Mathematical Biosciences and Engineering

MBE, 20(9): 15737–15764.
DOI: 10.3934/mbe.2023701
Received: 28 May 2023
Revised: 28 June 2023
Accepted: 09 July 2023
Published: 28 July 2023

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

Research article

An improved particle swarm optimization combined with double-chaos search

Xuepeng Zheng, Bin Nie*, Jiandong Chen, Yuwen Du, Yuchao Zhang and Haike Jin

School of Computer, Jiangxi University of Chinese Medicine, Nanchang 330004, China

* **Correspondence:** Email: ncunb@163.com.

Abstract: Particle swarm optimization (PSO) has been successfully applied to various complex optimization problems due to its simplicity and efficiency. However, the update strategy of the standard PSO algorithm is to learn from the global best particle, making it difficult to maintain diversity in the population and prone to premature convergence due to being trapped in local optima. Chaos search mechanism is an optimization technique based on chaotic dynamics, which utilizes the randomness and nonlinearity of a chaotic system for global search and can escape from local optima. To overcome the limitations of PSO, an improved particle swarm optimization combined with double-chaos search (DCS-PSO) is proposed in this paper. In DCS-PSO, we first introduce double-chaos search mechanism to narrow the search space, which enables PSO to focus on the neighborhood of the optimal solution and reduces the probability that the swarm gets trapped into a local optimum. Second, to enhance the population diversity, the logistic map is employed to perform a global search in the narrowed search space and the best solution found by both the logistic and population search guides the population to converge. Experimental results show that DCS-PSO can effectively narrow the search space and has better convergence accuracy and speed in most cases.

Keywords: chaos optimization algorithm; particle swarm optimization; chaotic dynamics; optimization problem

1. Introduction

In the past few decades, many meta-heuristic algorithms have been proposed to handle complex optimization problems, which are usually difficult to solve with traditional optimization methods

relying on gradients. Meta-heuristic algorithms are problem-solving methods that mimic natural or abstract processes to efficiently explore solution spaces and find near-optimal solutions for complex optimization problems. Kennedy and Eberhart [1] developed particle swarm optimization algorithm (PSO), a population-based optimization algorithm inspired by social behavior and collective intelligence, where particles iteratively update their positions based on their own experience and the information from the best-performing particles in the swarm. Holland [2] proposed genetic algorithm (GA) that imitates the process of natural selection and genetic evolution to iteratively search for optimal solutions to complex problems. Wang et al. [3] proposed a monarch butterfly optimization algorithm (MBO) that imitates the migration behavior of monarch butterflies, where the search process is guided by a combination of exploration and exploitation strategies to find optimal solutions. Li et al. [4] presented a slime mold algorithm (SMA) that simulates the foraging behavior of slime mold to efficiently explore and exploit search spaces by forming networks of interconnected pathways. Wang [5] presented a moth search algorithm (MSA) that mimics the behavior of moths to navigate toward the optimal solution by adjusting their flight direction based on attraction and repulsion forces. Yang et al. [6] proposed a hunger games search (HGS) algorithm, where individuals compete for resources and survival. Ahmadianfar et al. [7] presented Runge-Kutta optimizer (RUN) that utilizes the logic of slope variations computed by the Runge-Kutta method to perform active exploration and exploitation for global optimization with an enhanced solution quality mechanism to avoid local optima and improve convergence speed. Tu et al. [8] proposed colony predation algorithm (CPA) that mimics the collective hunting behavior of predators to explore and exploit the solution space for optimization efficiently. Ahmadianfar et al. [9] presented INFO which utilizes the weighted mean concept, enhanced updating rules, vector combining and local search to optimize various problems. Heidari et al. [10] proposed Harris hawks optimization (HHO), which is inspired by the hunting behavior of Harris hawks, utilizing different strategies such as exploration, exploitation and cooperative hunting to solve optimization problems. Su et al. [11] proposed a RIME optimization algorithm, a nature-inspired metaheuristic algorithm based on the freezing process of water droplets in rime formation, utilizing dynamic freezing and melting mechanisms for efficient optimization. Compared with conventional algorithms, these meta-heuristic algorithms can handle non-differentiable, multimodal, hybrid and high-dimensional problems without requiring explicit gradient information.

After they are proposed, many researchers successfully solve complex optimization problems by using them in intelligent systems. Li et al. [12] introduced ACO-S, a bionic optimization algorithm-based dimension reduction method specifically designed for high-dimensional datasets like microarray data, demonstrating its ability to generate compact and informative gene subsets with high classification accuracy compared to existing bionic optimization algorithms. Thawkar et al. [13] presented a hybrid feature selection method, combining the butterfly optimization algorithm (BOA) and the ant lion optimizer (ALO) to effectively predict the benign or malignant status of breast tissue using mammogram images. Chakraborty et al. [14] developed a computational tool using a modified whale optimization method (WOA) to quickly and accurately determine the severity of COVID-19 illness through chest X-ray image analysis. Gao et al. [15] introduced a multi-objective optimization model and an improved genetic algorithm for cooperative mission assignment of heterogeneous UAVs, considering the balance between mission gains and UAV losses. Yu et al. [16] presented a multi-objective optimization problem and an improved genetic algorithm for cooperative mission planning of heterogeneous UAVs in cross-regional joint operations, considering makespan minimization, value expectation maximization, flexible base return and ammunition inventory constraints. Li et al. [17]

investigated the scheduling problem of maintenance and repair tasks for carrier-based aircraft fleets in hangar bays, proposing a model and an improved teaching-learning-based optimization algorithm to enhance maintenance efficiency and reduce manual scheduling burden.

Among the algorithms mentioned above, PSO is popular because of its fast convergence, few parameters and easy programming implementation. However, the performance of the standard PSO depends mainly on its parameter settings [18] and different parameter settings may lead to different convergence outcomes. In addition, the update strategy of particles during the search process is mainly aimed at the global best, which leads the population to lose diversity in the population and increases the risk of being trapped in local optima [19–21]. To address these drawbacks and improve the performance of PSO, many PSO variants have been developed. Generally, research on PSO variants can be classified into three parts: parameter modification, population topology structure modification and novel learning strategy.

1) Parameter modification. PSO has some parameters such as inertia weight and acceleration coefficients. They have been modified in many ways. Shi and Eberhart [22] introduced the concept of inertia weight in PSO to improve velocity control and search effectiveness. Afterward, various adjustment strategies were advocated, including defining it as a time-varying function called global PSO (GPSO) [23]. Liu et al. [24] proposed a PSO algorithm with chaos inertia weight to enlarge the search range and increase population diversity. Chen et al. [25] proposed a hybrid PSO with sine cosine acceleration coefficients where the performance of PSO is improved by introducing sine and cosine acceleration coefficients. By using these strategies, the performance of PSO can be improved to some extent.

2) Population topology structure modification. In the standard PSO, the population size is fixed. A larger population size may increase computation costs, resulting in slow convergence, while a smaller size may converge in local optima. Some researchers have improved the performance of PSO by modifying the population topology structure. Kennedy and Mendes [26] proposed the local version of PSO (LPSO) with ring topology, in which different topologies based on neighborhood connections are used to maintain the diversity of the population. Liang and Suganthan [27] presented a dynamic multi-swarm PSO (DMS-PSO), in which the population comprises many small subgroups and a dynamic strategy is used to maintain population diversity.

3) Novel learning strategy. A well-designed learning strategy can enhance the efficiency and effectiveness of the PSO algorithm. Liang et al. [28] proposed the CLPSO, in which each particle updates its velocity by incorporating historical best information from different particles. This approach enables particles to acquire valuable information and improves their search behavior. Wang et al. [29] proposed a PSO-HLM algorithm that balances exploration and exploitation by utilizing a hybrid learning model and a multi-pools fusion strategy, improving convergence speed and avoiding local optima. Zhou et al. [30] proposed a LFIACL-PSO algorithm that enhances PSO by incorporating Lévy flight-based inverse learning, comprehensive learning strategy, ring-type topology and adaptive update of acceleration coefficients.

