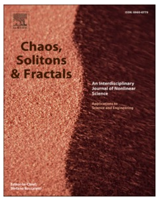
Contents lists available at ScienceDirect

Chaos, Solitons and Fractals

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



Chaotic evolution optimization: A novel metaheuristic algorithm inspired by chaotic dynamics

Yingchao Dong [a], Shaohua Zhang [b,*], Hongli Zhang [b], Xiaojun Zhou [c], Jiading Jiang [a]

[a] *School of Energy Engineering, Xinjiang Institute of Engineering, Urumqi, Xinjiang 830023, China*
[b] *School of Electrical Engineering, Xinjiang University, Urumqi, Xinjiang 830017, China*
[c] *School of Automation, Central South University, Changsha 410083, China*

A R T I C L E I N F O

Keywords:
Chaotic evolution optimization
Metaheuristic
Optimization
Discrete memristor
Hyperchaos

A B S T R A C T

In this paper, a novel population-based metaheuristic algorithm inspired by chaotic dynamics, called chaotic evolution optimization (CEO), is proposed. The main inspiration for CEO is derived from the chaotic evolution process of a two-dimensional discrete memristive map. By leveraging the hyperchaotic properties of the memristive map, the CEO algorithm is mathematically modeled to introduce random search directions for evolutionary processes. Then, the CEO is developed by integrating the crossover and mutation operations from the differential evolution (DE) framework. The proposed algorithm is evaluated by conducting experiments on 15 benchmark test problems and a sensor network localization problem, comparing its performance with 12 other metaheuristic algorithms. Experimental results demonstrate that CEO exhibits highly promising and competitive performance in comparison to widely used, classical, and well-established metaheuristic algorithms. Moreover, CEO effectively addresses the zero-bias problem observed in many recently proposed algorithms. The source code for CEO algorithm will publicly available at: https://github.com/Running-Wolf1010/CEO.

1. Introduction

Optimization algorithms have played a crucial role in solving complex, nonlinear, and multimodal problems across various fields, including engineering, economics, and artificial intelligence [1]. However, in practice, these problems often exhibit multimodal, discontinuous, non-convex, or high-dimensional characteristics. Traditional mathematical optimization methods, such as gradient descent, sequential quadratic programming, and quasi-Newton methods, tend to show limitations when dealing with such challenges [2]. These approaches heavily rely on the differentiability and convexity of the objective function, making them prone to getting trapped in local optima and inefficient when handling high-dimensional or multimodal problems [3]. In contrast, metaheuristic algorithms, a branch of optimization techniques, have emerged as valuable tools due to their independence from derivative information and their relatively relaxed requirements regarding the nature of the optimization problem [4]. Additionally, they offer robust global search capabilities, making them well-suited for addressing complex optimization tasks.

Since the 1970s, metaheuristic algorithms have experienced nearly three decades of a golden era, marked by the development and widespread application of numerous classical optimization algorithms. In 1975, Holland introduced the genetic algorithm (GA) [5], which evolved into an important family of algorithms known as evolutionary algorithms. In 1983, Kirkpatrick proposed the simulated annealing (SA) [6] algorithm based on the Metropolis criterion. In 1992, Dorigo, in his doctoral thesis, introduced the ant colony optimization (ACO) [7,8] algorithm, which demonstrated remarkable performance in solving combinatorial optimization problems. In 1995, Eberhart and colleagues, inspired by the social behavior and flight patterns of birds, proposed the particle swarm optimization (PSO) [9] algorithm. These foundational algorithms laid a critical groundwork for subsequent research in metaheuristic optimization.

In the 21st century, the golden era of metaheuristic algorithm development has gradually passed, but various new algorithms continue to emerge, particularly those inspired by natural processes. Examples include the grey wolf optimizer (GWO) [10], whale optimization algorithm (WOA) [11], bat algorithm (BA) [12], ant lion optimizer (ALO) [13], firefly algorithm (FA) [14], harris hawks optimization (HHO) [15], and runge kutta optimizer (RUN) [16]. However, as the number of nature-inspired algorithms has surged in recent years, researchers have observed that despite their diverse designs, many of these algorithms

* Corresponding author.
E-mail address: zhang676160@foxmail.com (S. Zhang).

https://doi.org/10.1016/j.chaos.2025.116049
Received 28 September 2024; Received in revised form 11 December 2024; Accepted 19 January 2025
Available online 29 January 2025
0960-0779/© 2025 Elsevier Ltd. All rights are reserved, including those for text and data mining, AI training, and similar technologies.

exhibit "core" operational similarities, often concealed behind different models [17–21]. This has led to calls from the evolutionary computation community, with many scholars expressing concern that most newly proposed algorithms are hidden behind metaphor-rich terminology and lack genuine innovation [22].

Despite the extensive number of metaheuristic algorithms that have been proposed and applied to various optimization tasks, several common issues and challenges persist. For instance, many algorithms are prone to premature convergence or stagnation during the search process, where they become trapped in local optima or stagnate without fully exploring the entire search space. Additionally, as pointed out by Jakub Kudela, most newly proposed metaheuristic algorithms, such as GWO, HHO, slime mould algorithm (SMA) [23], and RUN, suffer from a significant zero-bias problem. Specifically, these algorithms perform exceptionally well when the optimal solution of a test problem is zero, but their performance degrades significantly when the optimal solution deviates from zero [24,25].

In *Swarm Intelligence*, Kennedy, the creator of PSO, emphasized that "the degree of randomness determines the level of intelligence", highlighting the essence of intelligence as rooted in randomness [26]. This insight offers guidance on addressing the issues of local optima entrapment and stagnation in metaheuristic algorithms. Given that chaos exhibits strong randomness and unpredictability, many studies have introduced chaotic operators to enhance global search capabilities, achieving promising results. For example, Liu et al. [27] incorporated chaotic search into PSO, proposing the chaotic PSO (CPSO), and demonstrated its advantages over basic PSO. Kumar et al. [28] explored the impact of 10 chaotic maps on the search performance of the marine predators algorithm (MPA) [29]. Kaur et al. [30] introduced a chaotic map into WOA to develop the chaotic WOA (CWOA). Altay [31] applied a chaotic map to the SMA, accelerating its global convergence rate. Raj et al. [32] integrated the logistic map into the sine cosine algorithm (SCA) [33] to improve its search capabilities. However, previous studies have mainly used traditional one-dimensional chaotic maps to enhance the search capabilities of metaheuristic algorithms. In these approaches, individuals evolve independently during chaotic searches, ignoring collaborative interactions within the population.

Inspired by studies in Refs. [27–33], a novel population-based metaheuristic algorithm, called chaotic evolution optimization (CEO), is proposed in this paper. Specifically, the CEO algorithm leverages the hyperchaotic map of a two-dimensional discrete memristive system to guide the search directions of two individuals simultaneously, which forms the basis for the "Chaotic" aspect of the CEO name. Compared to traditional one-dimensional chaotic maps, the hyperchaotic map considers interactions among individuals and incorporates memory functionality. This enhances search diversity and allows the population to explore more promising regions of the search space. Moreover, CEO is classified as an evolutionary algorithm because it incorporates mutation, crossover, and selection operations from the differential evolution (DE) [34] framework. Although CEO follows the overall structure of DE, making it a specialized form of DE, there are fundamental distinctions between CEO and traditional DE, which are outlined in the following three aspects:

(1) The mutation operator in CEO differs from that in DE. CEO utilizes the sequence of the memristive hyperchaotic map to guide population evolution, providing more random evolution directions compared to DE. This helps avoid the issue encountered in DE where the differential term approaches zero in later stages of evolution, leading to local optima entrapment or stagnation.
(2) In DE, each individual only has a single evolution direction. However, in CEO, each individual can generate multiple chaotic evolution directions. This enhancement significantly improves the algorithm's ability to explore and exploit the current individual, thus increasing the likelihood of finding the global optimal solution.
(3) In DE, the crossover control parameter C_r is typically a fixed value within the interval [0,1]. In contrast, CEO does not maintain a fixed C_r value but instead randomly selects a value from the interval [0,1] at each iteration. Additionally, CEO discards the fixed scaling factor F used in DE, replacing it with a random number from the interval [0,1]. These modifications introduce greater randomness into the crossover control parameter and scaling factor, thereby enhancing the diversity of the search, reducing the risk of getting trapped in local optima, and simplifying the algorithm by reducing the number of parameters, thus making it easier to use.

The remainder of this paper is organized as follows. Section 2 introduces the background and inspiration behind the development of CEO. Section 3 presents the mathematical model and computational process of the CEO algorithm. Section 4 showcases the experimental results of CEO in solving 15 benchmark test problems and a sensor network localization problem, with comparative discussions against various algorithms. Finally, Section 5 concludes the work and provides directions for future research.

2. Background

Many dynamic evolutionary phenomena in real life exhibit chaos, which can be utilized to uncover the objective laws of nature and facilitate the rapid advancement of industrial applications. The chaotic behavior is characterized by dense periodic orbits, exhibiting various properties such as initial sensitivity, topological mixing, and unpredictability [35]. Current research is keenly focused on the construction of diverse chaotic dynamical systems, facilitating the broad application of chaos across numerous industrial fields. For example, leveraging the randomness inherent in chaotic systems, researchers have developed various secure communication strategies [36], metaheuristic algorithms [31], and privacy protection systems for the IoT [37]. Thus, it is evident that chaotic sequences play an irreplaceable and significant role in industrial processes.

However, with the deepening of dynamical analysis technology and the development of artificial intelligence, the chaotic degradation has been gradually discovered and the chaotic evolution can be accurately predicted, which undoubtedly brings great risks to chaos-based applications. Thus, a phenomenon known as hyperchaos, which is more complex than chaos itself and characterized by two positive Lyapunov exponents (LEs), has received considerable attention. Undeniably, to realize hyperchaos, the continuous system needs to be at the expense of high dimensions and huge analog circuits, which is inconsistent with the requirements of practical applications. Therefore, realizing hyperchaos within the discrete map represents a reliable path, as it only needs two dimensions, whereas continuous dynamical systems require a minimum of four dimensions [38].

