposed DO algorithm adopts log-positive distribution of random numbers in the rising stage, which can effectively avoid falling into local optimum. That is why DO can perform better on these functions. For CHOA and HOA, there are fewer random operators acting on different roles, which increase the risk of them falling into local optimum. LFD determines whether to update the current search agent by the threshold between two individuals. According to certain random number, the search agent searches randomly in the global scope or searches in Levy random walk. This setting is helpful for jumping out of local extrema but it takes more time to find two suitable individuals to update. Moreover, when local extremes are close to each other, updating by threshold may not be a better trade-off. Thus, the optimization performance of LFD on CEC'17 F_{12}, F_{13} and F_{14} functions is poor. As a result, the proposed DO algorithm has relatively better optimization capacity.

4. DO for engineering design problems

In this subsection, the optimization efficiency of the proposed DO algorithm in solving real-world optimization problems is evaluated. A total of 4 famous practical engineering problems are selected: the speed reducer design problem, tension/compression spring design problem, welded beam design problem, and pressure vessel design problem. DO is compared with the existing nature-inspired metaheuristic algorithms that are applied to these problems. In general, real-world optimization problems are constrained by equalities or inequalities. To make the algorithm easily address the constraint optimization problem, this paper draws on a simple constraint processing method, namely, the static penalty function. DO uses 250,000 (50×500) as the maximum number of evolutions and is ran independently 30 times on each engineering problem.

Table 5
Wilcoxon rank-sum test results of the different algorithms on the 100-dimensional CEC'17 functions.

Fun.	DO vs. SCA	DO vs. WOA	DO vs. MSA	DO vs. HHO	DO vs. SOA	DO vs. ChOA	DO vs. LFD	DO vs. HOA	DO vs. AO
F_1	3.0E−11 (+)	3.0E−11 (+)	5.9E−02 (−)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)
F_3	4.6E−10 (+)	3.0E−11 (+)	3.5E−10 (+)	2.2E−02 (+)	6.7E−03 (+)	4.1E−11 (+)	3.0E−11 (+)	5.5E−08 (+)	1.2E−03 (+)
F_4	3.0E−11 (+)	3.0E−11 (+)	3.2E−10 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)
F_5	3.0E−11 (+)	3.0E−11 (+)	2.6E−02 (+)	1.8E−10 (+)	1.5E−10 (+)	3.0E−11 (+)	1.0E−04 (+)	3.0E−11 (+)	4.6E−09 (+)
F_6	3.0E−11 (+)	5.0E−11 (+)	6.1E−03 (+)	1.1E−10 (+)	1.7E−09 (+)	3.0E−11 (+)	6.9E−04 (+)	3.0E−11 (+)	6.1E−10 (+)
F_7	3.0E−11 (+)	5.5E−11 (+)	4.6E−09 (+)	3.7E−11 (+)	6.1E−10 (+)	4.1E−11 (+)	3.5E−10 (+)	1.8E−10 (+)	2.2E−09 (+)
F_8	3.0E−11 (+)	3.0E−11 (+)	6.8E−05 (+)	3.3E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	5.6E−10 (+)	3.0E−11 (+)	6.7E−11 (+)
F_9	3.0E−11 (+)	2.7E−09 (+)	3.8E−04 (+)	1.3E−09 (+)	1.6E−08 (+)	3.0E−11 (+)	7.7E−02 (−)	3.0E−11 (+)	3.8E−09 (+)
F_{10}	3.0E−11 (+)	3.0E−11 (+)	3.1E−08 (+)	6.7E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	1.9E−09 (+)	3.0E−11 (+)	5.1E−10 (+)
F_{11}	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)
F_{12}	3.0E−11 (+)	3.0E−11 (+)	3.5E−01 (−)	6.7E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)
F_{13}	3.0E−11 (+)	3.0E−11 (+)	1.6E−05 (+)	1.8E−10 (+)	3.0E−11 (+)	3.0E−11 (+)	1.8E−10 (+)	3.0E−11 (+)	3.0E−11 (+)
F_{14}	3.0E−11 (+)	3.3E−11 (+)	1.4E−06 (+)	7.1E−08 (+)	5.5E−11 (+)	3.0E−11 (+)	4.1E−11 (+)	3.0E−11 (+)	7.4E−11 (+)
F_{15}	3.0E−11 (+)	3.0E−11 (+)	7.1E−08 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	9.8E−08 (+)	3.0E−11 (+)	3.0E−11 (+)
F_{16}	3.0E−11 (+)	3.0E−11 (+)	7.2E−05 (+)	2.2E−07 (+)	2.0E−10 (+)	3.0E−11 (+)	1.2E−10 (+)	3.0E−11 (+)	8.1E−10 (+)
F_{17}	1.2E−12 (+)	1.2E−12 (+)	7.5E−10 (+)	1.9E−07 (+)	3.4E−11 (+)	1.2E−12 (+)	3.4E−11 (+)	1.2E−12 (+)	1.2E−12 (+)
F_{18}	3.0E−11 (+)	1.5E−09 (+)	7.1E−09 (+)	9.1E−03 (+)	1.2E−08 (+)	3.0E−11 (+)	3.6E−08 (+)	3.0E−11 (+)	9.3E−09 (+)
F_{19}	3.0E−11 (+)	3.0E−11 (+)	3.4E−02 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	2.4E−10 (+)	3.0E−11 (+)	3.0E−11 (+)
F_{20}	3.0E−11 (+)	2.0E−07 (+)	6.1E−02 (−)	4.2E−02 (+)	4.5E−04 (+)	5.0E−11 (+)	1.1E−05 (+)	3.0E−11 (+)	6.6E−02 (−)
F_{21}	3.0E−11 (+)	3.0E−11 (+)	1.3E−08 (+)	3.0E−11 (+)	2.6E−10 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.7E−11 (+)
F_{22}	3.0E−11 (+)	3.0E−11 (+)	5.9E−06 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	6.7E−11 (+)	3.0E−11 (+)	4.1E−11 (+)
F_{23}	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	2.2E−07 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)
F_{24}	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	5.7E−01 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	4.1E−11 (+)
F_{25}	3.0E−11 (+)	3.0E−11 (+)	8.9E−10 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)
F_{26}	3.0E−11 (+)	3.0E−11 (+)	1.3E−10 (+)	1.3E−10 (+)	4.1E−02 (+)	6.7E−11 (+)	1.6E−10 (+)	2.4E−10 (+)	1.7E−06 (+)
F_{27}	3.0E−11 (+)	6.1E−11 (+)	6.7E−11 (+)	5.1E−10 (+)	5.1E−08 (+)	3.0E−11 (+)	2.8E−04 (+)	3.0E−11 (+)	4.1E−11 (+)
F_{28}	3.0E−11 (+)	3.0E−11 (+)	3.7E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	9.5E−06 (+)	3.0E−11 (+)	3.0E−11 (+)
F_{29}	3.0E−11 (+)	3.0E−11 (+)	8.9E−10 (+)	9.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	4.2E−10 (+)	3.0E−11 (+)	3.7E−11 (+)
F_{30}	3.0E−11 (+)	3.0E−11 (+)	4.9E−01 (−)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)	3.0E−11 (+)
+/−.	29/0	29/0	25/4	29/0	29/0	29/0	28/1	29/0	28/1