In this paper, an improved PSO combined with double-chaos search (DCS-PSO) is proposed. Chaotic dynamics is a branch of nonlinear dynamical systems. Chaos motion exhibits ergodicity, randomness and regularity properties and can traverse all states without repetition. For optimization problems, these properties of chaos can be used as a mechanism to avoid getting trapped in local optima during the search process. In DCS-PSO, we utilize two distinct chaotic systems to explore the search space globally and narrow it down, enabling PSO to focus on the vicinity of the optimal solution and

alleviate its premature convergence caused by getting trapped in local optima. Additionally, we incorporate a chaotic motion search in parallel with the PSO search. If the solution obtained through the chaotic search outperforms the PSO's global best, it will be replaced with the solution from the chaotic motion to enhance population diversity. This strategy prevents all particles in the PSO from relying solely on the global optimal particle for learning, thereby facilitating escape from local optima.

The main contributions of this paper are described as follows.

1) Introducing a novel approach, DCS-PSO, which combines the strengths of PSO and chaotic dynamics.

2) Addressing the issue of premature convergence by narrowing down the search space using double-chaos search and enhancing population diversity through the logistic map, resulting in improved convergence accuracy and speed.

3) Conducting extensive experimental evaluations to demonstrate the effectiveness and superiority of DCS-PSO compared to standard PSO and other optimization algorithms commonly used in the field.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 describes the proposed DCS-PSO. Section 4 is the experimental studies. Conclusion and future work are provided in Section 5.

2. Related works

2.1. Standard PSO

PSO is an optimization algorithm based on swarm intelligence inspired by collective behavior in ecosystems such as bird foraging. In PSO, the problem is treated as an optimization problem of finding the optimal solution in a D-dimensional search space, where each particle i in the population represents a possible solution and has two vectors: a position vector $X_i = \left[x_{i,1}, x_{i,2}, \cdots, x_{i,D} \right]$ and a velocity vector $V_i = \left[v_{i,1}, v_{i,2}, \cdots, v_{i,D} \right]$. In searching for the optimal solution, particles move and interact with each other in the search space. The particle i adjusts its position based on the optimal position vector $\boldsymbol{pbest_i} = \left[pbest_{i,1}, pbest_{i,2}, \cdots pbest_{i,D} \right]$ it has reached in the past and the globally optimal position vector $\boldsymbol{gbest} = \left[gbest_1, gbest_2, \cdots, gbest_D \right]$ discovered by the entire swarm. PSO attempts to find a balance between local search and global search in this way. The update rules for the velocity and position of particle i in the d th dimension are as follows:

$$v_{i,d}^{(n+1)} = v_{i,d}^{(n)} + c_1 \times r_1 \times \left(pbest_{i,d} - x_{i,d} \right) + c_2 \times r_2 \times \left(gbest_d - x_{i,d} \right), \tag{2.1}$$

$$x_{i,d}^{(n+1)} = x_{i,d}^{(n)} + v_{i,d}^{(n+1)} \tag{2.2}$$

where c_1 and c_2 denote the acceleration coefficients, usually set to 2, r_1 and r_2 are two uniformly distributed values in the range [0,1]. The parameters $V_{\min}$ and $V_{\max}$ can be used to determine an appropriate range of velocity values for each particle to prevent it from wandering too far outside the search space.

The global minimum optimization problem (2.3) for a D-dimensional continuous object can be described as:

$$\begin{aligned} &\min \quad f(X),\ X=[x_1,\cdots,x_D],\\ &\text{s.t.} \quad x_d \in [a_d,b_d],\ d=1,\cdots,D \end{aligned} \tag{2.3}$$

The standard PSO can be described in the following steps:

Step 1: Initialize N particles. For each particle i in the population:

Step 1.1: Initialize X_i randomly in the range $[\boldsymbol{a},\boldsymbol{b}]$.

Step 1.2: Initialize V_i randomly in the range $[V_{\min},V_{\max}]$.

Step 1.3: Evaluate the fitness value f_i.

Step 1.4: Set $\boldsymbol{pbest_i}$ as the particle's current position and $\boldsymbol{gbest}$ as the particle with the best fitness value in the population.

Step 2: Loop until the stop condition is satisfied:

Step 2.1: Update the velocity V_i and position X_i for each particle i according to Eqs (2.1) and (2.2).

Step 2.2: Evaluate the fitness value f_i of each particle i.

Step 2.3: Update $\boldsymbol{pbest_i}$ *if* $f_i < f_{\boldsymbol{pbest_i}}, \forall i \le N$ for each particle i.

Step 2.4: Update the global optimal position $\boldsymbol{gbest}$ $f_{\boldsymbol{gbest}} \le f_i,\ \forall i \le N$.

2.2. Chaos algorithms

2.2.1. Chaos optimization algorithm

Chaos is one of the most exciting properties exhibited by the brain [31]. It is the phenomenon of complex, unpredictable and random-like behavior arising from simple deterministic nonlinear systems rather than a state of disorder [32]. Researchers have recently developed many optimization algorithms based on chaos theory to address complex optimization problems [33–35].

COA was first proposed by Li and Jiang [32,35] in 1997. Its core idea is to introduce the chaotic state into optimization variables via the carrier wave method, map the traversal range of chaos motion to the value range of optimization variables and then search for the global optimal solution using chaotic dynamics instead of random search. The chaotic dynamics equation chosen in COA is the Logistic map [36], which is defined as follows:

$$x^{(n+1)} = \mu x^{(n)}\left(1-x^{(n)}\right),\quad 0 \leqslant x^{(0)} \leqslant 1 \tag{2.4}$$

where μ is the control parameter, $n=0,1,2,\ldots$. Although the above equation is definite, it exhibits chaotic dynamics when $\mu=4$ and $x^{(0)} \notin \{0,0.25,0.5,0.75,1\}$, the property that a minute difference in the initial value of the chaotic variable would lead to significant differences in output. This is the sensitive dependence on initial conditions, the basic characteristic of chaos. This property enables the trajectory of chaotic variables to traverse the entire search space. The process of COA can be defined through the following equation.

$$cx_d^{(k+1)} = 4cx_d^{(k)}\left(1-cx_d^{(k)}\right),\quad d=1,2,\cdots,D, \tag{2.5}$$

where cx_d is the d th chaotic variable and k represents the number of iterations. Obviously, $cx_d^{(k)}$

is distributed in the range (0,1) when the initial $cx_d^{(0)} \in (0,1)$ and $cx_d^{(0)} \notin \{0.25, 0.5, 0.75\}$.

For the optimization problem (2.3), the procedures of COA are as follows.

Step 1: Set $k = 0$, and initialize D chaotic variables $cx_d^{(0)}$ in the range (0,1) with very small differences. The current optimal function value f^* is initialized to a large value, with $X^* = \left[x_1^*, x_2^*, \cdots x_D^*\right]$ representing the corresponding optimal variables.

Step 2: Map the D chaotic variables cx_d to the D optimization variables x_d of the optimization problem (2.3) through the carrier wave method of Eq (2.6).

$$x_d^{(k)} = cx_d^{(k)}\left(b_d - a_d\right) + a_d \tag{2.6}$$

Step 3: Iterative search using chaotic variables.

If $f\left(X^{(k)}\right) \le f^*$, then $f^* = f\left(X^{(k)}\right)$, $X^* = X^{(k)}$. Else if $f\left(X^{(k)}\right) > f^*$, then give up $X^{(k)}$.

Step 4: Let $k = k + 1$ and continue the chaotic traversal through Eq (2.5).

Step 5: Repeat steps 2 to 4. If the best function value f^* remains unchanged after several iterations, the second carrier wave is applied according to Eq (2.7).

$$x_d^{(k+1)} = x_d^* + \alpha \cdot cx_d^{(k+1)} \tag{2.7}$$

where x_d^* is the current optimal solution, α is an adjusting constant which can be less than 1 and $\alpha \cdot cx_d^{(k+1)}$ generates i chaotic states with small ergodic ranges around x_d^*.

Step 6: Continue the iterative search using the chaotic variables after the second carrier wave until the stop criterion is satisfied.

2.2.2. Some representative variants of COA

As a simple and efficient optimization algorithm, COA has garnered the attention of many researchers for further improvement. Its primary disadvantage is that the sensitive dependence of chaotic motion on the initial values will cause the algorithm's instability and a large number of traversal searches are required before the search space is reduced. Moreover, the lack of an explicit search stop condition results in the need for parameter tuning according to specific problems, limiting the algorithm's generality. Therefore, many variants of COA aimed at these problems have emerged.