In 1971, memristor, as the fourth nonlinear electronic component, was proposed by Chua based on the circuit symmetry theory [39]. In recent years, ongoing explorations have unveiled the nonlinear, nanoscale, and biomimetic characteristics of memristors, thereby facilitating the development of chaotic and hyperchaotic systems. As its latest variant, the discrete memristor is widely employed in the construction of the hyperchaotic map. At the application level, discrete memristive maps have been reported in various fields, including the optical communication [40], the geolocation grid encryption [41], and the reservoir computing system [42]. However, the development of high-performance metaheuristic algorithms utilizing the discrete memristor hyperchaotic map remains a novel area of research. A charge-controlled discrete memristor model has been formulated as:

$$\begin{cases} v_t = M(q_t)i_t, \\ q_{t+1} = q_t + i_t, \end{cases} \tag{1}$$

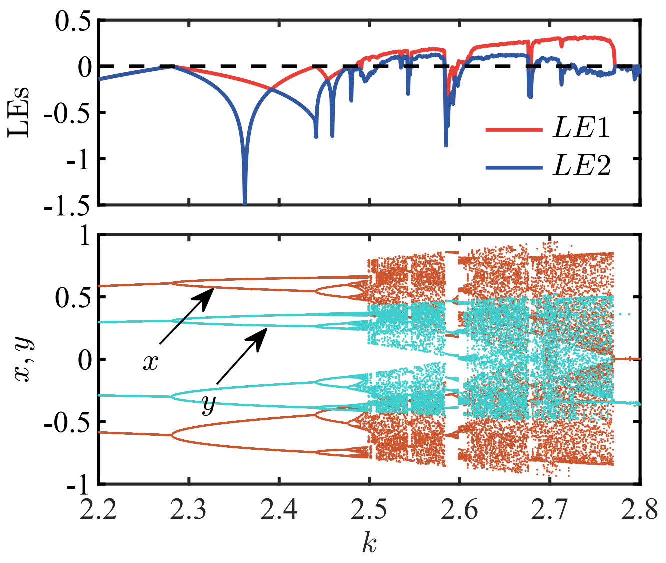


Fig. 1. The bifurcation diagram and LEs curve controlled by the parameter k in the E-DM map with $k \in [2.2, 2.8]$ and $(x_0, y_0) = (-0.5, 0.4)$.

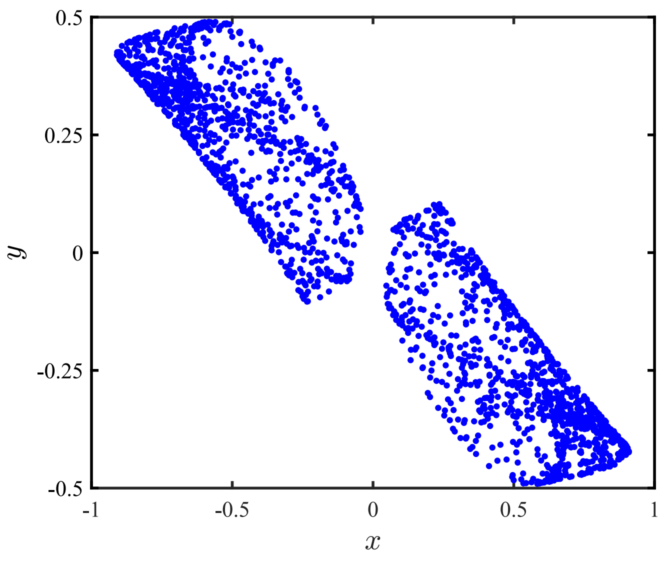


Fig. 3. When $(x_0, y_0) = (-0.5, 0.4)$, the phase portrait in the $x-y$ plane is illustrated.

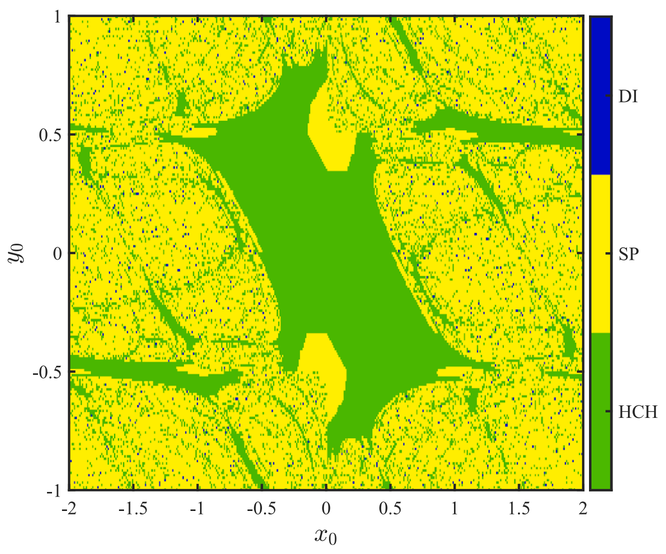


Fig. 2. For $k = 2.66$, the local basin of attraction exhibited by the E-DM map in the $x_0 - y_0$ plane with $x_0 \in [-2, 2]$ and $y_0 \in [-1, 1]$.

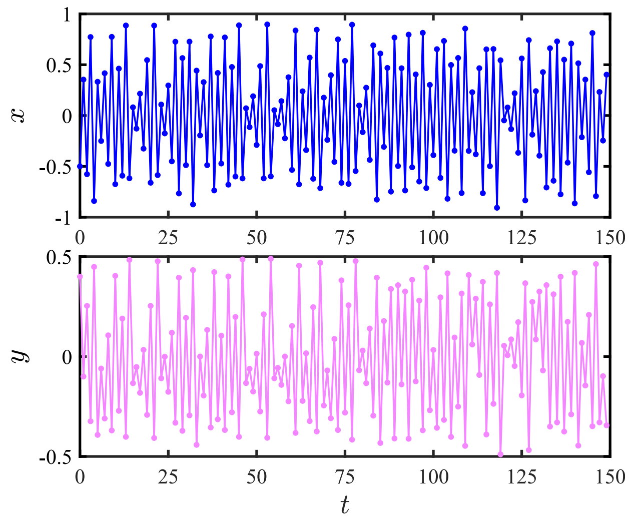


Fig. 4. When $(x_0, y_0) = (-0.5, 0.4)$, the time series concerning states x and y are illustrated.

where v_t, i_t, and q_t represent the sampled values of voltage $v(t)$, current $i(t)$, and charge $q(t)$ at the t-th iteration in the continuous memristor. In Eq. (1), the discrete memductance equation is deliberately chosen as $M(q_t) = e^{-\cos\pi q_t} - 1$, thereby deriving the exponential discrete memristor. According to literature [43], when the voltage v_t is processed through a proportional controller k and utilized as delayed feedback input, a unified memristive map can be obtained. Essentially, by substituting the output v_t and input i_t in Eq. (1) with x_{t+1} and x_t, respectively, an exponential discrete memristor (E-DM) map is constructed as:

$$\begin{cases} x_{t+1} = k \cdot (e^{-\cos\pi y_t} - 1) \cdot x_t \\ y_{t+1} = y_t + x_t \end{cases} \tag{2}$$

When the initial conditions are set to $(x_0, y_0) = (-0.5, 0.4)$ and k is varied within the interval $[1, 2]$, the bifurcation points x and y of the E-DM map are extracted using the maximum value method, as depicted by the brown and cyan point sets in the lower section of Fig. 1. Next, the LEs are computed using the QR decomposition method, and are represented by the red and dark blue curves in the upper section of Fig. 1. The bifurcation diagram reveals that the E-DM map undergoes forward period-doubling bifurcations at $k = 2.28$, $k = 2.44$, $k = 2.476$, and $k = 2.483$, resulting in period-1, period-2, period-4, and period-8 behaviors. During this process, both LE1 and LE2 do not exceed 0. However, the E-DM map enters a wide chaotic bifurcation interval at $k = 2.486$, characterized by a dense set of bifurcation points. In the interval $k \in [2.486, 2.77]$, the presence of two positive LEs identifies the hyperchaotic behavior, whereas a single positive LE corresponds to chaotic behavior. Interestingly, there are several narrow periodic windows in the chaotic bifurcation interval, and the map exits chaos and enters single-period operation at $k = 2.77$ through crisis scenario. Therefore, the bifurcation diagram and LEs curve demonstrate the complex dynamical behaviors induced by parameter variations in the E-DM map.

Further, $k = 2.66$ is selected according to the bifurcation diagram, and the initial dynamical analysis of the E-DM map is also conducted. Using the initial conditions x_0 and y_0 as control parameters, the dynamical attraction region of the E-DM map is calculated in the $x_0 - y_0$ plane. By calculating the LEs and the number of periodic cycles, the hyperchaos (HCH), stable point (SP), and divergent (DI) attractors are respectively marked in green, yellow, and blue, thereby rendering a local basin of attraction with a singular fractal boundary in the $x_0 - y_0$ plane, as illustrated in Fig. 2. Within this basin, the parameter domains of different attractors can be directly identified. For instance, the robust hyperchaotic attractor can be obtained within the region for $x_0 \in [-0.5, 0.5]$ and $y_0 \in [-0.4, 0.4]$. When $(x_0, y_0) = (-0.5, 0.4)$ is selected, Fig. 3 presents two separated hyperchaotic phase point sets and illustrates the complex evolutionary pathway of the hyperchaotic attractor. Simultaneously, the state variables corresponding to each iteration are represented by the time series shown in Fig. 4. It is evident that the iterative processes of sequences x and y are random and unpredictable, which is particularly beneficial for providing evolutionary directions in metaheuristic algorithms. This enhances the randomness of the search and increases the probability of discovering the global optimal solution.

3. Chaotic evolution optimization (CEO)

In this section, we mathematically model the proposed CEO algorithm. The overall framework of CEO is similar to the DE algorithm, including mutation, crossover, and selection operations. The key difference lies in the design of the mutation operator, where CEO employs a two-dimensional discrete memristive hyperchaotic map to provide the mutation direction for each individual.

3.1. Mutation operation

CEO, as a population-based evolutionary algorithm, utilizes the following unified search framework for its mutation operator:

$$\widetilde{\mathbf{x}}_{t+1} = \mathbf{x}_t + a \cdot \mathbf{d}_t \tag{3}$$

where $\mathbf{x}_t$ and $\widetilde{\mathbf{x}}_{t+1}$ are the current individual and the mutated individual respectively; a represents the search step size; $\mathbf{d}_t$ is the evolution direction, which is generated by the dynamical map Eq. (2).

The main idea of CEO is to utilize the hyperchaotic properties of the two-dimensional memristive hyperchaotic map, as shown in Eq. (2), to provide evolutionary direction for the population. It is important to emphasize that, in order for the proposed CEO algorithm to effectively exploit the hyperchaotic characteristics of Eq. (2), the two individuals $\mathbf{x}_t$ and $\mathbf{y}_t$ selected from CEO must be mapped to the ranges of [−0.5,0.5] and [−0.25,0.25], respectively, as specified in Eq. (4).

$$\begin{cases} \mathbf{x}_t^{'} = \dfrac{\mathbf{x}_t - \mathbf{lb}}{\mathbf{ub} - \mathbf{lb}} - 0.5 \\ \mathbf{y}_t^{'} = \dfrac{\mathbf{y}_t - \mathbf{lb}}{\mathbf{ub} - \mathbf{lb}} \times 0.5 - 0.25 \end{cases} \tag{4}$$

where $\mathbf{x}_t^{'}$ and $\mathbf{y}_t^{'}$ are the chaotic initial positions after mapping, respectively, and their values are within the intervals [−0.5,0.5] and [−0.25,0.25] respectively; $\mathbf{lb}$ and $\mathbf{ub}$ are the lower and upper bounds of the current population variables, respectively.