Table 6
Friedman test results of the different algorithms on the 100-dimensional CEC'17 functions.

Algorithm	Friedman rank test	Rank
DO	1.2414	1
SCA	8.9655	10
WOA	6.7931	7
MSA	2.5517	2
HHO	4.1724	3
SOA	5.3448	5
ChOA	8.3793	9
LFD	5.7241	6
HOA	7.6207	8
AO	4.2069	4

4.1. Speed reducer design problem

The objective of the speed reducer problem is to minimize the weight of a mechanical device under 11 constraints, such as the shaft pressure, bearing diameter, surface pressure, and gear bending force. A deceleration of the specific problem is shown in Fig. 14. There are 7 decision variables used to control this problem, namely, the width of flat ground b, gear module m, the number of teeth of pinion p, the length of the first shaft between bearings l_1, the length of the second shaft between bearings l_2, the diameter of the first bearing d_1, and the diameter of the second bearing d_2. The mathematical expression of the problem is as follows.

$$Consider\ \overline{z} = [z_1, z_2, z_3, z_4, z_5, z_6, z_7] = [b, m, p, l_1, l_2, d_1, d_2]$$

$$\min\ f = 0.7854 z_1 z_2^2 (3.3333 z_3^2 + 14.9334 z_3 - 43.0934)$$
$$-1.508 x_1 (z_6^2 + z_7^2) + 7.4777 (z_6^3 + z_7^3) + 0.7854 (z_4 z_6^2 + z_5 z_7^2)$$

$$s.t.\quad g_1(\overline{z}) = \frac{27}{(z_1 z_2^2 z_3)} - 1 \le 0$$

$$g_2(\overline{z}) = \frac{397.5}{(z_1 z_2^2 z_3^2)} - 1 \le 0$$

$$g_3(\overline{z}) = \frac{1.93 z_4^3}{(z_2 z_3 z_6^4)} - 1 \le 0$$

$$g_4(\overline{z}) = \frac{1.93 z_5^3}{(z_2 z_3 z_7^4)} - 1 \le 0$$

$$g_5(\overline{z}) = \frac{1}{(110 z_6^3)} \times \sqrt{\left(\frac{745 z_4}{z_2 z_3}\right)^2 + 16.9 \times 10^6} - 1 \le 0$$

$$g_6(\overline{z}) = \frac{1}{(85 z_7^3)} \times \sqrt{\left(\frac{745 z_5}{z_2 z_3}\right)^2 + 157.5 \times 10^6} - 1 \le 0$$

$$g_7(\overline{z}) = \frac{z_2 z_3}{40} - 1 \le 0$$

$$g_8(\overline{z}) = 5\frac{z_2}{z_1} - 1 \le 0$$

$$g_9(\overline{z}) = \frac{z_1}{12 z_2} - 1 \le 0$$

$$g_{10}(\overline{z}) = \frac{1.5 z_6 + 1.9}{z_4} - 1 \le 0$$

$$g_{11}(\overline{z}) = \frac{1.1 z_7 + 1.9}{z_5} - 1 \le 0$$

$$2.6 \le z_1 \le 3.6, 0.7 \le z_2 \le 0.8, 17 \le z_3 \le 28, 7.3 \le z_4 \le 8.3$$

$$7.3 \le z_5 \le 8.3, 2.9 \le z_6 \le 3.9, 5.0 \le z_7 \le 5.5$$

DO is compared with the harmony search algorithm (Dhiman and Kumar, 2017), particle swarm optimization (Dhiman and Kumar, 2019), the sine cosine algorithm (Dhiman and Kumar, 2017), the grey wolf optimizer (Dhiman and Kumar, 2017), MDE (Kamboj et al., 2020), the spotted hyena optimizer (Dhiman and Kumar, 2017), elephant herding optimization (Hashim et al., 2019), Henry gas solubility optimization (Hashim et al., 2019) and the aquila optimizer (Abualigah et al., 2021). Table 7 shows the results of all algorithms and the best solution, and Table 8 records the statistical results of all algorithms. By combining these two tables, it can be seen that DO achieves

Table 7
Optimal results of the different algorithms on the speed reducer design problem.