Xiu et al. [37] proposed a double-chaos optimization algorithm (DCOA) that uses two different chaotic systems to explore the search space independently and in parallel and narrows down the search space when the distance between the optimal positions of the two systems satisfies specific criteria. Therefore, DCOA overcomes some of the deficiencies of COA. Additionally, DCOA provides a condition for narrowing the search space, which improves the algorithm's generality, avoids multiple blind searches and reduces its running cost. Liang and Gu [38] developed a chaos optimization algorithm based on parallel computing (PCOA), which performs parallel searches using several different sets of chaotic variables, effectively reducing the sensitive dependence of chaotic motion on initial values.

3. Proposed method

Based on in-depth research of the chaos algorithms and the PSO algorithm, we combined the

advantages of the two types of algorithms and proposed DCS-PSO. The details are as follows:

3.1. Space reduction based on double-chaos search mechanism

Although a single chaos search mechanism is easy to implement and can avoid being trapped in local optima, it often requires a large number of iterations to narrow down the search space to ensure that the optimal solution is within the narrowed area.

Additionally, the stop condition of the search needs to be adjusted based on the specific problem, limiting the algorithm's generality. In contrast, the DCOA determines the requirement for narrowing the space based on the distance between the optimal positions found by each chaotic system, which improves the algorithm's generality. However, there is a risk that the algorithm may continuously narrow down the search space, making it difficult to converge accurately to a point and potentially excluding the global optimal solution from the search space. Nonetheless, experimental verification (see Section 4.1) suggests that the double-chaos search mechanism can narrow the search space to the neighborhood of the global optimum after only one sufficient search.

In the double-chaos search mechanism, we select two chaotic systems: The logistic map and the tent map. The orbit points of the logistic map are distributed near the edges, which to some extent can explain why chaotic motion has the advantage of escaping from local optima [39]. However, several breakpoints $\{0, 0.25, 0.5, 0.75, 1\}$ in the logistic map make it difficult to traverse specific chaos states during the search process. The orbit points of the tent map are distributed more evenly. Its equation is defined in Eq (3.1). Figure 1(a) and (b) show the scatter plots of the two chaos maps for 2000 iterations, respectively and (c) and (d) show their chaotic dynamics.

$$x^{(n+1)} = \begin{cases} x^{(n)} / \beta, & 0 < x^{(n)} \leqslant \beta \\ \left(1 - x^{(n)}\right) / \left(1 - \beta\right), & \beta < x^{(n)} \leqslant 1 \end{cases} \tag{3.1}$$

where $\beta = 0.4$.

The double-chaos search mechanism is illustrated in Figure 2. In the search space A, two chaotic systems $\boldsymbol{cx}$ and $\boldsymbol{cy}$ are used to perform independent parallel searches. When the distance between the optimal solutions $\boldsymbol{cx}^*$ and $\boldsymbol{cy}^*$ found by $\boldsymbol{cx}$ and $\boldsymbol{cy}$ is small enough, namely,

$$\left\| X^* - Y^* \right\|_2 < \gamma \left\| \boldsymbol{a} - \boldsymbol{b} \right\|_2 \tag{3.2}$$

where X^* and Y^* represent the corresponding optimization variables, $\gamma \in (0, 0.25]$ and is set to 0.15 in this study, the search space is narrowed down from A to B according to Eqs (3.3) and (3.4).

$$a_d = \max\left(a_d, \left(\min\left(x_d^*, y_d^*\right) - \xi \cdot \gamma \cdot \left\| X^* - Y^* \right\|_2\right)\right) \tag{3.3}$$

$$b_d = \min\left(b_d, \left(\max\left(x_d^*, y_d^*\right) + \xi \cdot \gamma \cdot \left\| X^* - Y^* \right\|_2\right)\right) \tag{3.4}$$

where $\xi \in [1, 2]$ and it is set to 1.5 in this study.

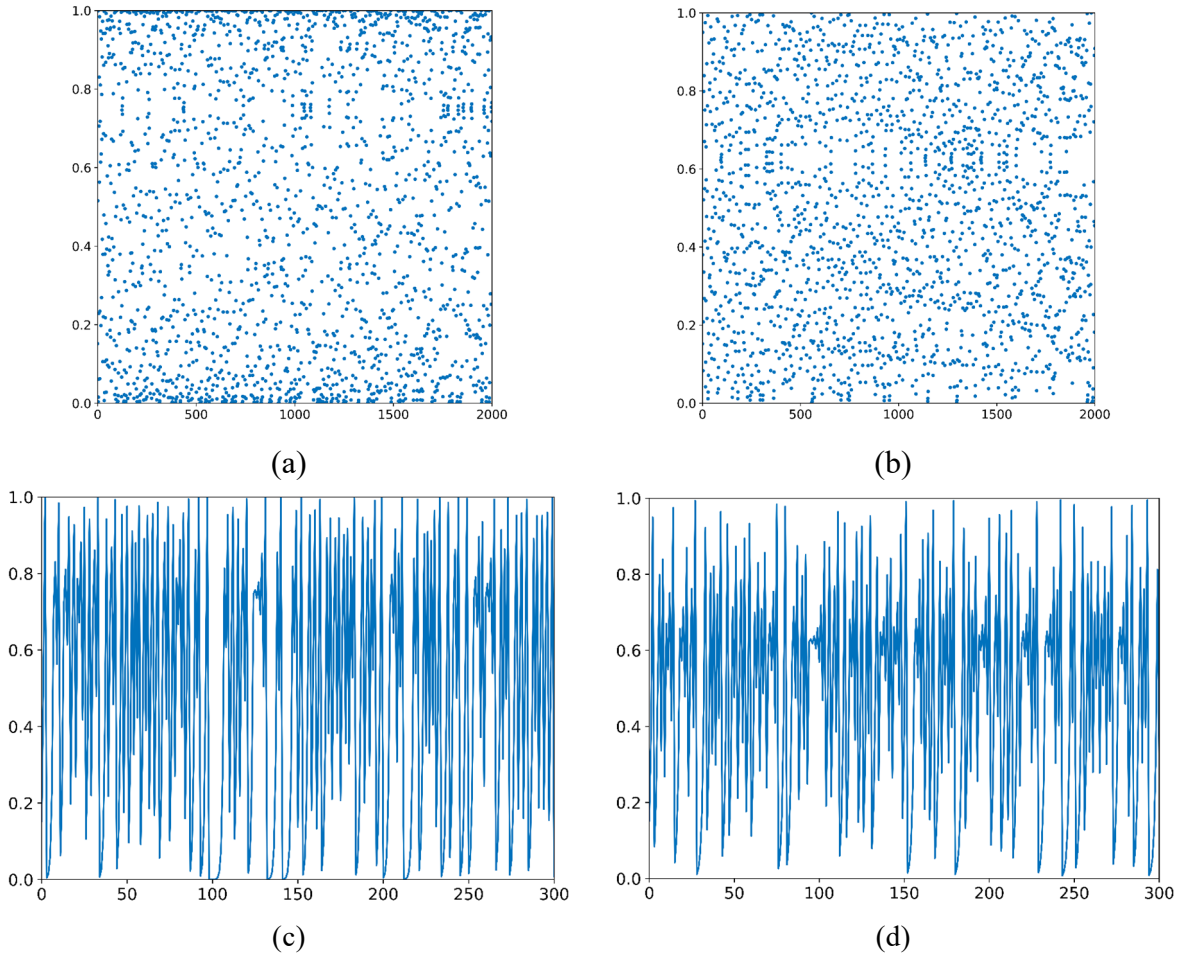


Figure 1. Scatter plots and chaotic dynamics diagrams of two types of chaos maps. (a)Logistic map. (b)Tent map. (c)Logistic dynamics. (d)Tent dynamics.