N chaotic candidate individuals $\mathbf{x\_chaos}$ and $\mathbf{y\_chaos}$ can be generated by applying Eq. (2) for individuals $\mathbf{x}_t^{'}$ and $\mathbf{y}_t^{'}$ respectively. The specific pseudocodes are as follows:

```
1: for $n \leftarrow 1$ to $N$ do
2: $\mathbf{x\_chaos}(n,:) \leftarrow k \cdot (e^{-\cos\pi \mathbf{y}_t^{'}} - 1) \cdot \mathbf{x}_t^{'}$
3: $\mathbf{y\_chaos}(n,:) \leftarrow \mathbf{y}_t^{'} + \mathbf{x}_t^{'}$
4: end for
```

In the aforementioned pseudocode, N represents the number of chaotic samples. Thus, by applying Eq. (2), N chaotic candidate individuals $\mathbf{x\_chaos} = \{\mathbf{x\_chaos}^1, \ldots, \mathbf{x\_chaos}^N\}$ and $\mathbf{y\_chaos} = \{\mathbf{y\_chaos}^1, \ldots, \mathbf{y\_chaos}^N\}$ can be generated. These chaotic individuals can then be mapped back to the actual positions $\mathbf{x\_chaos}' = \{\mathbf{x\_chaos}^{1'}, \ldots, \mathbf{x\_chaos}^{N'}\}$ and $\mathbf{y\_chaos}' = \{\mathbf{y\_chaos}^{1'}, \ldots, \mathbf{y\_chaos}^{N'}\}$ of the optimization problem through the inverse mapping, as described in Eq. (5).

$$\begin{cases} \mathbf{x\_chaos}^{n'} = (\mathbf{x\_chaos}^n + 0.5) \times (\mathbf{ub} - \mathbf{lb}) + \mathbf{lb} \\ \mathbf{y\_chaos}^{n'} = (\mathbf{y\_chaos}^n + 0.25) \times 2 \times (\mathbf{ub} - \mathbf{lb}) + \mathbf{lb} \end{cases} \tag{5}$$

Obviously, based on $\mathbf{x\_chaos}^{n'}$ and $\mathbf{y\_chaos}^{n'}$, N evolutionary directions can be generated for individuals $\mathbf{x}_t$ and $\mathbf{y}_t$ respectively, as shown in Eq. (6):

$$\begin{cases} \mathbf{d}_{x,t}^n = \mathbf{x\_chaos}^{n'} - \mathbf{x}_t \\ \mathbf{d}_{y,t}^n = \mathbf{y\_chaos}^{n'} - \mathbf{y}_t \end{cases} \tag{6}$$

where $n = \{1, 2, \cdots, N\}$; $\mathbf{d}_{x,t}^n$ and $\mathbf{d}_{y,t}^n$ are the evolutionary directions of $\mathbf{x}_t$ and $\mathbf{y}_t$ respectively.

In summary, by combining Eqs. (3) and (6), the mutation operator of CEO can be derived, as shown in Eq. (7).

$$\begin{cases} \widetilde{\mathbf{x}}_{t+1}^n = \mathbf{x}_t + a \cdot (\mathbf{x\_chaos}^{n'} - \mathbf{x}_t) \\ \widetilde{\mathbf{y}}_{t+1}^n = \mathbf{y}_t + a \cdot (\mathbf{y\_chaos}^{n'} - \mathbf{y}_t) \end{cases} \tag{7}$$

where a is the search step size, also known as the scale factor, which is a random number on the interval [0,1]. Obviously, N mutant individuals can be generated for individuals $\mathbf{x}_t$ and $\mathbf{y}_t$ respectively.

It can be seen from Eq. (7) that it has strong global exploration capabilities, but this may lead to slow convergence speed of the algorithm. Therefore, in order to further improve the local development capabilities of the algorithm, Eq. (8) is used to search near the best solution in the current population to speed up the convergence speed of the algorithm.

$$\begin{cases} \widetilde{\mathbf{x}}_{t+1}^n = \mathbf{Best}_t + a \cdot (\mathbf{x\_chaos}^{n'} - \mathbf{x}_t) \\ \widetilde{\mathbf{y}}_{t+1}^n = \mathbf{Best}_t + a \cdot (\mathbf{y\_chaos}^{n'} - \mathbf{y}_t) \end{cases} \tag{8}$$

where $\mathbf{Best}_t$ is the best solution for the current population.

3.2. Crossover operation

After mutation, perform a binomial crossover operator on $(\mathbf{x}_t, \widetilde{\mathbf{x}}_{t+1}^n)$ and $(\mathbf{y}_t, \widetilde{\mathbf{y}}_{t+1}^n)$ respectively to generate trial vectors $\mathbf{x\_trial}_t^n = \left(x\_trial_{1,t}^n, x\_trial_{2,t}^n, \cdots, x\_trial_{Dim,t}^n\right)$ and $\mathbf{y\_trial}_t^n = \left(y\_trial_{1,t}^n, y\_trial_{2,t}^n, \cdots, y\_trial_{Dim,t}^n\right)$. Taking $(\mathbf{x}_t, \widetilde{\mathbf{x}}_{t+1}^n)$ as an example, the specific crossover process is shown in Eq. (9):

$$x\_trial_{j,t}^n = \begin{cases} \widetilde{x}_{j,t+1}^n, \text{if } \left(rand_j(0,1] \le C_r\right) \text{ or } (j = j_{\text{rand}}) \\ x_{j,t}, \text{otherwise} \end{cases} \tag{9}$$

where Dim is the dimension of the optimization problem, $j = 1,2,\cdots,Dim$; j_{rand} is an integer randomly selected in the interval [1,Dim]; $rand_j(0,1]$ is a random number generated for each j and evenly distributed between 0 and 1, C_r is the crossover control parameter. In CEO, the value of C_r is a random number in [0,1] for each iteration.

3.3. Selection operation

For $\mathbf{x}_t$ and $\mathbf{y}_t$, N trial vectors $\mathbf{x\_trial}_t^n$ and $\mathbf{y\_trial}_t^n$ can be generated respectively. CEO adopts a greedy criterion to select the generated experimental vectors. The selection operators are shown in Eqs. (10) and (11).

$$\mathbf{x}_{t+1} = \begin{cases} \mathbf{x\_trial}_t^*, \text{if } f\left(\mathbf{x\_trial}_t^*\right) \le f(\mathbf{x}_t) \\ \mathbf{x}_t, \quad \text{otherwise} \end{cases} \tag{10}$$

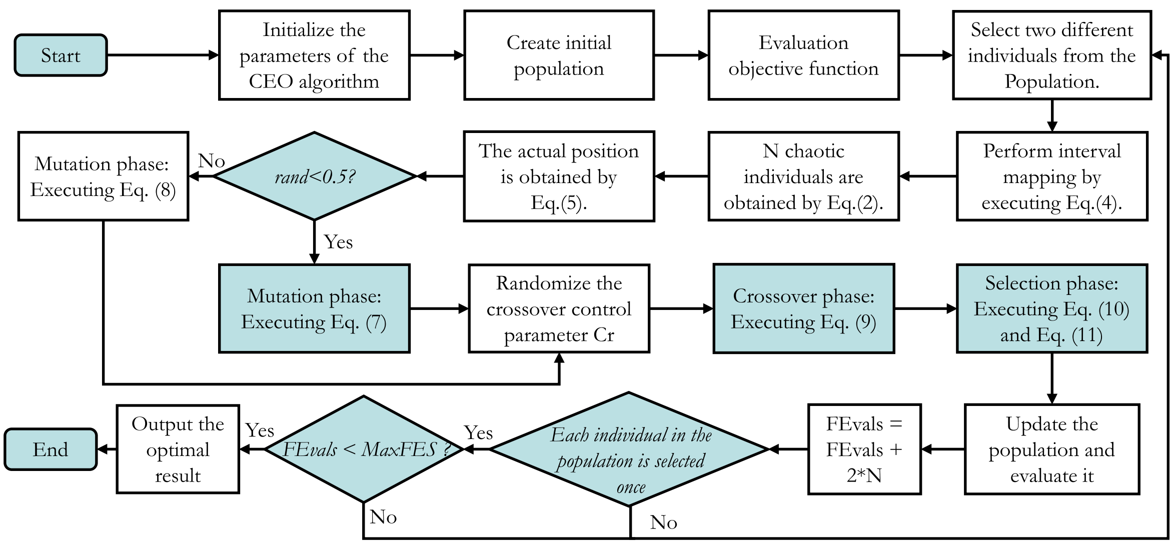

Start
Initialize the parameters of the CEO algorithm
Create initial population
Evaluation objective function
Select two different individuals from the Population.
Perform interval mapping by executing Eq.(4).
N chaotic individuals are obtained by Eq.(2).
The actual position is obtained by Eq.(5).
rand<0.5?
No
Yes
Mutation phase: Executing Eq. (8)
Mutation phase: Executing Eq. (7)
Randomize the crossover control parameter Cr
Crossover phase: Executing Eq. (9)
Selection phase: Executing Eq. (10) and Eq. (11)
Update the population and evaluate it
FEvals = FEvals + 2*N
Each individual in the population is selected once
Yes
No
FEvals < MaxFES ?
Yes
No
Output the optimal result
End

Fig. 5. Flowchart of the proposed CEO algorithm.

Table 1
Fifteen well-known benchmark functions.