Algorithms	b	m	p	l_1	l_2	d_1	d_2	Optimal value
HS (Dhiman and Kumar, 2017)	3.520124	0.7	17	8.37	7.8	3.366970	5.288719	3029.002
PSO (Dhiman and Kumar, 2019)	3.500019	0.7	17	8.3	7.8	3.352412	5.286715	3005.763
SCA (Dhiman and Kumar, 2017)	3.508755	0.7	17	7.3	7.8	3.461020	5.289213	3030.563
GWO (Dhiman and Kumar, 2017)	3.506690	0.7	17	7.380933	7.815726	3.357847	5.286768	3001.288
MDE (Kamboj et al., 2020)	3.50001	0.7	17	7.300156	7.800027	3.350221	5.286685	2996.35669
SHO (Dhiman and Kumar, 2017)	3.50159	0.7	17	7.3	7.8	3.35127	5.28874	2998.5507
EHO (Hashim et al., 2019)	2.900	0.70	17	7.30	7.80	3.10	5.200	3019.01
HGSO (Hashim et al., 2019)	3.498	0.71	17.02	7.67	7.810	3.36	5.289	2997.10
AO (Abualigah et al., 2021)	3.5021	0.7000	17.0000	7.3099	7.7476	3.3641	5.2994	3007.7328
DO	3.5000	0.7000	17.000	7.3057	7.7193	3.3507	5.2867	**2994.7531**

Table 8
Statistical results of the different algorithms on the speed reducer design problem.

Algorithms	Eval	Best	Mean	Worst	Std
HS (Dhiman and Kumar, 2017)	30,000	3029.002	3295.329	3619.465	5.7E+01
PSO (Dhiman and Kumar, 2019)	100,000	3005.763	3105.252	3211.174	7.96E+01
SCA (Dhiman and Kumar, 2017)	30,000	3030.563	3065.917	3104.779	1.81E+01
GWO (Dhiman and Kumar, 2017)	30,000	3001.288	3005.845	3008.752	5.84E+00
MDE (Kamboj et al., 2020)	/	2996.35669	/	/	/
SHO (Dhiman and Kumar, 2017)	30,000	2998.5507	2999.640	3003.889	1.93E+00
EHO (Hashim et al., 2019)	50,000	3019.0124	3100.12	3100.145	2.5262E+01
HGSO (Hashim et al., 2019)	50,000	2997.10	2996.4	2996.9	4.39E−05
AO (Abualigah et al., 2021)	25,000	3007.7328	/	/	/
DO	25,000	**2994.7531**	3000.70	3011.70	4.53E+00

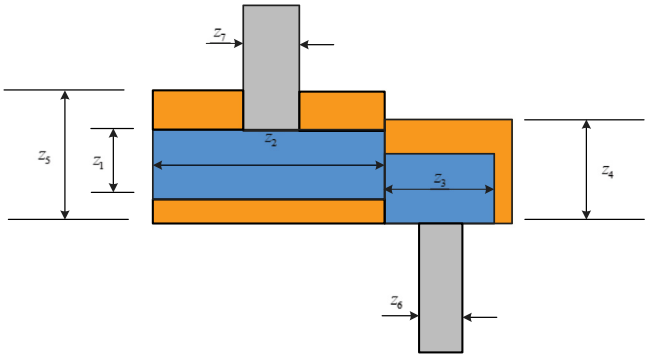


Fig. 14. Speed reducer design.

better results with fewer function evaluations. The best value of DO is obviously better than those of HS, SCA, and EHO with 30 000 evolution times. Compared with suboptimal MDE, DO also shows a relatively significant improvement. In terms of the mean indicator, DO obtains a better effect under the condition of fewer evolutionary iterations. For the std and worst-score indicators, DO performs well compared to the other algorithms and is within acceptable limits. The results show that the algorithm has good applicability to this problem.

4.2. Tension/compression spring design problem

The second engineering problem is the tension/compression spring design problem shown in Fig. 15. The purpose of this problem is to minimize the weight of a spring. This problem is controlled by three-decision variables: wire diameter d, average coil diameter D and the number of active coils N. The mathematical expression of the problem is as follows.

$$\vec{x} = [x_1, x_2, x_3] = [d, D, N]$$

$$\min \quad f(\vec{x}) = (x_3 + 2)x_2{x_1}^2$$

$$s.t. \quad g_1(\vec{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \le 0$$

$$g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \le 0$$

$$g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \le 0$$

$$g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \le 0$$

$$0.05 \le x_1 \le 2.00$$

$$0.25 \le x_2 \le 1.30$$

$$2.00 \le x_3 \le 15.00$$

The nature-inspired metaheuristic algorithms that are used to solve this problem include the harmony search algorithm (Dhiman and Kumar, 2017), particle swarm optimization (Dhiman and Kumar, 2017), the cultural algorithm (Coello Coello and Becerra, 2004), coevolutionary particle swarm optimization (He and Wang, 2007), elephant herding optimization (Hashim et al., 2019), the multiverse optimizer (Dhiman and Kumar, 2017), the sine cosine algorithm (Hashim et al., 2021), the whale optimization algorithm (Hashim et al., 2021), Harris hawks optimization (Hashim et al., 2021), and the Archimedes optimization algorithm (Hashim et al., 2021). DO is compared with these algorithms. Table 9 shows the optimal results obtained by the different algorithms. Table 10 shows the statistical results of all the algorithms. DO achieves a better optimal value with fewer function evaluations than the other algorithms. For the mean indicator, DO is third only to HS and CPSO. For the worst indicator, DO ranks second among all algorithms. Although CPSO has better std results, its evolution time is approximately 10 times that of the DO algorithm. With the increase of in DO evolution iterations, the std results improve. The above results verify that DO has a strong constraint programming ability and can be used to solve such problems.