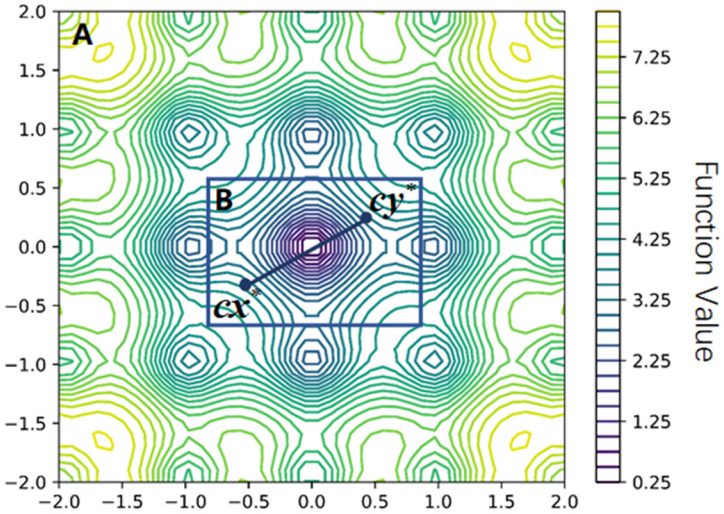


Figure 2. Diagram of the double-chaos search mechanism with the contour lines of a 2-dimensional multimodal function projected onto the XOY plane in the background.

3.2. An improved particle swarm optimization combined with double-chaos search

Based on the chaos search mechanism, we propose a two-stage iteration strategy named DCS-PSO. The algorithm employs the double-chaos search mechanism for global exploration and narrowing the search space and then uses the improved PSO for further fine search in the narrowed search space.

For the D-dimensional optimization problem (2.3), the detailed steps of the first stage of DCS-PSO are listed in Algorithm 1:

Algorithm 1 Space reduction based on double-chaos search mechanism.

1. Initialize two chaotic systems $\boldsymbol{cx}^{(0)}$ and $\boldsymbol{cy}^{(0)}$ in the range (0,1);
2. Initialize the optimal function values of the two chaotic systems f_x^* and f_y^* with large values;
3. Initialize the corresponding optimization variables X^* and Y^*;
4. set $k=0$;
5. **while** True **do**:
6. Map $\boldsymbol{cx}^{(k)}$ and $\boldsymbol{cy}^{(k)}$ to optimization variables $X^{(k)}$ and $Y^{(k)}$;
7. Update $f_x^*=f\left(X^{(k)}\right)$ and $X^*=X^{(k)}$, if $f\left(X^{(k)}\right)<f_x^*$;
8. Update $f_y^*=f\left(Y^{(k)}\right)$ and $Y^*=Y^{(k)}$, if $f\left(Y^{(k)}\right)<f_y^*$;
9. **if** $k>h$ **then**
10. **if** $\left\|X^*-Y^*\right\|_2<\gamma\left\|\boldsymbol{a}-\boldsymbol{b}\right\|_2$ **then**
11. Narrow down the search space $[\boldsymbol{a},\boldsymbol{b}]$ according to Eqs (3.3) and (3.4);
12. **break**
13. **end if**
14. **end if**
15. Update $\boldsymbol{cx}^{(k)}$ and $\boldsymbol{cy}^{(k)}$ according to Eqs (2.5) and (3.1), respectively;
16. $k++$;
17. **end while**
18. Output the narrowed search space $\boldsymbol{a}=[a_1,a_2,\cdots a_d],\boldsymbol{b}=[b_1,b_2,\cdots,b_d]$
19. Output ***cbest***, one of X^* and Y^* that has the smallest fitness value

The flowchart of the first stage of DCS-PSO is shown in Figure 3(a).

In conventional PSO, all particles learn from the global optimal particle in the population to update their positions and velocities. This mechanism is advantageous in achieving fast convergence, but it also leads to the need for more diversity and often causes premature convergence due to falling into local optimum [40]. The conventional PSO may require longer to find the optimal solution when dealing with multimodal optimization problems with a large search space.

Although the risk of being trapped in local optima can be reduced by narrowing the search space, PSO may still fall into a local optimum in the reduced search space. To enhance the population's diversity, in the narrowed search space, we perform a chaos search on the best solution found by the two chaotic systems using the logistic map. The best solution found by the logistic map and the

population guides the algorithm to converge toward the global optimum. The velocity of particle i on dimension d is updated according to the following rules.

$$\boldsymbol{best} = \begin{cases} \boldsymbol{cbest}, & if\ f\left(\boldsymbol{cbest}\right) \leqslant f\left(\boldsymbol{gbest}\right) \\ \boldsymbol{gbest}, & else \end{cases} \tag{3.5}$$

$$v_{i,d}^{(n+1)} = w \times v_{i,d}^{(n)} + c_1 \times r_1 \times \left(pbest_{i,d} - x_{i,d} \right) + c_2 \times r_2 \times \left(best_d - x_{i,d} \right) \tag{3.6}$$

where $\boldsymbol{best}$ denotes the best solution between chaos search and population search, $f\left(\boldsymbol{cbest}\right)$ is the optimal fitness value found by chaos search and $\boldsymbol{cbest}$ is the corresponding solution.

The detailed steps of the second stage of DCS-PSO are listed in Algorithm 2:

Algorithm 2 Improved PSO combined with chaos

1. Input the narrowed search space $\left[\boldsymbol{a},\boldsymbol{b}\right]$ and $\boldsymbol{cbest}$
2. Transform $\boldsymbol{cbest}$ to chaotic variables $\boldsymbol{cx}$;
3. Initialize N particles' position and velocity within the new search space;
4. Evaluate the fitness value of every particle;
5. Update $\boldsymbol{pbest}_i$ and $\boldsymbol{gbest}$;
6. **for** $i = 1:\ \mathrm{Max\_Iter}$ **do**
7. Implement chaos search globally and update $\boldsymbol{cbest}$ via logistic map;
8. **if** $f\left(\boldsymbol{cbest}\right) \le f\left(\boldsymbol{gbest}\right)$ **then**
9. $\boldsymbol{best} = \boldsymbol{cbest}$;
10. **else** $\boldsymbol{best} = \boldsymbol{gbest}$;
11. **end if**
12. Update particles' velocity and position according to Eqs (3.6) and (2.2), respectively;
13. Evaluate the fitness value of every particle;
14. Update $\boldsymbol{pbest}_i$ and $\boldsymbol{gbest}$;
15. **end for**

The flowchart of the second stage of DCS-PSO is shown in Figure 3(b).

It is worth noting that many researchers have attempted to introduce the chaos search mechanism into intelligent optimization algorithms [39,41]. The most common strategy is to use chaotic local search (CLS) after each iteration to conduct a fine search near the current optimum. Although this approach improves the convergence accuracy of the algorithm, it needs to address the problem of the population falling into local optima. Therefore, in addition to CLS, other strategies are often required to maintain the diversity of the population. However, some of these strategies may sacrifice the performance of convergence speed.