Name	Functions	Range	$\mathbf{x}^*$	$f(\mathbf{x}^*)$
Spherical	$f(x) = \sum_{i=1}^{n} x_i^2$	[−100,100]	[0,0, …]	0
Schwefel 2.22	$f(x) = \sum_{i=1}^{n} \lvert x_i\rvert + \prod_{i=1}^{n} \lvert x_i\rvert$	[−10,10]	[0,0, …]	0
Schewefel 1.2	$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j\right)^2$	[−100,100]	[0,0, …]	0
Rosenbrock	$f(x) = \sum_{i=1}^{n} \left(100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2\right)$	[−30,30]	[1,1, …]	0
Schwefel 2.4	$f(x) = \sum_{i=1}^{n} \left[(x_i - 1)^2 + \left(x_1 - x_i^2\right)^2\right]$	[0,10]	[1,1, …]	0
High conditioned elliptic	$f(x) = \sum_{i=1}^{n} \left(10^6\right)^{\frac{i-1}{n-1}} x_i^2$	[−100,100]	[0,0, …]	0
Tablet	$f(x) = 10^6 \cdot x_1^2 + \sum_{i=2}^{n} x_i^6$	[−100,100]	[0,0, …]	0
Zakharov	$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{n} 0.5ix_i\right)^4$	[−5,10]	[0,0, …]	0
Penalized 1	$f(x) = \frac{\pi}{n}\left\{ \sum_{i=1}^{n-1} (y_i - 1)^2 \left[1 + 10sin^2(\pi y_{i+1})\right] + (y_n - 1)^2 + 10sin(\pi y_1)\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i > a \\ 0 - a < x_i < a \\ k(-x_i - a)^m x_i < a \end{cases}$	[−50,50]	[−1,-1, …]	0
Penalized 2	$f(x) = 0.1\left\{ \sum_{i=1}^{n-1} (x_i - 1)^2 \left[1 + sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2 \left[1 + sin^2(2\pi x_n)\right] + sin^2(3\pi x_1)\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	[−50,50]	[1,1, …]	0
Ackley	$f(x) = 20 + e - 20exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - exp\left(\frac{1}{n}\sum_{i=1}^{n} cos(2\pi x_i)\right)$	[−32,32]	[0,0, …]	0
Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	[−600, 600]	[0,0, …]	0
Rastrigin	$f(x) = \sum_{i=1}^{n} \left(x_i^2 - 10cos(2\pi x_i) + 10\right)$	[−5.12, 5.12]	[0,0, …]	0
Levy and Montalvo 1	$f(x) = \frac{\pi}{n}\left(10sin^2(\pi y_1) + \sum_{n-1}^{i=1} (y_i - 1)^2 \left[1 + 10sin^2(\pi y_{i+1})\right] + (y_n - 1)^2\right), y_i = 1 + \frac{1}{4}(x_i + 1)$	[−10,10]	[−1,-1, …]	0
Levy and Montalvo 2	$f(x) = 0.1\left(\sum_{i=1}^{n-1} (x_i - 1)^2 \left(1 + \left(sin^2(3\pi x_{i+1})\right)\right) + (x_n - 1)^2 \left(1 + \left(sin^2(2\pi x_n)\right)\right) + \left(sin^2(3\pi x_1)\right)\right)$	[−5,5]	[1,1, …]	0

Note: $\mathbf{x}^*$, optimal solution; $f(\mathbf{x}^*)$, the optimal function value.

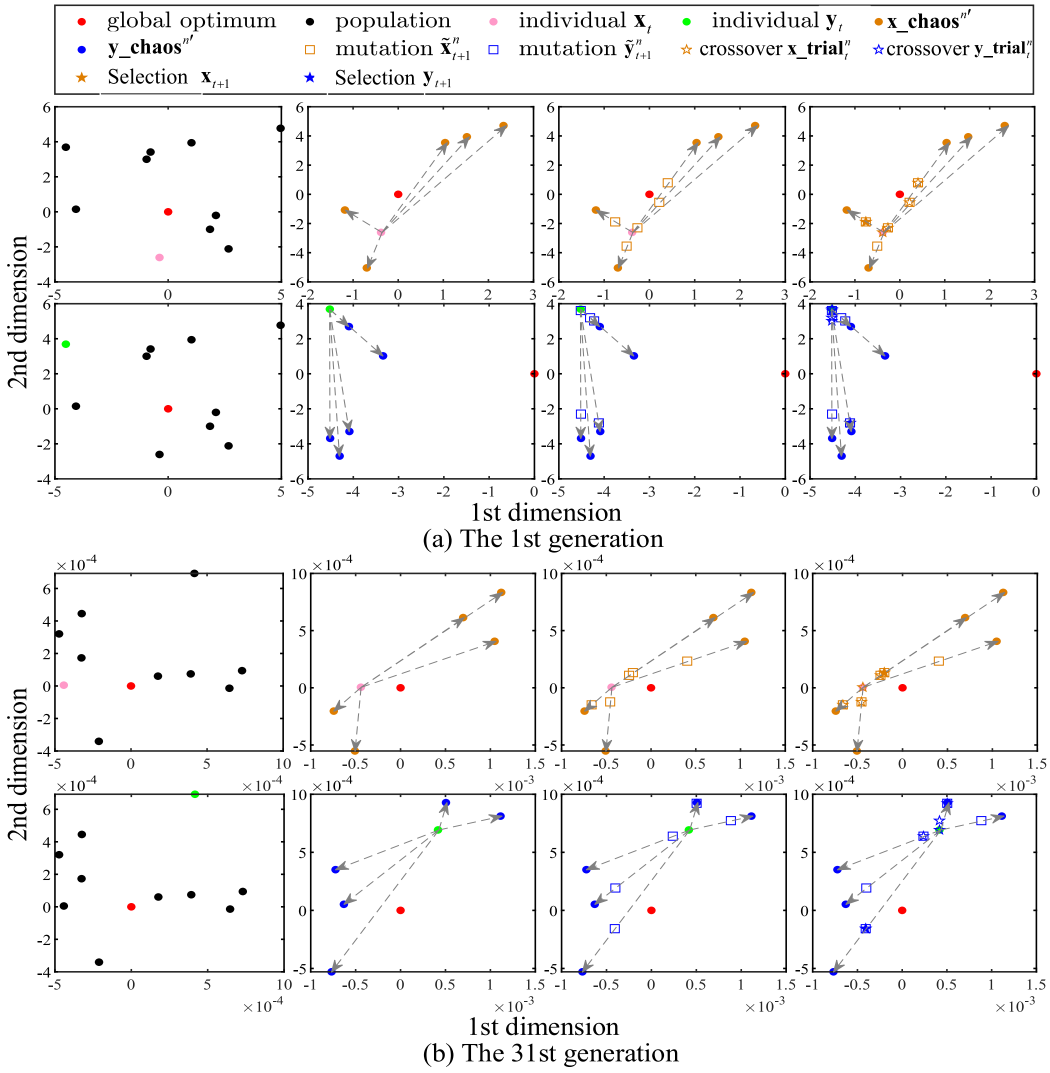


Fig. 6. Illustration of the CEO evolution process on the Rastrigin function.

$$\mathbf{y}_{t+1} = \begin{cases} \mathbf{y\_trial}^*_t, \text{if } f(\mathbf{y\_trial}^*_t) \le f(\mathbf{y}_t) \\ \mathbf{y}_t, \quad \text{otherwise} \end{cases} \tag{11}$$

where $\mathbf{x\_trial}^*_t$ and $\mathbf{y\_trial}^*_t$ are the best trial vectors of trial vectors $\mathbf{x\_trial}^n_t = \left(x\_trial^n_{1,t}, x\_trial^n_{2,t}, \cdots, x\_trial^n_{Dim,t}\right)$ and $\mathbf{y\_trial}^n_t = \left(y\_trial^n_{1,t}, y\_trial^n_{2,t}, \cdots, y\_trial^n_{Dim,t}\right)$ respectively.

3.4. *Pseudocode of CEO*

By integrating the mutation, crossover, and selection processes of CEO, a detailed pseudocode for solving minimization optimization problems is presented in Algorithm 1.

Algorithm 1. Pseudocode of CEO.

```
Input: func (Objective function); N (Number of chaotic sampling); Np (Population size);
  MaxFES (Maximum number of evaluation functions);
Output: Best (the optimal variable); fBest (the optimal function value).
1:   t = 1 /* Initialization iteration number */
2:   /* Initialize the population and evaluate the population*/
3:   [Population, fit, fBest, Best] = Initialization (func, Np, Dim)
4:   FEvals = Np;
5:   while FEvals < MaxFES do
6:   repeat
7:   [x_t, y_t] ← Select two different individuals from the Population.
8:   [x'_t, y'_t] ← Perform interval mapping on x_t and y_t by executing Eq. (4).
9:   [x_chaos, y_chaos] ← N chaotic individuals are obtained by executing Eq. (2).
10:  [x_chaos', y_chaos'] ← Executing Eq. (5) yields the actual position of the
11:  corresponding optimization problem.
12:  if rand <0.5 then
13:  [$\tilde{x}^n_{t+1}$, $\tilde{y}^n_{t+1}$]←Executing Eq. (7). ▷ Mutation phase
14:  else
15:  [$\tilde{x}^n_{t+1}$, $\tilde{y}^n_{t+1}$]←Executing Eq. (8). ▷ Mutation phase
16:  end if
17:  $C_r$ = rand (0,1)
18:  [x_trial$^n_t$, y_trial$^n_t$] ← Executing Eq. (9). ▷ Crossover phase
19:  [$x_{t+1}$, $y_{t+1}$] ← Executing Eq. (10) and Eq. (11). ▷ Selection phase
20:  [Population, fit] ← Update the population and evaluate it.
21:  FEvals = FEvals + 2*N;
22:  until all individuals in the population are selected once.
23:  t = t + 1;
24:  End while
25:  Return Best, fBest
```

(continued on next column)

(continued)

In Algorithm 1, since two individuals are selected from the population of CEO for each iteration, the population size *Np* is set to an even number >2.

Fig. 5 also shows the flowchart of the proposed CEO algorithm.

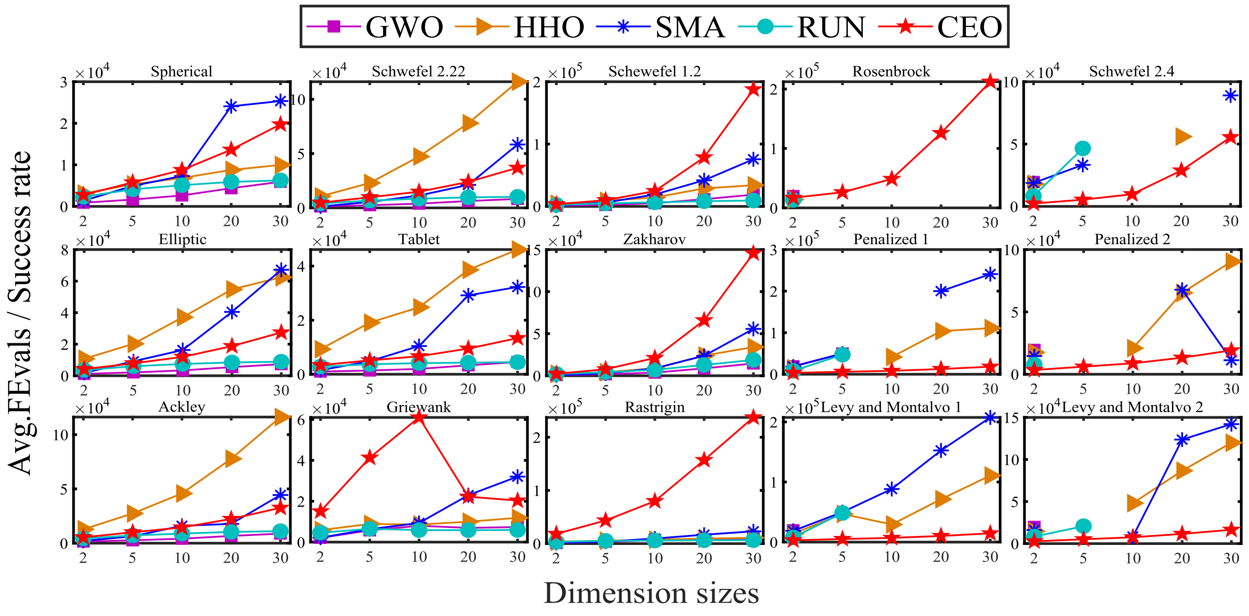


Fig. 7. Comparison of GWO, HHO, SMA, RUN, and CEO on 15 unshifted benchmark functions.