4.3. Welded beam design

The design cost of welding a beam is minimized as much as possible under the constraints of weld shear stress τ, bending stress σ in the beam, buckling load P_c on the rod and the deflection of the beam end. Fig. 16 is a diagram of a welding beam design. This problem

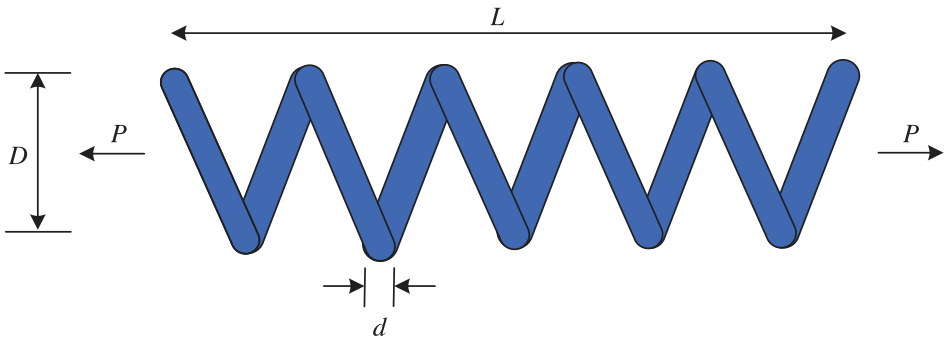


Fig. 15. Tension/compression spring design problem.

Table 9
Optimal results of the different algorithms on the tension/compression spring design problem.

Algorithms	*d*	*D*	*N*	Optimal value
HS (Dhiman and Kumar, 2017)	0.001622	0.316351	15.23960	0.012776352
PSO (Dhiman and Kumar, 2017)	0.05000	0.310414	15.0000	0.013192580
CA (Coello Coello and Becerra, 2004)	0.050000	0.317395	14.031795	0.012721
CPSO (He and Wang, 2007)	0.051728	0.357644	11.244543	0.012674
EHO (Hashim et al., 2019)	0.0580	0.5278	5.5820	0.0135
MVO (Dhiman and Kumar, 2017)	0.05000	0.315956	14.22623	0.012816930
SCA (Hashim et al., 2021)	0.0500	0.3171	14.1417	0.012797
WOA (Hashim et al., 2021)	0.0507	0.3339	12.7645	0.012683
HHO (Hashim et al., 2021)	0.0562	0.4754	6.6670	0.013016
AOA (Hashim et al., 2021)	0.0508	0.3348	11.7020	0.012681
DO	0.051215	0.345416	11.983708	**0.012669**

Table 10
Statistical results of the different algorithms on the tension/compression spring design problem.

Algorithms	Eval	Best	Mean	Worst	Std
HS (Dhiman and Kumar, 2017)	30,000	0.012776352	0.013069872	0.015214230	3.75E−04
PSO (Dhiman and Kumar, 2017)	30,000	0.013192580	0.014817181	0.017862507	2.272E−03
CA (Coello Coello and Becerra, 2004)	50,000	0.012721	0.013568	0.0151156	8.4E−04
CPSO (He and Wang, 2007)	200,000	0.012674	0.012730	0.012924	5.19E−05
EHO (Dhiman and Kumar, 2017)	50,000	0.0135	0.0155	0.0189	1.1E−03
MVO (Dhiman and Kumar, 2017)	30,000	0.012816930	0.014464372	0.017839737	1.622E−03
SCA (Hashim et al., 2021)	30,000	0.012807	0.013859	0.015869	4.30E−04
WOA (Hashim et al., 2021)	30,000	0.012683	0.014709	0.017211	2.30E−03
HHO (Hashim et al., 2021)	30,000	0.013026	0.014160	0.016034	1.64E−03
AOA (Hashim et al., 2021)	30,000	0.012681	0.013369	0.015625	7.44E−04
DO	25,000	**0.012669**	0.013242	0.014830	6.15E−04

mainly optimizes four decision variables: the welding thickness, rod attachment length, rod height, and rod thickness. The mathematical expression of this problem is as follows.

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

$$\min f(\vec{x}) = 1.10471 x_1^2 x_2 + 0.04811 x_3 x_4 (14.0 + x_2)$$

$$s.t. \quad g_1(\vec{x}) = \tau(\vec{x}) - \tau_{\max} \le 0$$

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

$$g_3(\vec{x}) = \delta(\vec{x}) - \delta_{\max} \le 0$$

$$g_4(\vec{x}) = x_1 - x_4 \le 0$$

$$g_5(\vec{x}) = P - P_c(\vec{x}) \le 0$$

$$g_6(\vec{x}) = 0.125 - x_1 \le 0$$

$$g_7(\vec{x}) = 1.10471 x_1^2 + 0.04811 x_3 x_4 (14.0 + x_2) - 5.0 \le 0$$

$$0.1 \le x_1 \le 2.00 \quad 0.1 \le x_2, x_3 \le 10 \quad 0.1 \le x_4 \le 2.00$$

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

$$\tau' = \frac{P}{\sqrt{2} x_1 x_2}, \tau'' = \frac{MR}{J}, M = P\left(L + \frac{x_2}{2}\right)$$

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

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

$$\sigma(\vec{x}) = \frac{6PL}{x_4 x_3^2}, \delta(\vec{x}) = \frac{6PL^3}{E x_3^2 x_4}$$

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

$P = 6000$ lb, $L = 14$ in., $\delta_{\max} = 0.25$ in.