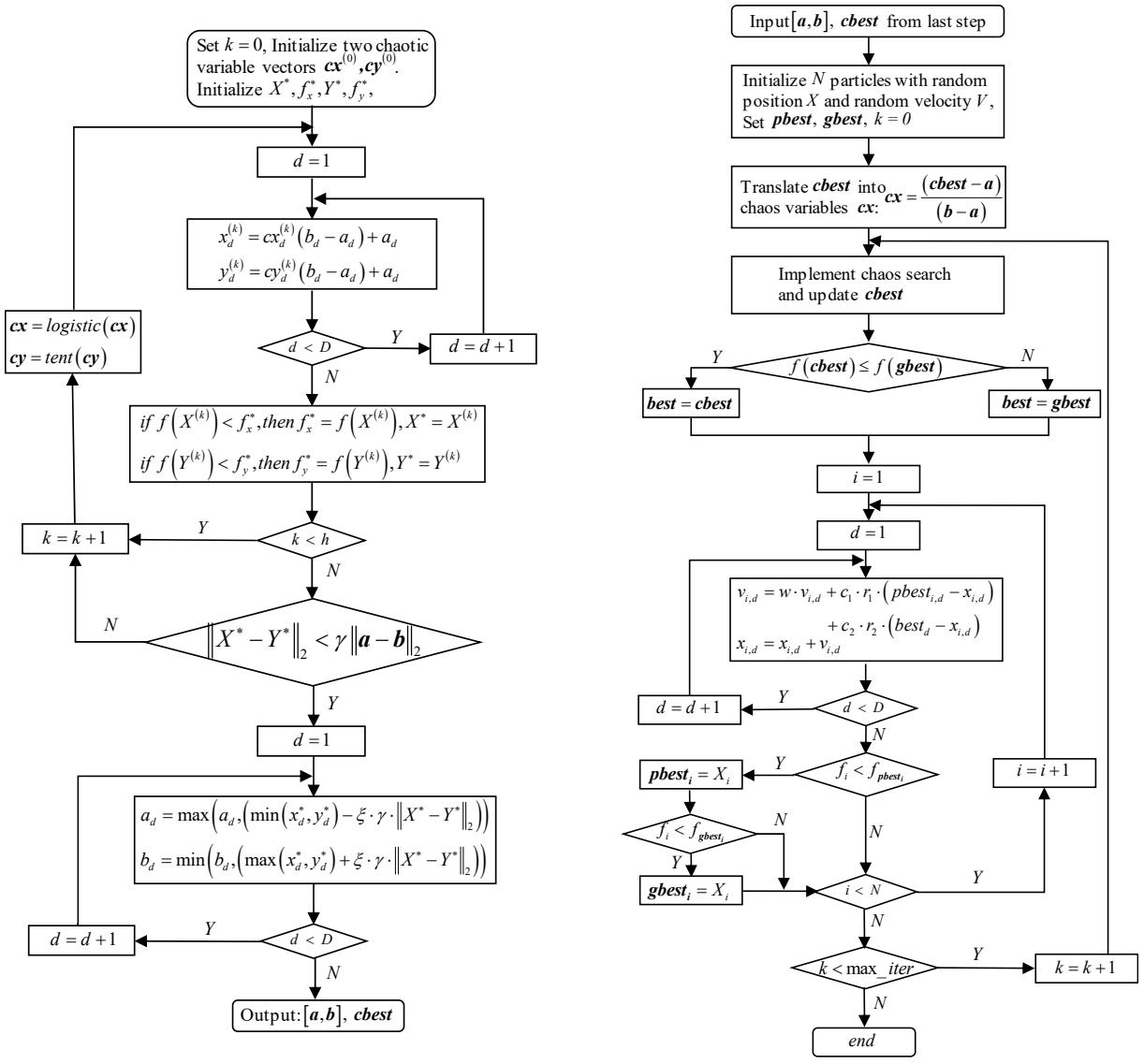


Figure 3. The flowcharts of DCS-PSO. (a) The first stage. (b) The second stage.

4. Experimental studies

In this chapter, simulation experiments are conducted to validate the performance of DCS-PSO. We first conduct experiments using six 2-dimensional benchmark functions to validate the effectiveness of the DCS-PSO search space narrowing mechanism. After that, the performance of the DCS-PSO algorithm is compared with five typical PSO variants placed on ten benchmark functions in terms of solution accuracy, statistical significance test and convergence speed. Finally, to verify the performance of DCS-PSO in complex and multimodal conditions, an authoritative test suite CEC2017 is selected in this paper.

All simulation experiments were conducted on the same PC. The configuration information of the PC is shown in Table 1.

Table 1. The computer configuration.

PC configuration	Information
CPU	Inter(R) Core(TM) i7-8700
Frequency	3.2GHz CPU
RAM	16.0GB
Operating system	Windows 10(64Bit)
Language	Python 3.9.13

4.1. Experiments on DCS-PSO and other chaos algorithms

4.1.1. Six 2-Dim benchmark functions

To validate the effectiveness of the search space narrowing mechanism in DCS-PSO, we conducted experiments using six 2-dimensional benchmark functions [38,42]. DCS-PSO was compared with other chaos-based algorithms, namely COA, DCOA and PCOA. The narrowed search space can be visually observed through the contour map of the 2-dimensional function. The six benchmark functions are listed below.

(1) Jong function:

$$f_J = 100\cdot\left(x_1^2 - x_2\right)^2 + \left(1 - x_1\right)^2, \quad -2.048 \le x_1, x_2 \le 2.048. \tag{4.1}$$

The function has a global minimum of 0 at (1,1).

(2) Camel function:

$$f_C = \left(4 - 2.1\cdot x_1^2 + \frac{x_1^4}{3}\right)\cdot x_1^2 + x_1\cdot x_2 + \left(-4 + 4\cdot x_2^2\right)\cdot x_2^2, \quad -2 \le x_1, x_2 \le 2. \tag{4.2}$$

The function has a global minimum of −1.031628 at two points (−0.0898, 0.7126) and (0.0898, −0.7126).

(3) GP-Goldstein-Price:

$$\begin{aligned} f_G(x) = &\left[1 + \left(x_1 + x_2 + 1\right)^2\left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right] \\ &\times\left[30 + \left(2x_1 - 3x_2\right)^2\left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right], \quad -2 \le x_1, x_2 \le 2. \end{aligned} \tag{4.3}$$

The function has a global minimum of 3 at (0,−1) and there are four local minima in the minimization region.

(4) BR-Branin:

$$f_B(x) = \left(x_2 - \frac{5.1}{4\pi^2}x_1^2 + \frac{5}{\pi}x_1 - 6\right)^2 + 10\left(1 - \frac{1}{8\pi}\right)\cos x_1 + 10, \quad -5 \le x_1 \le 10, \ \ 0 \le x_2 \le 15. \tag{4.4}$$

The global minimum of this function is approximately equal to 0.398 at three points (−3.142, 12.275), (3.142, 2.275) and (9.425, 2.425).

(5) RA-Rastrigin:

$$f_R(x) = x_1^2 + x_2^2 - \cos 18x_1 - \cos 18x_2, \quad -1 \le x_1, x_2 \le 1 \tag{4.5}$$

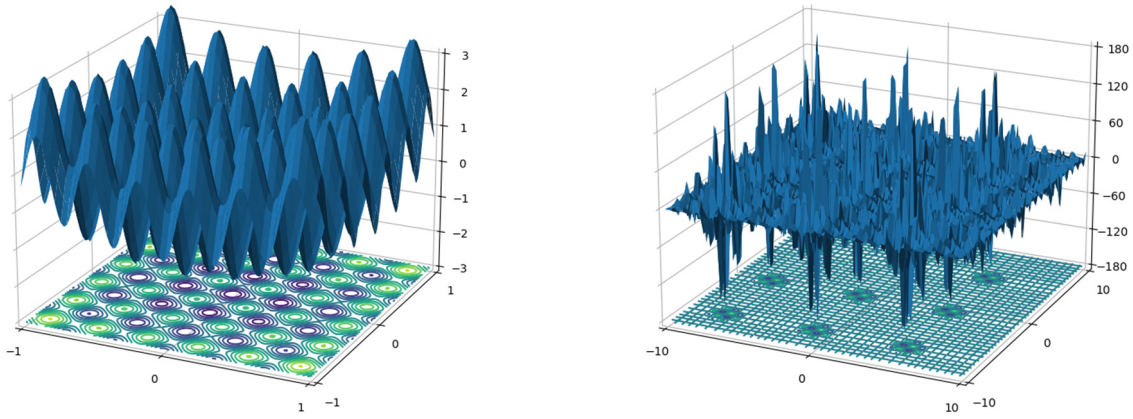
The function has a global minimum of −2 at (0,0) and its lattice structure has about 50 local minima.

(6) SH-Shuber:

$$f_s(x) = \left\{\sum_{i=1}^{5} i\cos\left((i+1)x_1 + i\right)\right\}\left\{\sum_{i=1}^{5} i\cos\left((i+1)x_2 + i\right)\right\}, \quad -10 \le x_1, x_2 \le 10. \tag{4.6}$$

The function has 760 local minima, 18 of which are global with a value of −186.7309.

The RA-Rastrigin function and the SH-Shuber function are relatively complex and have a large number of local minima, which can effectively test the algorithm's ability to escape from the local optima. Their 3D stereograms are shown in Figure 4.



(a) (b)

Figure 4. The 3D stereograms of the RA-Rastrigin function and the SH-Shuber function. (a) RA-Rastrigin function. (b) SH-Shuber function.

4.1.2. Experimental strategies

To ensure the fairness of the experimental results, each algorithm will run on each benchmark function independently 50 times and the mean execution time, the mean optimal values and the standard deviations will be provided. The specific parameter settings of all algorithms are listed in Table 2.