3.5. A detailed description of the CEO evolution process

In this section, the evolutionary process of the CEO algorithm is described in detail using the Rastrigin function provided in the Table 1 as an example. Fig. 6 illustrates the search range of the population at the 1st and 31st iterations of CEO, including the chaos sampling, mutation, crossover, and selection stages.

In this experiment, the population size *Np* is set to 10, and the number of chaotic samples is set to 5. The following conclusions can be drawn from Fig. 6:

(1) In the 1st iteration of CEO, the population's position is far from the global optimal solution, whereas by the 31st iteration, the population is much closer to the global optimal solution.
(2) For individuals $\mathbf{x}_t$ and $\mathbf{y}_t$, multiple distinct evolutionary directions are generated through chaotic dynamical evolution, leading to N trial vectors via mutation and crossover operations. Subsequently, the selection operation updates the current solution, retaining the best individuals. Since each individual generates multiple trial vectors in each iteration, CEO effectively leverages information from the current solutions, significantly enhancing the algorithm's search capabilities.
(3) As the evolutionary direction in CEO is derived from the dynamical map, and the step size is controlled by the scaling factor a, the differential term does not converge to zero as long as a remains non-zero. This prevents the algorithm from stagnating or becoming trapped in local optima during the later stages of evolution.

4. Results and discussion

4.1. Experimental setup

To evaluate the performance of CEO, fifteen widely used benchmark functions from literature [44] are employed. The specific expressions of these functions are provided in Table 1. CEO's results are compared with 12 other metaheuristic algorithms, including some recently popular methods such as GWO [10], HHO [15], SMA [23], and RUN [16], which have been cited >160,000 times, 4600 times, 2300 times, and 700 times, respectively, since their introduction. The comparison also included classical evolutionary and swarm intelligence algorithms like ABC [45], RCGA [46], PSO [9], and DE [34], as well as several successful and widely-used improved metaheuristics, such as CLPSO [47], GL-25 [48], SaDE [49], and LSHADE [50]. Among these, LSHADE has performed exceptionally well in several CEC (IEEE Congress on Evolutionary Computation) competitions, achieving top placements and winning global optimization contests, making it a frequent baseline algorithm.

In the experiments, the population size is set to 50, and the dimensionality of the test functions is set to 2, 5, 10, 20, and 30, with the maximum number of function evaluations (*MaxFES*) set to *Dim**10,000. To ensure a fair comparison, the detailed parameters of the comparison algorithms are kept consistent with their respective original studies. Except for solving functions Ackley and Rastrigin, where the chaotic sample numbers for CEO are set to 5 and 20, respectively, all other functions use a sample number of 1. All algorithms are independently run 51 times in MATLAB (version R2020b) on a Windows 10 desktop with an Intel(R) Core(TM) i5-9500F CPU @ 3.00 GHz. During the execution of each algorithm, iterations terminate either when the function evaluations (*FEvals*) exceed *MaxFES* or when the error tolerance of the optimal solution reaches 10^{-8}.

4.2. Comparisons with GWO, HHO, SMA, and RUN

In this section, the proposed CEO algorithm is compared with four popular metaheuristic algorithms, including GWO, HHO, SMA, and RUN. Fig. 7 illustrates the average number of function evaluations (AvgFEvals) required by the five algorithms to reach an error precision of 10^{-8} upon the completion of optimization on 15 non-shifted benchmark functions. As shown in Fig. 7, all five algorithms achieve an error precision of 10^{-8} on the non-shifted functions Spherical, Schwefel 2.22, Schewefel 1.2, Elliptic, Tablet, Zakharov, Ackley, Griewank, and Rastrigin across dimensions of 2, 5, 10, 20, and 30, indicating that the algorithms terminate before reaching the *MaxFES*. However, for the non-shifted functions Rosenbrock, Schwefel 2.4, Penalized 1, Penalized 2, Levy and Montalvo 1, and Levy and Montalvo 2, GWO, HHO, SMA, and RUN do not achieve the 10^{-8} error precision across all dimensions. Specifically, on the non-shifted Rosenbrock function, these algorithms only reach the precision of 10^{-8} in the 2-dimensional case. In contrast, the proposed CEO algorithm consistently attains an error precision of 10^{-8} across all 15 non-shifted benchmark functions.

Moreover, an interesting phenomenon is observed in Table 1. The optimal solutions of the functions Spherical, Schwefel 2.22, Schewefel

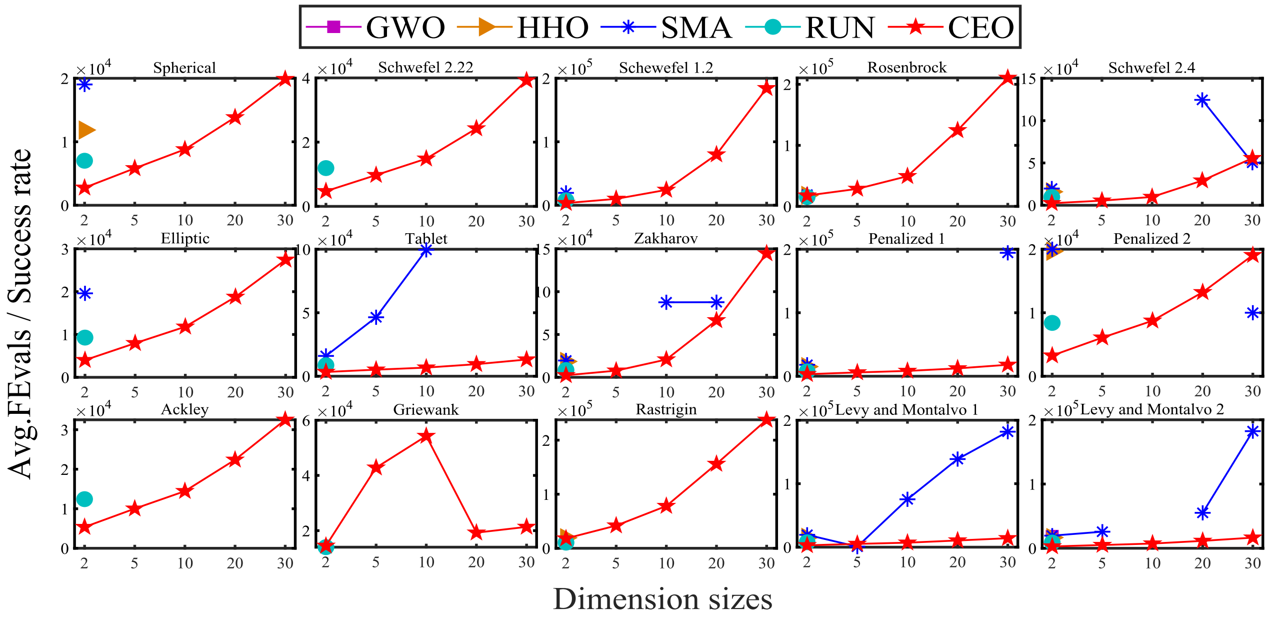


Fig. 8. Comparison of GWO, HHO, SMA, RUN, and CEO on 15 shifted benchmark functions.

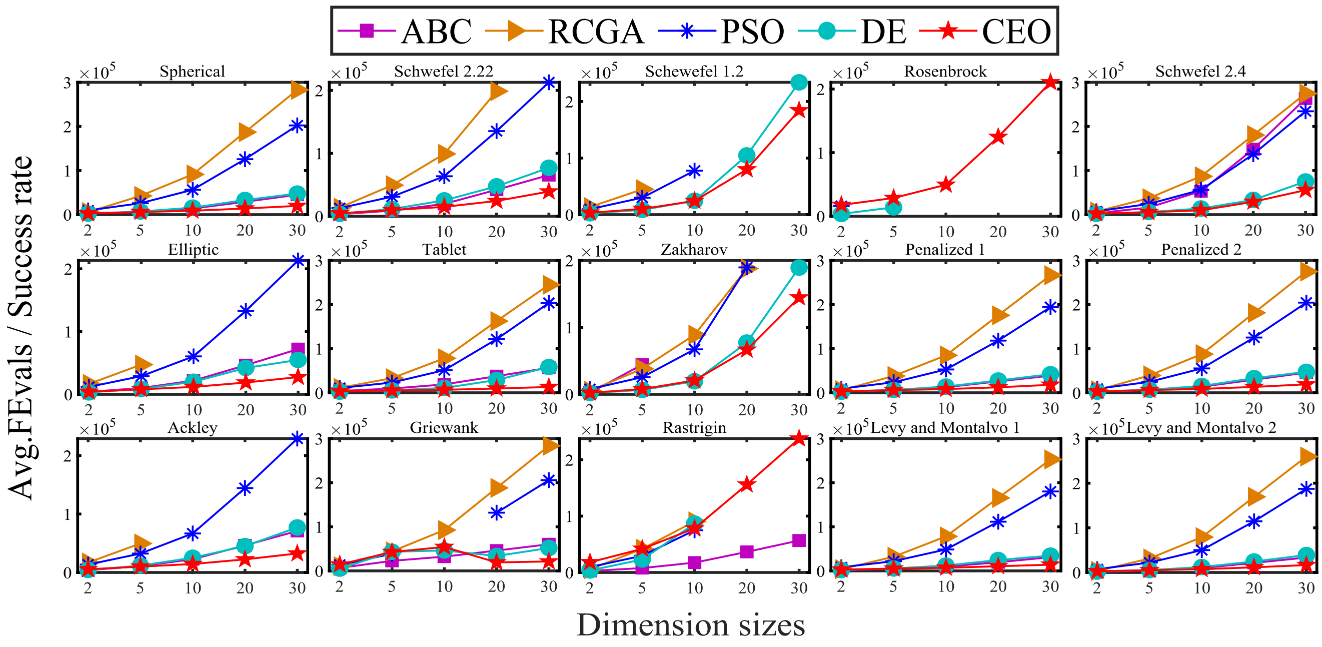


Fig. 9. Comparison of ABC, RCGA, PSO, DE, and CEO on 15 shifted benchmark functions.