$E = 30 \times 1^6$ psi, $G = 12 \times 10^6$ psi

$\tau_{\max} = 13\,600$ psi, $\sigma_{\max} = 30\,000$ psi

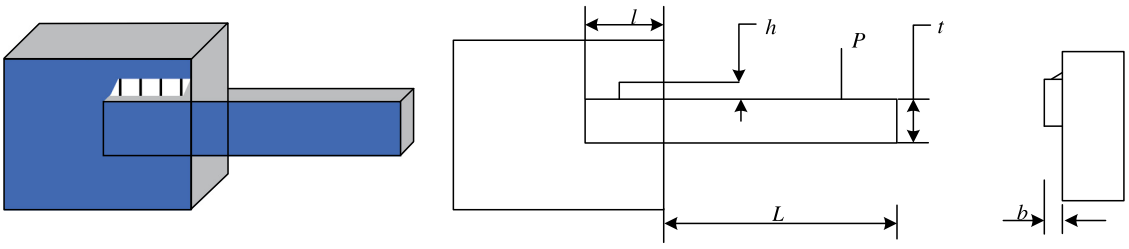


Fig. 16. Welded beam design.

Table 11
Optimal results of the different algorithms on the welded beam design problem.

Algorithms	h	l	t	b	Optimal value
GA (Dhiman and Kumar, 2017)	0.164171	4.032541	10.00000	0.223647	1.873971
PSO (Dhiman and Kumar, 2017)	0.197411	3.315061	10.00000	0.201395	1.820395
CPSO (He and Wang, 2007)	0.2024	3.5442	9.04821	0.2057	1.7280
SCA (Dhiman and Kumar, 2017)	0.204695	3.536291	9.004290	0.210025	1.759173
WOA (Hashim et al., 2019)	0.1876	3.9298	8.9907	0.2308	1.9428
HHO (Hashim et al., 2021)	0.2134	3.5601	8.4629	0.2346	1.8561
HGSO (Hashim et al., 2019)	0.2054	3.4476	9.0269	0.2060	1.7260
DO	0.2061	3.4656	9.0286	0.2061	**1.7249**

Table 12
Statistical results of the different algorithms on the welded beam design problem.

Algorithms	Eval	Best	Mean	Worst	Std
GA (Dhiman and Kumar, 2017)	30,000	1.873971	2.119240	2.320125	3.4820E−02
PSO (Dhiman and Kumar, 2017)	30,000	1.820395	2.230310	3.048231	3.2453E−01
CPSO (He and Wang, 2007)	200,000	1.7280	1.7280	1.782143	1.2926E−02
SCA (Dhiman and Kumar, 2017)	30,000	1.759173	1.817657	1.873408	2.7543E−02
WOA (Hashim et al., 2019)	50,000	1.9428	3.3865	5.9905	8.251E−01
HHO (Hashim et al., 2021)	30,000	1.8561	1.9302	1.9759	6.47E−02
HGSO (Hashim et al., 2019)	50,000	1.7260	1.7265	1.7325	7.66E−03
DO	25,000	**1.7249**	1.7276	1.7456	4.38E−03

Table 11 shows the best results obtained by the proposed DO algorithm, the genetic algorithm (Dhiman and Kumar, 2017), particle swarm optimization (Dhiman and Kumar, 2017), coevolutionary particle swarm optimization (He and Wang, 2007), the sine cosine algorithm (Dhiman and Kumar, 2017), the whale optimization algorithm (Hashim et al., 2019), Harris hawks optimization (Hashim et al., 2021), and Henry gas solubility optimization (Hashim et al., 2019) on the welding beam problem. Table 12 displays the statistical results of the above algorithms. It can be seen that DO performs better than the other algorithms. With fewer function evaluations, DO has the optimal std results, indicating that DO has good robustness. In terms of the mean and worst indicators, DO ranks second only to HGSO and the evolution time of DO is half that of HGSO. Therefore, DO is effective at optimizing the solution.

4.4. Pressure vessel design

The last engineering problem is the pressure vessel design problem, which is shown in Fig. 17. The objective of this problem is to minimize the total cost of materials, shaping, and welding for cylindrical vessels. The problem contains four decision variables, which are the thickness of the shell T_s, the thickness of the head T_h, the radius of entry R, and the length of the cylindrical section L without considering the head. The four constraints and objective function of this problem are mathematically expressed as follows.

$$\vec{x} = [x_1, x_2, x_3, x_4] = [T_s, T_h, R, L]$$

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

$$s.t. \quad g_1\left(\vec{x}\right) = -x_1 + 0.0193x_3 \le 0$$

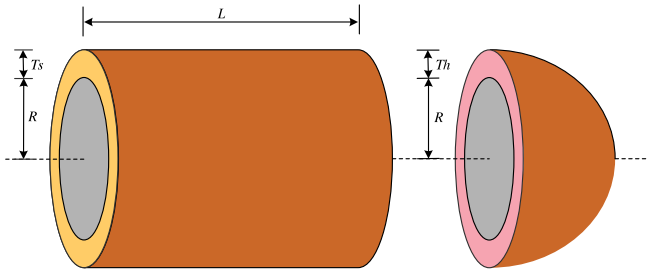


Fig. 17. Pressure vessel design.

$$g_2\left(\vec{x}\right) = -x_3 + 0.00954x_3 \le 0$$

$$g_3\left(\vec{x}\right) = -\pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \le 0$$

$$g_4\left(\vec{x}\right) = x_4 - 240 \le 0$$

$$0 \le x_1 \le 99 \quad 0 \le x_2 \le 99 \quad 10 \le x_3 \le 200 \quad 10 \le x_4 \le 200$$

The algorithms that have been applied to this problem include the harmony search algorithm (Dhiman and Kumar, 2017), particle swarm optimization (Dhiman and Kumar, 2017), coevolutionary particle swarm optimization (He and Wang, 2007), the sine cosine algorithm (Dhiman and Kumar, 2017), the marine predators algorithm (Faramarzi et al., 2020), and the aquila optimizer (Abualigah et al., 2021). The proposed DO algorithm is compared with the other algorithms, and the best results obtained by each algorithm are shown in Table 13. Table 14 displays the statistical results of the 7 algorithms. It can be seen from

Table 13
Optimal results of the different algorithms on the pressure vessel design problem.