Table 2. Parameters settings of the four algorithms.

Algorithm	Parameters
COA	$\max\_iter1 = 10^5,\ \max\_iter2 = 1.5\times10^5,\ \alpha = 0.3$
DCOA	$h = 3000,\ \gamma = 0.25,\ \xi = 1.5$
PCOA	$n\_group = 3,\ \max\_iter1 = 10^5,\ \max\_iter2 = 1.5\times10^5,\ \alpha = 0.3$
DCS-PSO	$N = 20,\ w = 0.4,\ c1 = c2 = 2,\ V_{\max} = 0.2\times Range,\ \max\_iter = 1000$

4.1.3. Results and discussion

Table 3 shows the execution time (E.T.), the mean optimal values (Mean) and standard deviations (S.D.) of COA, DCOA, PCOA and DCS-PSO. The best results of each algorithm on each test function are indicated in bold and the second-best results are marked with an underline.

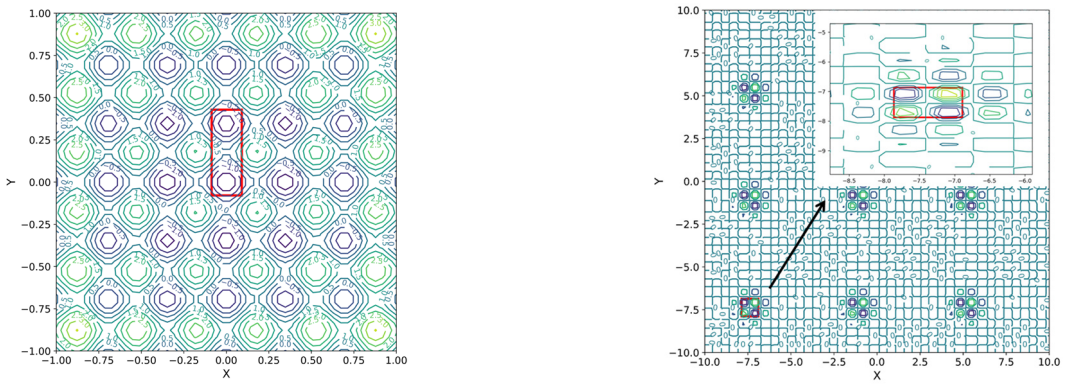
Table 3. Test results of the four algorithms.

Function	Criteria	COA	DCOA	PCOA	DCS-PSO
	E.T.	30.8898	0.5753	17.2234	**0.2793**
f_J	Mean	**0.0000**	0.0005	**0.0000**	**0.0000**
	S.D.	**0.0000**	0.0014	**0.0000**	**0.0000**
	E.T.	7.6564	0.6478	10.3990	**0.3610**
f_C	Mean	**−1.0316**	−1.0312	**−1.0316**	**−1.0316**
	S.D.	**0.0000**	0.0012	**0.0000**	**0.0000**
	E.T.	14.1867	1.0656	14.0814	**0.6097**
f_G	Mean	3.0022	3.0100	3.0024	**3.0000**
	S.D.	0.0022	0.0126	0.0022	**0.0000**
	E.T.	1.5663	0.8653	5.7455	**0.4513**
f_B	Mean	0.3983	0.4003	**0.3980**	0.3979
	S.D.	0.0003	0.0039	0.0001	**0.0000**
	E.T.	4.4150	0.6652	5.8342	**0.4366**
f_R	Mean	−1.9993	−1.9652	−1.9992	**−2.0000**
	S.D.	0.0006	0.0558	0.0008	**0.0000**
	E.T.	5.8609	2.1743	13.8397	**2.0615**
f_S	Mean	−186.6428	-186.4517	−186.6861	**−186.7309**
	S.D.	0.0711	0.4203	0.0680	**0.0000**

Based on the mean values in Table 3, the proposed DCS-PSO outperforms the other three chaos-based methods. Compared with COA, DCOA and PCOA, DCS-PSO achieves superior results on five functions (f_J, f_C, f_G, f_R and f_S). COA performs well in solving simple optimization problems and acquires the theoretically optimal solutions on both functions f_J and f_C. However, it performs poorly on function f_S, which has multiple local minima, possibly due to the challenge of attaining certain chaos states in a large search space. The convergence accuracy of DCOA is relatively poor. As mentioned earlier, DCOA may exclude the optimal solution while continuously narrowing the search space. However, compared with COA and PCOA, DCOA shows a shorter convergence time, which reflects its efficiency in narrowing the search space. The overall performance of PCOA is comparable to COA and it is the only algorithm that converges to the global optimum on function f_B. DCS-PSO has a faster convergence speed due to the combination of double-chaos search and PSO. The S.D. values in Table 3 show that DCS-PSO has the smallest S.D. values among the six benchmark functions, indicating that DCS-PSO is more stable and reliable.

Figure 5(a) and (b) show the contour plots of functions f_R and f_S projected onto the XOY plane, respectively. The red rectangle in each plot represents the narrowed search space, demonstrating the ability of DCS-PSO to reduce the search space of complex optimization problems. These plots

visually display that DCS-PSO can accurately converge to the vicinity of the global optimum, validating the effectiveness of the proposed method.



(a) (b)

Figure 5. Diagrams of DCS-PSO narrow the search space in the RA-Rastrigin and the SH-Shuber functions. (a) RA-Rastrigin function. (b) SH-Shuber function.

4.2. Experiments on DCS-PSO and other PSO variants

In this subsection, five representative PSO variants are selected for comparison. The first one is the PSO with inertia weight (PSO-w) [22], which can better control the search process and is characteristic of fast convergence when dealing with unimodal problems. The second one is the global version of PSO (GPSO) [23], whose inertia weight is decreased linearly from 0.9 to 0.4, improving the population's global search ability. The third one is the local version of PSO (LPSO) [26] with ring topology, in which different topologies based on neighborhood connections are used to maintain the diversity of the population. The fourth one is the Comprehensive Learning PSO (CLPSO) [28], in which each particle updates its velocity by learning the historical best information of different particles. The fifth is the dynamic multi-swarm PSO (DMS-PSO) [27], in which the population comprises many small subgroups and a dynamic strategy is used to maintain population diversity. CLPSO and DMS-PSO are designed to improve the performance of PSO on multimodal problems.

4.2.1. Benchmark functions

To investigate the performance of DCS-PSO on different optimization problems, ten benchmark functions are tested. Based on the properties of these functions, they are divided into two groups. The first group consists of 5 unimodal benchmark functions with only one global optimum value, mainly used to test the convergence accuracy and speed. The second group consists of 5 multimodal benchmark functions used to test the algorithm's global search ability and its ability to escape from local optima. These benchmark functions are widely used in the literature [25,43,44]. The details of these benchmarks are shown in Table 4.

In the first group, the global optimal position of function f_4 is $[1]^D$.The global optimal position of the remaining four functions is $[0]^D$ and the theoretical optimal values of the five functions are 0. f_1 is the sphere function and is easy to solve. f_2 and f_3 are the Schwefel functions. f_4 is the

Rosenbrock function, it becomes a simple multimodal problem when its dimension is more than three and has a narrow gap between the perceived local optima and the global optimum. f_5 is the noise function, which is challenging to find the global optimum because there is a perturbation factor.

In the second group, the global optimal positions of all functions are $[0]^D$ and the global optimal values are 0. Rastrigin's function (f_6) has a large number of local optima. f_7 has a narrow global basin and a large number of local optima. f_8 is the Griewank's function. f_9 and f_{10} are the penalized functions.

In f_9, the y_i is updated by $y_i = 1 + \frac{1}{4}(x_i + 1)$.

In f_9 and f_{10}, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i - a)^m, & x_i < -a \end{cases}$.

Table 4. Benchmark functions.