1.2, Elliptic, Tablet, Zakharov, Ackley, Griewank, and Rastrigin are all located at zero, while the functions Rosenbrock, Schwefel 2.4, Panalized 1, Panalized 2, Levy and Montalvo 1, and Levy and Montalvo 2 have optimal solutions away from zero. As pointed out in the literature [25], when an algorithm performs exceptionally well on test functions with optimal solutions at zero but performs poorly on non-zero optimal functions, the algorithm may suffer from the zero-bias problem. This indicates that the algorithm might incorporate operators that are biased toward zero, causing it to perform “exceptionally well” on zero-optimal functions.

To verify whether the five comparative algorithms exhibit the zero-bias problem, Fig. 8 presents the AvgFEvals required by these algorithms on 15 shifted benchmark functions. By comparing the results of Fig. 8 with those in Fig. 7, it is found that when the *MaxFES* is set to *Dim**10000, GWO, HHO, SMA, and RUN only achieve an error precision of 10^{-8} on some 2-dimensional test functions. Moreover, SMA achieves 10^{-8} precision on only a few higher-dimensional test functions. In contrast, the proposed CEO algorithm reaches 10^{-8} precision on all test functions, and the AvgFEvals on both the shifted and non-shifted functions is very similar. These experimental results show that GWO, HHO, SMA, and RUN exhibit the zero-bias problem, while the proposed CEO algorithm does not. CEO demonstrates superior optimization performance and convergence speed compared to the other four algorithms. Furthermore, it is worth noting that the zero-bias problem in GWO, HHO, SMA, and RUN has already been validated in the literature [24,25].

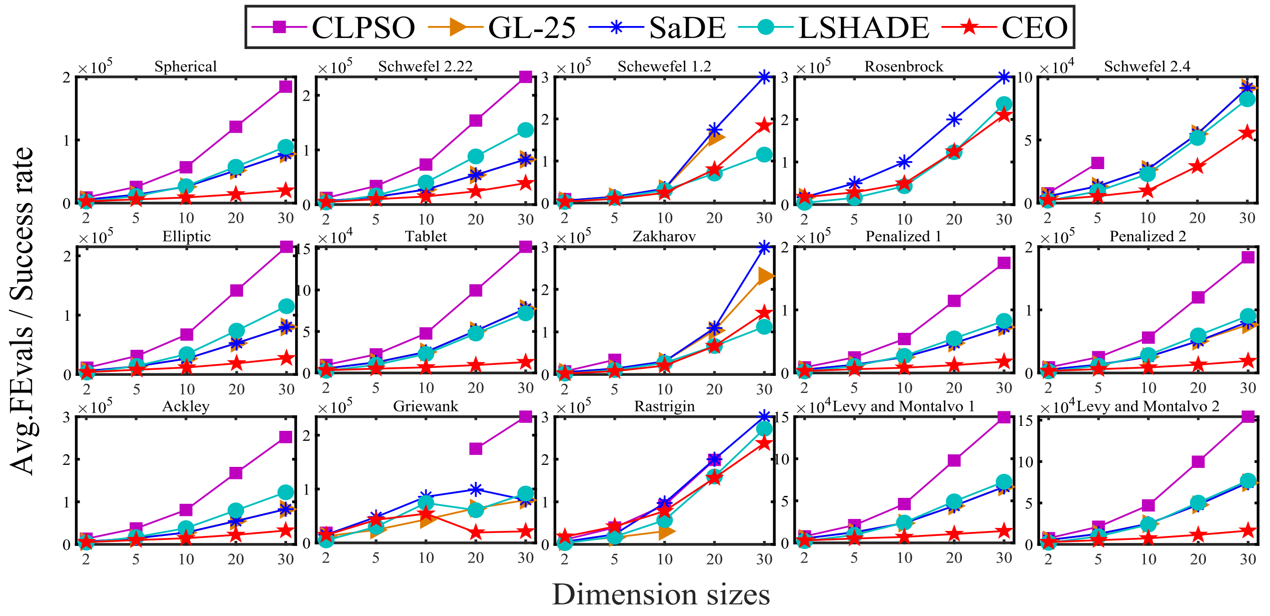


Fig. 10. Comparison of CLPSO, GL-25, SaDE, LSHADE, and CEO on 15 shifted benchmark functions.

4.3. *Comparisons with ABC, RCGA, PSO, and DE*

In this section, the proposed CEO algorithm is compared with four classic evolutionary or swarm intelligence algorithms: ABC, RCGA, PSO, and DE, all of which do not suffer from the zero-bias problem. Fig. 9 shows the AvgFEvals for the five comparative algorithms on 15 shifted benchmark functions. As shown in Fig. 9, for the functions Spherical, Schwefel 2.4, Tablet, Panalized 1, Panalized 2, Levy and Montalvo 1, and Levy and Montalvo 2, all five algorithms achieve an optimal precision of 10^{-8}. However, CEO algorithm generally consumes fewer *FEvals*, indicating that its convergence speed is faster than that of ABC, RCGA, PSO, and DE. For the remaining test functions, only CEO achieves the precision of 10^{-8} across all dimensional settings (2, 5, 10, 20, and 30 dimensions). The other four algorithms fail to reach the preset precision of 10^{-8} in some cases. Notably, for the Rosenbrock function, only the DE algorithm meets the precision of 10^{-8} in 2D and 5D, while ABC, RCGA, and PSO do not achieve the desired precision even after reaching the *MaxFES* across any dimension. Moreover, RCGA shows weaker performance on the Schwefel 1.2, Elliptic, Ackley, and Rastrigin functions. ABC performs poorly on the Schwefel 1.2 and Zakharov functions, while DE performs poorly on the Rastrigin function in 20D and 30D. Overall, CEO algorithm consumes the fewest function evaluations, while RCGA consumes the most. The proposed CEO algorithm exhibits significantly better search capability and convergence speed compared to ABC, RCGA, PSO, and DE, with RCGA performing the worst among them.

4.4. *Comparisons with CLPSO, GL-25, SaDE, and LSHADE*

In this section, the proposed CEO algorithm is compared through experimental simulations with several popular improved metaheuristic algorithms, including CLPSO, GL-25, SaDE, and LSHADE, to further validate its performance. Fig. 10 illustrates the AvgFEvals for the five comparative algorithms on 15 shifted benchmark functions. Specifically, as shown in Fig. 10, for functions such as Spherical, Schwefel 2.22, Elliptic, Tablet, Panalized 1, Panalized 2, Ackley, Levy and Montalvo 1, and Levy and Montalvo 2, all five algorithms achieve an optimal precision of 10^{-8} across all dimensions. However, CEO consistently requires fewer *FEvals* than CLPSO, GL-25, SaDE, and LSHADE. For the Schwefel 1.2 function, when the dimension exceeds 2, CLPSO fails to obtain a satisfactory solution before stopping, and for the 30-dimensional case, GL-25 is also unable to meet the precision requirement of 10^{-8}. The Rosenbrock function, which has a valley-shaped structure and becomes non-convex when the dimension exceeds 2, poses a challenge. Its global minimum is located within a long, narrow parabolic valley, which is easy to locate, but finding the exact global minimum is difficult due to the subtle changes within the valley. As shown in Fig. 10, when the dimension of the Rosenbrock function exceeds 2, only SaDE, LSHADE, and CEO are able to locate the global optimum. For the Schwefel 2.4 and Zakharov functions, CLPSO performs worse than the other four algorithms when the dimension exceeds 5. Additionally, for the Griewank function with dimensions of 5 and 10, CLPSO fails to locate the global optimum within the fixed *FEvals*. Overall, CEO performs the best in this experiment, followed by LSHADE, while CLPSO shows the weakest performance. These results demonstrate the advantages and effectiveness of the proposed CEO algorithm.

4.5. *Application for the sensor network localization*

To further verify the effect of the proposed CEO algorithm in practical application problems, this subsection applies it to sensor network localization (SNL) problems.

4.5.1. *Sensor network localization problem*

The SNL problem can be stated as follows. Given the position of n anchor points $a_1, a_2, \cdots, a_n \in R^d$ ($d = 2$ in this paper), the distance between the ith sensor and the kth anchor point is d_{ik} if $(i, k) \in N_a$, and the distance between the ith sensor and the jth sensor is e_{ij} if $(i, j) \in N_x$, where, $N_a = \{(i,k) : \|x_i - a_k\| = d_{ik} \le r_d\}$ and $N_d = \{(i,j) : \|x_i - x_j\| = e_{ij} \le r_d\}$, here r_d is the radio range, the SNL problem is to estimate m different sensor positions x_i, $i = 1, 2, \cdots, m$, such that

$$\|x_i - a_k\|^2 = d_{ik}^2, \forall (i,k) \in N_d \tag{12}$$

$$\|x_i - x_j\|^2 = e_{ij}^2, \forall (i,j) \in N_x \tag{13}$$

Since distance d_{ik} and e_{ij} can contain noise, making Eqs. (12) and (13) unfeasible, this study uses the least square method to model the SNL problem, which can be expressed as a non-convex optimization problem, as shown in the following equation

$$min \sum_{(i,j)\in N_x} \left(\|x_i - x_j\|^2 - e_{ij}^2 \right)^2 + \sum_{(i,k)\in N_a} \left(\|x_i - a_k\|^2 - d_{ik}^2 \right)^2 \tag{14}$$

Table 2
Experimental results for SNL instances.

index	CLPSO	GL-25	SMA	DE	LSHADE	CEO
Best	3.48×10^{-3}	2.62×10^{-4}	5.43×10^{-3}	2.56×10^{-5}	2.51×10^{-5}	1.78×10^{-7}
Mean	9.42×10^{-3}	1.33×10^{-2}	1.24×10^{-2}	2.46×10^{-2}	2.42×10^{-2}	2.88×10^{-3}
Worst	1.69×10^{-2}	5.30×10^{-2}	1.78×10^{-2}	5.55×10^{-2}	6.02×10^{-2}	6.78×10^{-3}
Running time/s	235.8	252.7	388.5	199.3	208.6	203.3

4.5.2. Experiments on SNL problems

The sensor example is created randomly by SFSDF [51], a MATLAB package for solving SNL problems. This study created a SNL examples with 4 anchor points a_1, a_2, a_3, a_4. The positions of the 4 anchor points are (0,0), (0,1), (1,0), (1,1), respectively. The SNL problem has 50 sensors, using a radio range of 0.3 and a noise factor of 0.001. When solving the SNL problem of 50 sensors, the experimental results are compared with CLPSO, GL-25, SMA, DE and LSHADE. The *MaxFES* of all algorithms is set to 5×10^6, the search range is set to [0,1], and the other parameters are set the same as those in the numerical experiment. All algorithms run independently 30 times. The experimental results of all SNL instances are shown in Table 2 and Figs. 11 and 12.