Algorithms	T_s	T_h	R	L	Optimal value
HS (Dhiman and Kumar, 2017)	1.099523	0.906579	44.456397	179.65887	6550.0230
PSO (Dhiman and Kumar, 2017)	0.778961	0.384683	40.320913	200.00000	5891.3879
CPSO (He and Wang, 2007)	0.8125	0.4375	42.091266	176.746500	6061.0777
SCA (Dhiman and Kumar, 2017)	0.817577	0.417932	41.74939	183.57270	6137.3724
MPA (Faramarzi et al., 2020)	0.8125	0.4375	42.098445	176.636607	6059.7144
AO (Abualigah et al., 2021)	1.0540	0.182806	59.6219	38.8050	5949.2258
DO	0.7784	0.3848	40.3310	199.8421	**5885.7766**

Table 14
Statistical results of the different algorithms on the pressure vessel design problem.

Algorithms	Eval	Best	Mean	Worst	Std
HS (Dhiman and Kumar, 2017)	30,000	6550.0230	6643.9870	8005.4397	6.5752E+02
PSO (Dhiman and Kumar, 2017)	30,000	5891.3879	6531.5032	7394.5879	5.341E+02
CPSO (He and Wang, 2007)	200,000	6061.0777	6147.1332	6363.8041	8.6455E+01
SCA (Dhiman and Kumar, 2017)	30,000	6137.3724	6326.7606	6512.3541	1.266E+01
MPA (Faramarzi et al., 2020)	25,000	6059.7144	6102.8271	6410.0929	1.0661E+02
AO (Abualigah et al., 2021)	25,000	5949.2258	/	/	/
DO	25,000	**5885.7766**	6374.0396	7318.5197	5.2906E+02

Table 13 that DO with 25,000 evolutions is significantly better than the HS and SCA algorithms with 30,000 evolutions. With the same number of evolutions, DO achieves a better solution than MPA and AO, which were proposed in recent years. Therefore, DO solves this problem at a lower cost than the previous nature-inspired metaheuristic algorithms.

5. Conclusion and future works

This paper proposes a novel swarm intelligence optimization algorithm called dandelion optimization. DO simulates the flight modes of a dandelion seed in the three stages of rising, descending, and landing. CEC2017 unconstrained benchmark functions were used to evaluate the optimization performance of DO, and the performance was compared with the performance of 9 famous comparison algorithms. Finally, DO was applied to 4 real-world problems and compared with many nature-inspired metaheuristic algorithms to verify the constrained programmability performance.

The statistical results demonstrate that DO has strong optimization performance on 50- and 100-dimensional CEC'17 functions. The 100-dimensional convergence curve further proves that the proposed DO algorithm can jump out of local extreme values and refine the exploitation accuracy in the iterative optimization process. The scalability analysis shows that the proposed DO algorithm still has high optimization accuracy for all dimensions. The results of the Wilcoxon rank-sum test and Friedman test verify the effectiveness of DO in terms of statistical significance. Finally, the application results of real-world problems show that DO can replace the previous nature-inspired metaheuristic algorithms and achieve better convergence accuracy with fewer evolutions.

In future works, although DO will achieve satisfactory results, other well-known operators, such as opposition-based learning mechanisms and chaos mapping, can be introduced into DO to better enhance the optimization performance. It is necessary to develop a binary version of DO to solve classification problems. In addition, the multiobjective version of DO can be improved to solve multiobjective optimization problems. Finally, DO can be applied to the hyperparameter optimization of machine learning algorithms, image segmentation and other fields.

CRediT authorship contribution statement

Shijie Zhao: Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing, Software, Visualization, Investigation. **Tianran Zhang:** Conceptualization, Methodology, Formal analysis, Writing – original draft, Writing – review & editing, Formal analysis, Software, Visualization, Investigation. **Shilin Ma:** Writing – review & editing, Visualization, Investigation, Supervision. **Miao Chen:** Writing – review & editing, Visualization, Investigation, Supervision.

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.

Acknowledgements

This work was supported in part by the Education Department of Liaoning Province Fund, China Project (LJ2019JL017), Doctoral Research Start-up Fund of Science and Technology Department of Liaoning Province, China (2019-BS-118).

References

Abualigah, L., Yousri, D., Abd Elaziz, M., Ewees, A.A., Al-qaness, M.A., Gandomi, A.H., 2021. Aquila optimizer: A novel meta-heuristic optimization algorithm. Comput. Ind. Eng. 157, 107250.

Agushaka, J.O., Ezugwu, A.E., Abualigah, L., 2022. Dwarf mongoose optimization algorithm. Comput. Methods Appl. Mech. Engrg. 391, 114570.

Ahmadianfar, I., Heidari, A.A., Gandomi, A.H., Chu, X., Chen, H., 2021. RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method. Expert Syst. Appl. 181, 115079.

Askari, Q., Younas, I., Saeed, M., 2020. Political optimizer: A novel socio-inspired meta-heuristic for global optimization. Knowl.-Based Syst. 195, 105709.

Awad, N.H., Ali, M.Z., Suganthan, P.N., Liang, J.J., Qu, B.Y., 2017. Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization. In: 2017 IEEE Congress on Evolutionary Computation (CEC).

Azizi, M., 2021. Atomic orbital search: A novel metaheuristic algorithm. Appl. Math. Model. 93, 657–683.