Name	Functions	Range	Optimum
Sphere	$f_1(\boldsymbol{x}) = \sum_{i=1}^{D} x_i^2$	[−100, 100]	0
Schwefel's 2.22	$f_2(\boldsymbol{x}) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	[−10, 10]	0
Schwefel's 1.2	$f_3(\boldsymbol{x}) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2$	[−100, 100]	0
Rosenbrock	$f_4(\boldsymbol{x}) = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + (x_i - 1)^2 \right]$	[−10, 10]	0
Noise	$f_5(\boldsymbol{x}) = \sum_{i=1}^{D} i x_i^4 + random[0,1)$	[−1.28, 1.28]	0
Rastrigin	$f_6(\boldsymbol{x}) = \sum_{i=1}^{D} \left(x_i^2 - 10\cos(2\pi x_i) + 10 \right)$	[−5.12, 5.12]	0
Ackley	$f_7(\boldsymbol{x}) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2} \right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D} \cos(2\pi x_i) \right) + 20 + e$	[−32, 32]	0
Griewank	$f_8(\boldsymbol{x}) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}} \right) + 1$	[−600, 600]	0
Penalized 1	$f_9(\boldsymbol{x}) = \frac{\pi}{D}\left\{ 10\sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 \left[1 + 10\sin^2(\pi y_{i+1}) \right] + (y_D - 1)^2 \right\} + \sum_{i=1}^{D} u(x_i, 10, 100, 4)$	[−50, 50]	0
Penalized 2	$f_{10}(\boldsymbol{x}) = \frac{1}{10}\left\{ \sin^2(3\pi x_1) + \sum_{i=1}^{D} (x_i - 1)^2 \left[1 + \sin^2(3\pi x_i + 1) \right] + (x_D - 1)^2 \left[1 + \sin^2(2\pi x_D) \right] \right\} + \sum_{i=1}^{D} u(x_i, 5, 100, 4)$	[−50, 50]	0

4.2.2. Experimental strategies

In the experiments, a population size of 40 and a maximum number of iterations of 2000 were used for all algorithms. The parameters for the five PSO variants were kept consistent with their respective literature. The benchmark functions were tested with dimensionalities of 10, 30 and 50 and each benchmark function was independently run 50 times in different dimensions to ensure the fairness of the experimental results. The error between the best optimum found by the algorithm and the actual global optimum was recorded and the mean error and standard deviation were displayed. The specific parameter settings for all algorithms can be found in Table 5.

Furthermore, the nonparametric Wilcoxon rank-sum tests were executed for each benchmark function to determine whether the results obtained by DCS-PSO are significantly different from the best results achieved by the other algorithms

Table 5. Parameters settings of the six algorithms.

Algorithm	Parameters
PSO-w	$w: 0.4,\ c_1 = c_2 = 2.0,\ V_{\max} = 0.2 \times Range$
GPSO	$w: 0.9 \sim 0.4,\ c_1 = c_2 = 2.0,\ V_{\max} = 0.2 \times Range$
LPSO	$\chi = 0.7298,\ c_1 = c_2 = 1.49445,\ V_{\max} = 0.2 \times Range$
CLPSO	$w: 0.9 \sim 0.4,\ c = 1.49445,\ V_{\max} = 0.2 \times Range,\ m = 7$
DMS-PSO	$\chi = 0.7298,\ c_1 = c_2 = 1.49445,\ V_{\max} = 0.2 \times Range,\ M = 4,\ R = 10$
DCS-PSO	$w: 0.4, c_1 = c_2 = 2.0, V_{\max} = 0.2 \times Range,\ h = 3000, \gamma = 0.15, \xi = 1.5$

4.2.3. Results and discussion

Tables 6, 7 and 8 display the means (Mean) and standard deviations (S.D.) of six algorithms on 10 test functions with dimensions of 10, 30 and 50, respectively. The *k* value presented in the second column of each table is the outcome of the nonparametric Wilcoxon rank-sum tests on our proposed algorithm and the second-best algorithms (indicated with underlining in tables). A *k* value of 1 indicates a significant difference in the performances of the two algorithms with 95% certainty, while a *k* value of 0 suggests no significant difference. Moreover, Table 7 ranks the algorithms according to their mean solution accuracy. The best results for each benchmark function are denoted in bold and the second-best results are underlined.

1) Results for the 10-D problems: For f_1, the performance of DCS-PSO is second only to LPSO, which maintains population diversity based on different neighborhood topologies, so its performance varies greatly on problems with different topologies. For f_2 and f_3, our DCS-PSO very clearly outperforms all other algorithms, in solution accuracy. DCS-PSO achieves comparable result to other algorithms on f_4 with CLPSO and DMS-PSO exceeding it. f_5 is a noise function that is difficult to solve due to the perturbation factor. However, DCS-PSO still gets the best result on this problem.

For multimodal problems, comparing DCS-PSO with CLPSO and DMS-PSO, which are algorithms proposed to improve the performance of PSO on multimodal problems, can demonstrate the competitiveness of DCS-PSO. DCS-PSO and CLPSO perform equally well with zero error on

Rastrigin's function (f_6), which has many local optima. For f_7, DCS-PSO achieves the best result, but according to the statistical test result, its performance is comparable to that of DMS-PSO. For f_8, PSO-w, GPSO, DMS-PSO and DCS-PSO perform similarly, but CLPSO performs relatively better. For two penalized functions f_9 and f_{10}, although the performance of DCS-PSO on f_9 is not as good as that of GPSO, CLPSO and DMS-PSO, DCS-PSO achieves the lowest error on f_{10}.

Overall, DCS-PSO obtains the best results in solving three unimodal functions (f_2, f_3 and f_5) and three multimodal functions (f_6, f_7 and f_{10}). Additionally, Table 6 shows that the S.D. values of DCS-PSO on most test functions are much smaller than those of other algorithms, indicating that DCS-PSO is more stable during the convergence process.

Table 6. Test results of the six algorithms for 10-D problems.

Function	k	Criteria	PSO-w	GPSO	LPSO	CLPSO	DMS-PSO	DCS-PSO
f_1	1	Mean	1.84×10^{-144}	1.47×10^{-73}	**2.78×10^{-169}**	2.18×10^{-21}	6.30×10^{-62}	2.35×10^{-151}
		S.D.	9.13×10^{-144}	1.04×10^{-72}	**0.00**	5.44×10^{-21}	4.25×10^{-61}	7.75×10^{-151}
f_2	1	Mean	1.89×10^{-81}	6.24×10^{-38}	7.59×10^{-09}	4.30×10^{-14}	3.45×10^{-20}	**9.65×10^{-84}**
		S.D.	1.26×10^{-80}	2.99×10^{-37}	3.80×10^{-08}	1.39×10^{-13}	5.86×10^{-20}	**4.81×10^{-83}**
f_3	1	Mean	1.22×10^{-44}	9.93×10^{-17}	6.11×10^{-13}	2.18×10^{-20}	6.13×10^{-10}	**1.21×10^{-48}**
		S.D.	7.42×10^{-44}	3.36×10^{-16}	4.32×10^{-12}	3.25×10^{-20}	3.33×10^{-09}	**2.38×10^{-48}**
f_4	1	Mean	2.79	2.80	4.00	2.49	**1.60**	9.11
		S.D.	1.39	1.16	2.56	**4.95×10^{-01}**	9.79×10^{-01}	1.89
f_5	1	Mean	7.23×10^{-04}	1.02×10^{-03}	3.25×10^{-03}	1.49×10^{-03}	8.80×10^{-04}	**2.15×10^{-04}**
		S.D.	3.90×10^{-04}	5.46×10^{-04}	2.43×10^{-03}	7.09×10^{-04}	4.49×10^{-04}	**1.58×10^{-04}**
f_6	0	Mean	5.99	2.13	9.35	**0.00**	2.67	**0.00**
		S.D.	2.74	1.15	3.53	**0.00**	1.36	**0.00**
f_7	0	Mean	4.14×10^{-15}	4.07×10^{-15}	4.67×10^{-15}	3.16×10^{-14}	4.00×10^{-15}	**3.91×10^{-15}**
		S.D.	7.03×10^{-16}	5.02×10^{-16}	1.62×10^{-15}	5.29×10^{-14}	**0.00**	5.62×10^{-16}
f_8	1	Mean	7.44×10^{-02}	6.88×10^{-02}	1.06×10^{-01}	**1.64×10^{-03}**	7.12×10^{-02}	6.65×10^{-02}
		S.D.	3.41×10^{-02}	3.13×10^{-02}	5.81×10^{-02}	**7.61×10^{-04}**	3.55×10^{-02}	3.75×10^{-02}
f_9	1	Mean	1.24×10^{-02}	**4.71×10^{-32}**	6.22×10^{-03}	2.99×10^{-30}	1.38×10^{-28}	1.78×10^{-01}
		S.D.	6.16×10^{-02}	**1.11×10^{-47}**	4.40×10^{-02}	3.07×10^{-30}	$2.65 \times 10-28$	6.11×10^{-02}
f_{10}	1	Mean	3.56×10^{-04}	9.33×10^{-20}	9.80×10^{-04}	3.16×10^{-27}	$2.57 \times 10-13$	**1.35×10^{-32}**
		S.D.	2.52×10^{-03}	1.90×10^{-19}	6.86×10^{-03}	6.79×10^{-27}	4.33×10^{-13}	**8.29×10^{-48}**