For the SNL problem with 50 sensors, only the CEO algorithm is able to find the global optimum. More specifically, as shown in Table 2, the best, average, and worst values obtained by CEO outperform those of the other five comparative algorithms. Additionally, CEO's running time is only slightly longer than that of the DE algorithm. Fig. 11 presents a comparison between the best sensor positions found by the five algorithms and the actual sensor positions, where circles indicate the actual positions and asterisks represent the sensor positions identified by the algorithms. From Fig. 11, it is evident that the sensor positions found by the CLPSO, GL-25, and SMA algorithms deviate significantly from the actual positions, while the DE and LSHADE algorithms result in two out of 50 sensor positions that deviate from the true locations. Only the CEO algorithm identifies sensor positions that perfectly coincide with the actual sensor positions, indicating that CEO has successfully found the global optimum for the SNL problem, whereas CLPSO, GL-25, SMA, DE, and LSHADE all become trapped in local optima.

Furthermore, as seen in the fitness iteration curves in Fig. 12, it is evident that, due to some level of noise, the sensor positions found by CEO still exhibit an error in the order of 10^{-7} compared to the true positions. However, compared to other optimization algorithms, CEO converges to an approximately global optimal solution with significantly higher precision and faster convergence. This demonstrates that CEO outperforms the other four algorithms in both search accuracy and convergence speed.

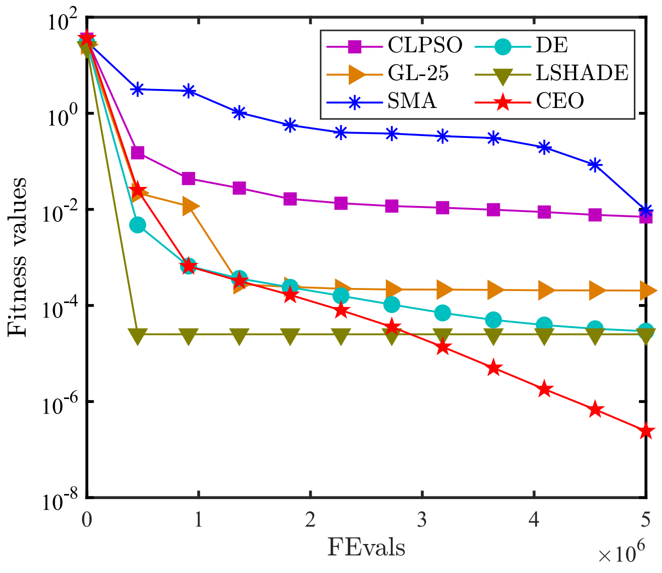


Fig. 12. Fitness values iteration curve of three different algorithms for 50 sensors and 4 anchor points.

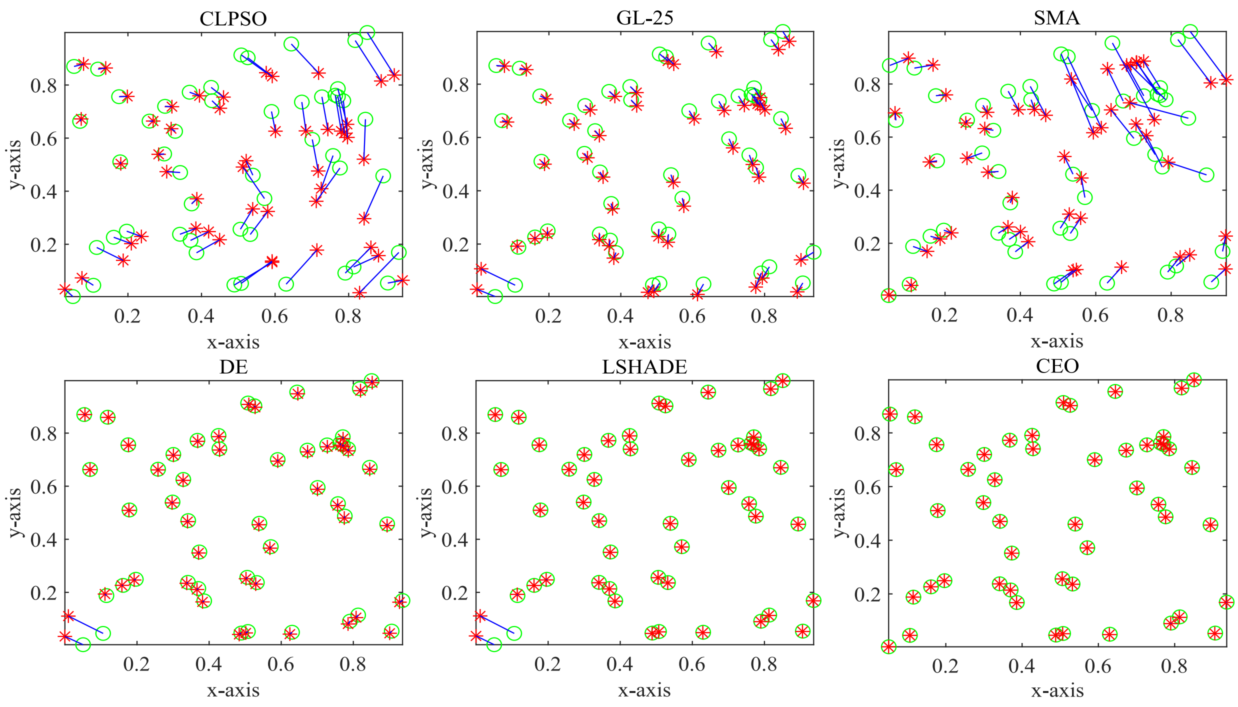


Fig. 11. Best results obtained by CLPSO, GL-25, SMA, DE, LSHADE, CEO for the SNL problem with 50 sensors.

5. Conclusions

This paper proposes a novel population-based metaheuristic optimization algorithm inspired by chaotic dynamics, called chaotic evolution optimization (CEO). The primary inspiration for CEO stems from the dynamical evolution process of a two-dimensional discrete memristive map. The hyperchaotic characteristics of the map are utilized to model the algorithm and provide random search directions for evolution. While CEO broadly adopts the framework of the DE algorithm, it differs in its mutation operator. Unlike traditional DE, CEO employs a chaotic dynamical map to generate more random evolution directions for the population, mitigating the local optima and stagnation issues caused by diminishing difference terms in the later stages of DE. Additionally, CEO introduces multiple chaotic evolution directions, enhancing the exploration and exploitation capabilities of the algorithm. By randomizing the crossover rate and scaling factor, CEO further increases search diversity, reducing the risk of being trapped in local optima while also lowering the difficulty of parameter tuning. The paper also provides graphical representations to visually and comprehensively describe the evolutionary process of CEO, clearly illustrating the mechanism of the algorithm.

Through experimental comparisons on 15 benchmark test problems and a sensor network localization problem with 50 sensors, CEO's performance is comprehensively evaluated against 12 other metaheuristic algorithms. The results demonstrate that CEO achieves promising and competitive outcomes, outperforming the comparative algorithms in terms of robustness and effectiveness, and avoiding the zero-bias problem prevalent in many recent algorithms, such as GWO, HHO, SMA, and RUN. Overall, CEO emerges as an advanced and reliable optimization tool, capable of efficiently and accurately solving complex continuous optimization problems. It is worth noting that the proposed CEO can be regarded as a framework for chaotic-based evolutionary optimization algorithms. This framework allows for the integration of other chaotic maps to extend or enhance the CEO algorithm. For example, it can incorporate offset-boosted chaotic maps [52,53], dual memristors-based hyperchaotic maps [54,55], or memristor-coupled hyperchaotic maps [56,57]. This flexibility also provides new avenues for the development of chaotic-based optimization algorithms.

Future research will focus on expanding CEO by exploring chaos-based evolutionary algorithms, as well as extending CEO to discrete optimization tasks, multi-objective optimization problems, and broader real-world applications, such as system identification, economic dispatch in power systems, and integrated energy system planning and operation.

CRediT authorship contribution statement

Yingchao Dong: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Conceptualization. **Shaohua Zhang:** Writing – review & editing, Visualization, Validation, Supervision, Software. **Hongli Zhang:** Writing – review & editing, Validation, Supervision, Funding acquisition, Formal analysis, Data curation. **Xiaojun Zhou:** Visualization, Validation, Supervision, Formal analysis, Conceptualization. **Jiading Jiang:** Visualization, Validation, Supervision, Project administration, Formal analysis, Data curation, Conceptualization.

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.

Acknowledgments

This work was partially supported by Sponsored by the Natural Science Foundation of Xinjiang Uygur Autonomous Region (Grant No. 2022D01C367), the National Natural Science Foundation of China (Grant Nos. 52267010, 72361033, 52266018), the Tianchi Talent Introduction Plan Project of Xinjiang Autonomous Region (2024XGYT-CYC12) and the Innovation Team at Xinjiang Institute of Engineering.