Back, T., 1996. Evolutionary Algorithms in Theory and Practice: evolution strategies, evolutionary programming, genetic algorithms. Oxford University press.

Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J., 2009. A survey on metaheuristics for stochastic combinatorial optimization. Nat. Comput. 8 (2), 239–287.

Blum, C., Roli, A., 2003. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. ACM Comput. Surv. (CSUR) 35 (3), 268–308.

Casseau, V., De Croon, G., Izzo, D., Pandolfi, C., 2015. Morphologic and aerodynamic considerations regarding the plumed seeds of Tragopogon pratensis and their implications for seed dispersal. PLoS One 10 (5), e0125040.

Cavieres, L.A., Quiroz, C.L., Molina-Montenegro, M.A., 2008. Facilitation of the non-native taraxacum officinale by native nurse cushion species in the high andes of central Chile: are there differences between nurses? Funct. Ecol. 22 (1), 148–156.

Chan-Ley, M., Olague, G., 2020. Categorization of digitized artworks by media with brain programming. Appl. Opt. 59 (14), 4437–4447.

Chou, J.S., Nguyen, N.M., 2020. FBI inspired meta-optimization. Appl. Soft Comput. 93, 106339.

Coello Coello, C.A., Becerra, R.L., 2004. Efficient evolutionary optimization through the use of a cultural algorithm. Eng. Optim. 36 (2), 219–236.
Cornuéjols, G., 2008. Valid inequalities for mixed integer linear programs. Math. Program. 112 (1), 3–44.
Cummins, C., Seale, M., Macente, A., Certini, D., Mastropaolo, E., Viola, I.M., Nakayama, N., 2018. A separated vortex ring underlies the flight of the dandelion. Nature 562 (7727), 414–418.
Dhiman, G., Kumar, V., 2017. Spotted hyena optimizer: a novel bio-inspired based metaheuristic technique for engineering applications. Adv. Eng. Softw. 114, 48–70.
Dhiman, G., Kumar, V., 2019. Seagull optimization algorithm: Theory and its applications for large-scale industrial engineering problems. Knowl.-Based Syst. 165, 169–196.
Dorigo, M., Blum, C., 2005. Ant colony optimization theory: A survey. Theoret. Comput. Sci. 344 (2–3), 243–278.
Dorigo, M., Stützle, T., 2019. Ant colony optimization: overview and recent advances. In: Handbook of Metaheuristics. pp. 311–351.
Einstein, A., 1956. Investigations on the Theory of the Brownian Movement. Courier Corporation.
Faramarzi, A., Heidarinejad, M., Mirjalili, S., Gandomi, A.H., 2020. Marine predators algorithm: A nature-inspired metaheuristic. Expert Syst. Appl. 152, 113377.
Fogel, D.B., 1998. Artificial Intelligence Through Simulated Evolution. Wiley-IEEE Press, pp. 227–296.
Fonseca, C.M., Fleming, P.J., 1995. An overview of evolutionary algorithms in multiobjective optimization. Evol. Comput. 3 (1), 1–16.
Galli, L., Lin, C.J., 2021. A study on truncated Newton methods for linear classification. IEEE Trans. Neural Netw. Learn..
Gong, C., Han, S., Li, X., Zhao, L., Liu, X., 2018. A new dandelion algorithm and optimization for extreme learning machine. J. Exp. Theor. Artif. Intell. 30 (1), 39–52.
Gupta, S., Abderazek, H., Yıldız, B.S., Yildiz, A.R., Mirjalili, S., Sait, S.M., 2021. Comparison of metaheuristic optimization algorithms for solving constrained mechanical design optimization problems. Expert Syst. Appl. 183, 115351.
Halim, A.H., Ismail, I., Das, S., 2021. Performance assessment of the metaheuristic optimization algorithms: an exhaustive review. Artif. Intell. Rev. 54 (3), 2323–2409.
Hashim, F.A., Houssein, E.H., Mabrouk, M.S., Al-Atabany, W., Mirjalili, S., 2019. Henry gas solubility optimization: A novel physics-based algorithm. Future Gener. Comput. Syst. 101, 646–667.
Hashim, F.A., Hussain, K., Houssein, E.H., Mabrouk, M.S., Al-Atabany, W., 2021. Archimedes optimization algorithm: a new metaheuristic algorithm for solving optimization problems. Appl. Intell. 51 (3), 1531–1551.
Hashim, F.A., Hussien, A.G., 2022. Snake optimizer: A novel meta-heuristic optimization algorithm. Knowl.-Based Syst. 242, 108320.
He, Q., Wang, L., 2007. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng. Appl. Artif. Intell. 20 (1), 89–99.
Heidari, A.A., Mirjalili, S., Faris, H., Aljarah, I., Mafarja, M., Chen, H., 2019. Harris hawks optimization: Algorithm and applications. Future Gener. Comput. Syst. 97, 849–872.
Holland, J.H., 1992. Genetic algorithms. Sci. Am. 267 (1), 66–73.
Houssein, E.H., Saad, M.R., Hashim, F.A., Shaban, H., Hassaballah, M., 2020. Lévy flight distribution: A new metaheuristic algorithm for solving engineering optimization problems. Eng. Appl. Artif. Intell. 94, 103731.
Hussain, K., Salleh, M.N.M., Cheng, S., Shi, Y., 2019. On the exploration and exploitation in popular swarm-based metaheuristic algorithms. Neural Comput. Appl. 31 (11), 7665–7683.
Jain, M., Singh, V., Rani, A., 2019. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. Swarm Evol. Comput. 44, 148–175.