Data availability

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

References

[1] Dong Y, Zhang H, Wang C, et al. Robust optimal scheduling for integrated energy systems based on multi-objective confidence gap decision theory. Expert Syst Appl 2023;228:120304. https://doi.org/10.1016/j.eswa.2023.120304.
[2] Nocedal J, Wright SJ. Numerical optimization. New York: Springer; 2006.
[3] Zhang Q, Gao H, Zhan ZH, et al. Growth optimizer: a powerful metaheuristic algorithm for solving continuous and discrete global optimization problems. Knowledge-Based Syst 2023;261:110206. https://doi.org/10.1016/j.knosys.2022.110206.
[4] Bahbouhi JE, Elkouay A, Bouderba SI, et al. The whale optimization algorithm and the evolution of cooperation in the spatial public goods game. Chaos Solitons Fractals 2024;182:114873. https://doi.org/10.1016/j.chaos.2024.114873.
[5] Holland JH. Genetic algorithms. Sci Am 1992;267(1):66–73. https://www.jstor.org/stable/24939139.
[6] Kirkpatrick S, Gelatt Jr CD, Vecchi MP. Optimization by simulated annealing. Science 1983;220(4598):671–80. https://doi.org/10.1126/science.220.4598.671.
[7] Dorigo M, Gambardella LM. Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evol Comput 1997;1(1):53–66. https://doi.org/10.1109/4235.585892.
[8] Bonabeau E, Dorigo M, Theraulaz G. Inspiration for optimization from social insect behaviour. Nature 2000;406(6791):39–42. https://doi.org/10.1038/35017500.
[9] Kennedy J, Eberhart R. Particle swarm optimization[C]//proceedings of ICNN'95-international conference on neural networks. IEEE 1995;4:1942–8.
[10] Mirjalili S, Mirjalili SM, Lewis A. Grey wolf optimizer. Adv Eng Software 2014;69:46–61. https://doi.org/10.1016/j.advengsoft.2013.12.007.
[11] Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Software 2016;95:51–67. https://doi.org/10.1016/j.advengsoft.2016.01.008.
[12] Yang XS, Hossein Gandomi A. Bat algorithm: a novel approach for global engineering optimization. Eng Comput 2012;29(5):464–83. https://doi.org/10.1108/02644401211235834.
[13] Mirjalili S. The ant lion optimizer. Adv Eng Software 2015;83:80–98. https://doi.org/10.1016/j.advengsoft.2015.01.010.
[14] Yang XS, He X. Firefly algorithm: recent advances and applications. Int J Swarm Intell 2013;1(1):36–50. https://doi.org/10.1504/IJSI.2013.055801.
[15] Heidari AA, Mirjalili S, Faris H, et al. Harris hawks optimization: algorithm and applications. Future Gener Comput Syst 2019;97:849–72. https://doi.org/10.1016/j.future.2019.02.028.
[16] Ahmadianfar I, Heidari AA, Gandomi AH, et al. RUN beyond the metaphor: an efficient optimization algorithm based on Runge Kutta method. Expert Syst Appl 2021;181:115079. https://doi.org/10.1016/j.eswa.2021.115079.
[17] Camacho-Villalón CL, Dorigo M, Stützle T. Exposing the grey wolf, moth-flame, whale, firefly, bat, and antlion algorithms: six misleading optimization techniques inspired by bestial metaphors. Int Tran Oper Res 2023;30(6):2945–71. https://doi.org/10.1111/itor.13176.
[18] Weyland D. A rigorous analysis of the harmony search algorithm: how the research community can be misled by a "novel" methodology. Int J Appl Metaheuristic Comput 2010;1(2):50–60. https://doi.org/10.4018/jamc.2010040104.
[19] Camacho-Villalón CL, Dorigo M, Stützle T. The intelligent water drops algorithm: why it cannot be considered a novel algorithm: a brief discussion on the use of metaphors in optimization. Swarm Intell 2019;13:173–92. https://doi.org/10.1007/s11721-019-00165-y.
[20] Camacho-Villalón CL, Stützle T, Dorigo M. Cuckoo search≡(+)-evolution strategy—A rigorous analysis of an algorithm that has been misleading the research community for more than 10 years and nobody seems to have noticed[R]. Technical Report TR/IRIDIA/2021-006. Belgium: IRIDIA, Université Libre de Bruxelles; 2021.
[21] Piotrowski AP, Napiorkowski JJ, Rowinski PM. How novel is the "novel" black hole optimization approach? Inform Sci 2014;267:191–200. https://doi.org/10.1016/j.ins.2014.01.026.
[22] Aranha C, Camacho Villalón CL, Campelo F, et al. Metaphor-based metaheuristics, a call for action: the elephant in the room. Swarm Intell 2022;16(1):1–6. https://doi.org/10.1007/s11721-021-00202-9.
[23] Li S, Chen H, Wang M, et al. Slime mould algorithm: a new method for stochastic optimization. Future Gener Comput Syst 2020;111:300–23. https://doi.org/10.1016/j.future.2020.03.055.
[24] Kudela J. The evolutionary computation methods no one should use. arXiv preprint 2023. https://doi.org/10.48550/arXiv.2301.01984. arXiv:2301.01984.
[25] Kudela J. A critical problem in benchmarking and analysis of evolutionary computation methods. Nat Mach Intell 2022;4(12):1238–45. https://doi.org/10.1038/s42256-022-00579-0.

[26] Kennedy J. Swarm intelligence[M]//Handbook of nature-inspired and innovative computing: Integrating classical models with emerging technologies. Boston, MA: Springer US; 2006. p. 187–219.
[27] Liu B, Wang L, Jin YH, et al. Improved particle swarm optimization combined with chaos. Chaos Solitons Fractals 2005;25(5):1261–71. https://doi.org/10.1016/j.chaos.2004.11.095.
[28] Kumar S, Yildiz BS, Mehta P, et al. Chaotic marine predators algorithm for global optimization of real-world engineering problems. Knowledge-Based Syst 2023;261:110192. https://doi.org/10.1016/j.knosys.2022.110192.
[29] Faramarzi A, Heidarinejad M, Mirjalili S, et al. Marine predators algorithm: a nature-inspired metaheuristic. Expert Syst Appl 2020;152:113377. https://doi.org/10.1016/j.eswa.2020.113377.
[30] Kaur G, Arora S. Chaotic whale optimization algorithm. J Comput Des Eng 2018;5(3):275–84. https://doi.org/10.1016/j.jcde.2017.12.006.
[31] Altay O. Chaotic slime mould optimization algorithm for global optimization. Artif Intell Rev 2022;55(5):3979–4040. https://doi.org/10.1007/s10462-021-10100-5.
[32] Raj S, Shiva CK, Vedik B, et al. A novel chaotic chimp sine cosine algorithm part-I: for solving optimization problem. Chaos Solitons Fractals 2023;173:113672. https://doi.org/10.1016/j.chaos.2023.113672.
[33] Mirjalili S. SCA: a sine cosine algorithm for solving optimization problems. Knowledge-Based Syst 2016;96:120–33. https://doi.org/10.1016/j.knosys.2015.12.022.
[34] Storn R, Price K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. J Global Optim 1997;11:341–59. https://doi.org/10.1023/A:1008202821328.
[35] Zhang S, Zhang H, Wang C. Memristor initial-boosted extreme multistability in the novel dual-memristor hyperchaotic maps. Chaos Solitons Fractals 2023;174:113885. https://doi.org/10.1016/j.chaos.2023.113885.
[36] Hua Z, Wu Z, Zhang Y, et al. Two-dimensional cyclic chaotic system for noise-reduced OFDM-DCSK communication. IEEE Trans Circuits Syst I Regul Pap 2024. https://doi.org/10.1109/TCSI.2024.3454535.
[37] Lai Q, Hu G. A nonuniform pixel split encryption scheme integrated with compressive sensing and its application in IoM. IEEE Trans Industr Inform 2024;20(9):11262–72. https://doi.org/10.1109/TII.2024.3403266.
[38] Lin H, Wang C, Cui L, et al. Brain-like initial-boosted hyperchaos and application in biomedical image encryption. IEEE Trans Industr Inform 2022;18(12):8839–50. https://doi.org/10.1109/TII.2022.3155599.
[39] Chua LO. Memristor-the missing circuit element. IEEE Trans Circuit Theory 1971;18(5):507–19. https://doi.org/10.1109/TCT.1971.1083337.
[40] Li Y, Li C, Lei T, et al. Offset boosting-entangled complex dynamics in the memristive Rulkov neuron. IEEE Trans Ind Electron 2024;71(8):9569–79. https://doi.org/10.1109/TIE.2023.3325558.
[41] Bao H, Su Y, Hua Z, et al. Grid homogeneous coexisting hyperchaos and hardware encryption for 2-D HNN-like map. IEEE Trans Circuits Syst I Regul Pap 2024;71(9):4145–55. https://doi.org/10.1109/TCSI.2024.3423805.
[42] Deng Y, Li Y. A 2D hyperchaotic discrete memristive map and application in reservoir computing. IEEE Trans Circuits Syst II Express Briefs 2022;69(3):1817–21. https://doi.org/10.1109/TCSII.2021.3118646.
[43] Bao H, Hua Z, Li H, et al. Discrete memristor hyperchaotic maps. IEEE Trans Circuits Syst I Regul Pap 2021;68(11):4534–44. https://doi.org/10.1109/TCSI.2021.3082895.
[44] Dong Y, Zhang H, Wang C, et al. An adaptive state transition algorithm with local enhancement for global optimization. Appl Soft Comput 2022;121:108733. https://doi.org/10.1016/j.asoc.2022.108733.
[45] Karaboga D, Basturk B. A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Global Optim 2007;39:459–71. https://doi.org/10.1007/s10898-007-9149-x.
[46] Tran TD, Jin GG. Real-coded genetic algorithm benchmarked on noiseless black-box optimization testbed[C]. In: Proceedings of the 12th annual conference companion on genetic and evolutionary computation; 2010. p. 1731–8.
[47] Liang JJ, Qin AK, Suganthan PN, et al. Comprehensive learning particle swarm optimizer for global optimization of multimodal functions. IEEE Trans Evol Comput 2006;10(3):281–95. https://doi.org/10.1109/TEVC.2005.857610.
[48] García-Martínez C, Lozano M, Herrera F, et al. Global and local real-coded genetic algorithms based on parent-centric crossover operators. Eur J Oper Res 2008;185(3):1088–113. https://doi.org/10.1016/j.ejor.2006.06.043.
[49] Qin AK, Huang VL, Suganthan PN. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE Trans Evol Comput 2008;13(2):398–417. https://doi.org/10.1109/TEVC.2008.927706.
[50] Tanabe R, Fukunaga AS. Improving the search performance of SHADE using linear population size reduction[C]. In: 2014 IEEE congress on evolutionary computation (CEC). IEEE; 2014. p. 1658–65.
[51] Kim S, Kojima M, Waki H, et al. Algorithm 920: SFSDP: a sparse version of full semidefinite programming relaxation for sensor network localization problems. ACM Trans Math Software 2012;38(4):1–19. https://doi.org/10.1145/2331130.2331135.
[52] Wang Z, Li C, Li Y, et al. A chaotic map with two-dimensional offset boosting. Chaos Interdiscip J Nonlinear Sci 2024;34(6):063130. https://doi.org/10.1063/5.0207875.
[53] Lai Q, Yang L, Chen G. Design and performance analysis of discrete memristive hyperchaotic systems with stuffed cube attractors and ultraboosting behaviors. IEEE Trans Ind Electron 2024;71(7):7819–28. https://doi.org/10.1109/TIE.2023.3299016.
[54] He S, Hu K, Wang M, et al. Design and dynamics of discrete dual-memristor chaotic maps and its application in speech encryption. Chaos Solitons Fractals 2024;188:115517. https://doi.org/10.1016/j.chaos.2024.115517.
[55] Zhang S, Ma P, Zhang H, et al. Dual memristors-radiated discrete hopfield neuron with complexity enhancement. Nonlinear Dyn 2024. https://doi.org/10.1007/s11071-024-10364-w.
[56] Wang Z, Li C, Li Y, et al. A class of memristive Hénon maps. Phys Scr 2024;99(10):105227. https://doi.org/10.1088/1402-4896/ad71fe.
[57] Zhao Q, Bao H, Zhang X, et al. Complexity enhancement and grid basin of attraction in a locally active memristor-based multi-cavity map. Chaos Solitons Fractals 2024;182:114769. https://doi.org/10.1016/j.chaos.2024.114769.