Kamboj, V.K., Nandi, A., Bhadoria, A., Sehgal, S., 2020. An intensify Harris Hawks optimizer for numerical and engineering optimization problems. Appl. Soft Comput. 89, 106018.
Karaboga, D., Basturk, B., 2007. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J. Glob. Optim. 39 (3), 459–471.
Kaur, S., Awasthi, L.K., Sangal, A.L., Dhiman, G., 2020. Tunicate swarm algorithm: A new bio-inspired based metaheuristic paradigm for global optimization. Eng. Appl. Artif. Intell. 90, 103541.
Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, Vol. 4. IEEE, pp. 1942–1948.
Khishe, M., Mosavi, M.R., 2020. Chimp optimization algorithm. Expert Syst. Appl. 149, 113338.
Kurban, R., Durmus, A., Karakose, E., 2021. A comparison of novel metaheuristic algorithms on color aerial image multilevel thresholding. Eng. Appl. Artif. Intell. 105, 104410.
Li, X., Han, S., Zhao, L., Gong, C., Liu, X., 2017. New dandelion algorithm optimizes extreme learning machine for biomedical classification problems. Comput. Intel. Neurosci. 2017.
Mantegna, R.N., 1994. Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Phys. Rev. E 49 (5), 4677.
Meng, Q.A., Wang, Q., Zhao, K., Wang, P., Liu, P., Liu, H., Jiang, L., 2016. Hydroactuated configuration alteration of fibrous dandelion pappi: Toward self-controllable transport behavior. Adv. Funct. Mater. 26 (41), 7378–7385.
MiarNaeimi, F., Azizyan, G., Rashki, M., 2021. Horse herd optimization algorithm: A nature-inspired algorithm for high-dimensional optimization problems. Knowl.-Based Syst. 213, 106711.
Mirjalili, S., 2016. SCA: a sine cosine algorithm for solving optimization problems. Knowl.-Based Syst. 96, 120–133.
Mirjalili, S., Lewis, A., 2016. The whale optimization algorithm. Adv. Eng. Softw. 95, 51–67.
Mirjalili, S., Mirjalili, S.M., Hatamlou, A., 2016. Multi-verse optimizer: a nature-inspired algorithm for global optimization. Neural Comput. Appl. 27 (2), 495–513.
Mohamed, A.A.A., Mohamed, Y.S., El-Gaafary, A.A., Hemeida, A.M., 2017. Optimal power flow using moth swarm algorithm. Electr. Power Syst. Res. 142, 190–206.
Mohammadi-Balani, A., Nayeri, M.D., Azar, A., Taghizadeh-Yazdi, M., 2021. Golden eagle optimizer: A nature-inspired metaheuristic algorithm. Comput. Ind. Eng. 152, 107050.
Nand, R., Sharma, B.N., Chaudhary, K., 2021. Stepping ahead firefly algorithm and hybridization with evolution strategy for global optimization problems. Appl. Soft Comput. 109, 107517.
Nematollahi, A.F., Rahiminejad, A., Vahidi, B., 2020. A novel meta-heuristic optimization method based on golden ratio in nature. Soft Comput. 24 (2), 1117–1151.
Pereira, J.L.J., Francisco, M.B., Diniz, C.A., Oliver, G.A., Cunha, Jr., S.S., Gomes, G.F., 2021. Lichtenberg algorithm: A novel hybrid physics-based meta-heuristic for global optimization. Expert Syst. Appl. 170, 114522.
Pu, Y.F., Zhou, J.L., Zhang, Y., Zhang, N., Huang, G., Siarry, P., 2013. Fractional extreme value adaptive training method: fractional steepest descent approach. IEEE Trans. Neural Netw. Learn. Syst. 26 (4), 653–662.
Punnathanam, V., Kotecha, P., 2016. Yin-Yang-pair optimization: A novel lightweight optimization algorithm. Eng. Appl. Artif. Intell. 54, 62–79.
Sheldon, J.C., Burrows, F.M., 1973. The dispersal effectiveness of the achene–pappus units of selected compositae in steady winds with convection. New Phytol. 72 (3), 665–675.
Soons, M.B., Heil, G.W., Nathan, R., Katul, G.G., 2004. Determinants of long-distance seed dispersal by wind in grasslands. Ecology 85 (11), 3056–3068.
Soubervielle-Montalvo, C., Perez-Cham, O.E., Puente, C., Gonzalez-Galvan, E.J., Olague, G., Aguirre-Salado, C.A., Cuevas-Tello, J.C., Ontanon-Garcia, L.J., 2022. Design of a low-power embedded system based on a SoC-FPGA and the honeybee search algorithm for real-time video tracking. Sensors 22 (3), 1280.
Storn, R., Price, K., 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J. Glob. Optim. 11 (4), 341–359.
Talatahari, S., Azizi, M., 2021. Chaos game optimization: a novel metaheuristic algorithm. Artif. Intell. Rev. 54 (2), 917–1004.
Tan, Y., Zhu, Y., 2010. Fireworks algorithm for optimization. In: International Conference in Swarm Intelligence. Springer, Berlin, Heidelberg, pp. 355–364.
Wilcoxon, F., 1992. Individual comparisons by ranking methods. In: Breakthroughs in Statistics. Springer, New York, NY, pp. 196–202.
Wolpert, D.H., Macready, W.G., 1997. No free lunch theorems for optimization. IEEE Trans. Evol. Comput. 1 (1), 67–82.
Zahedi, Z.M., Akbari, R., Shokouhifar, M., Safaei, F., Jalali, A., 2016. Swarm intelligence based fuzzy routing protocol for clustered wireless sensor networks. Expert Syst. Appl. 55, 313–328.
Zhou, J., Qiu, Y., Zhu, S., Armaghani, D.J., Li, C., Nguyen, H., Yagiz, S., 2021. Optimization of support vector machine through the use of metaheuristic algorithms in forecasting TBM advance rate. Eng. Appl. Artif. Intell. 97, 104015.