2) Results for the 30-D and 50-D problems: The experiments conducted on 10-D problems are repeated for 30-D and 50-D problems and the results are presented in Tables 7 and 8. All algorithms exhibited similar characteristics as on the 10-D problems. However, high dimensionality leads to increased complexity, resulting in a decline in the performance of optimization algorithms on most problems. As shown in Tables 7 and 8, for f_1, f_2 and f_5 with 30 and 50 dimensions, DCS-PSO consistently outperforms all other algorithms, particularly on f_2. Notably, for the Rosenbrock function (f_4), DCS-PSO performs similarly to other algorithms in 10 dimensionalities. However, its performance does not deteriorate as the dimension increases, illustrating that high dimensionality does not significantly decrease the performance of DCS-PSO on this problem.

For f_6 with 30 and 50 dimensions, CLPSO always performs best among all algorithms. DCS-PSO is still best on f_7 with 30 dimensionalities, while GPSO performs better on this problem with 50 dimensions. CLPSO and DMS-PSO retain their superiority on f_8 and f_9 with 30 and 50 dimensionalities, respectively. However, the nonparametric Wilcoxon rank-sum test results indicate that DCS-PSO is not statistically significantly poorer than CLPSO and DMS-PSO. DCS-PSO remains the best-performing algorithm on f_{10}, which suggests that its performance is not affected by the increased dimensionality of this function.

Table 7. Test results of the six algorithms for 30-D problems.

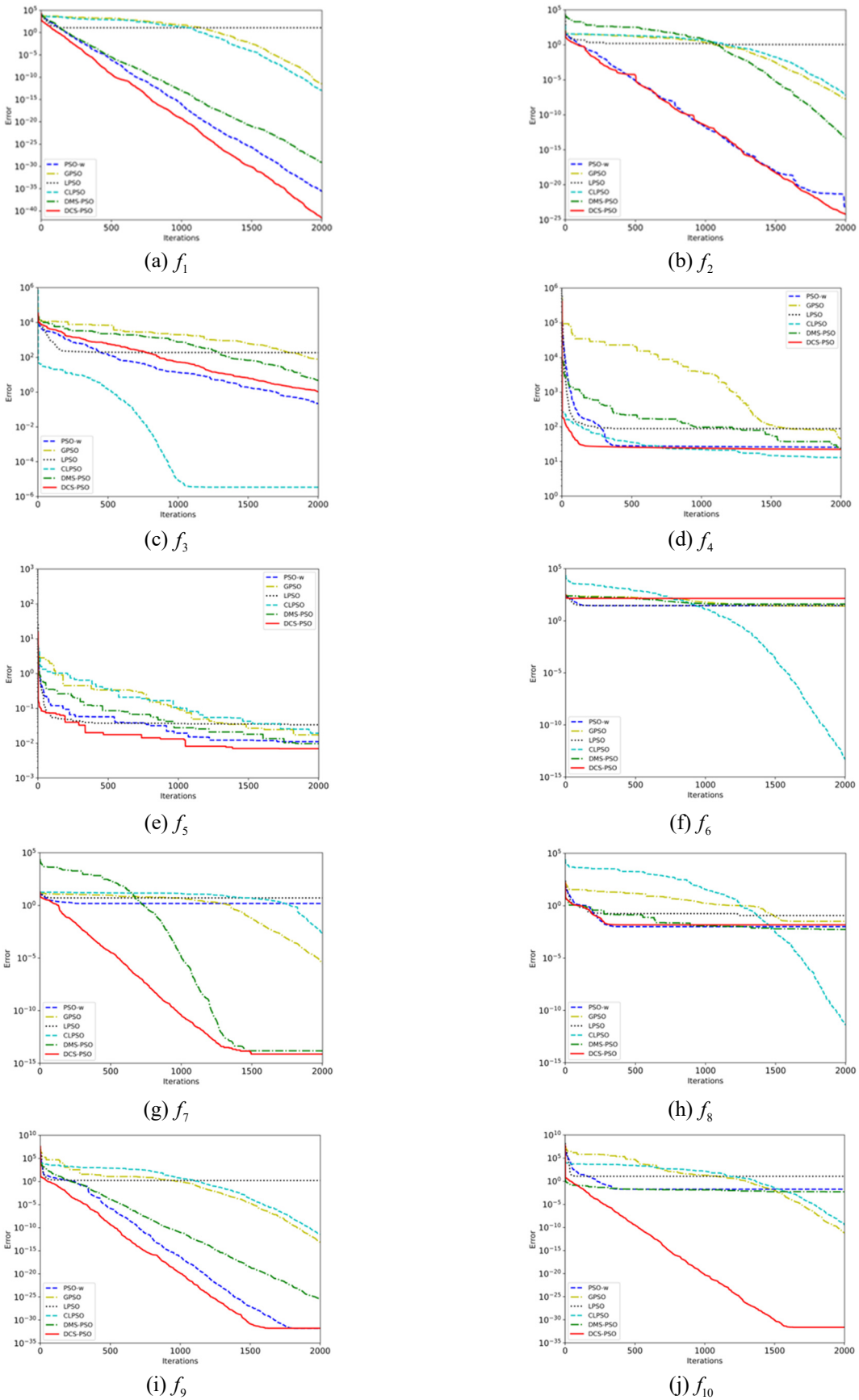
Function	k	Criteria	PSO-w	GPSO	LPSO	CLPSO	DMS-PSO	DCS-PSO
		Mean	1.28×10^{-39}	3.42×10^{-14}	3.82	3.83×10^{-13}	6.76×10^{-28}	**1.05×10^{-40}**
f_1	1	S.D.	2.48×10^{-39}	1.20×10^{-13}	$2.18\times 10^{+01}$	9.48×10^{-13}	5.49×10^{-28}	**2.01×10^{-40}**
		Rank	2	4	6	5	3	**1**
		Mean	1.67×10^{-09}	1.27×10^{-08}	1.83	4.45×10^{-08}	3.74×10^{-13}	**2.49×10^{-23}**
f_2	1	S.D.	1.15×10^{-08}	2.46×10^{-08}	1.83	2.84×10^{-08}	2.33×10^{-13}	**5.70×10^{-23}**
		Rank	3	4	6	5	2	**1**
		Mean	6.50×10^{-01}	$9.58 \times 10^{+01}$	$4.79\times 10^{+02}$	**4.37×10^{-04}**	$1.16 \times 10^{+01}$	5.90
f_3	1	S.D.	5.22×10^{-01}	$5.10 \times 10^{+01}$	$3.08\times 10^{+02}$	**3.04×10^{-04}**	5.95	4.45
		Rank	2	5	6	**1**	4	3
		Mean	$2.98\times 10^{+01}$	$4.17 \times 10^{+01}$	$9.42\times 10^{+01}$	**$1.81 \times 10^{+01}$**	$2.26 \times 10^{+01}$	$2.21 \times 10^{+01}$
f_4	1	S.D.	$2.16\times 10^{+01}$	$2.87 \times 10^{+01}$	$5.46\times 10^{+01}$	3.80	2.29	**6.32×10^{-01}**
		Rank	4	5	6	**1**	3	2
		Mean	8.74×10^{-03}	1.89×10^{-02}	8.00×10^{-02}	3.83×10^{-02}	9.99×10^{-03}	**5.12×10^{-03}**
f_5	1	S.D.	3.28×10^{-03}	6.58×10^{-03}	3.51×10^{-02}	1.47×10^{-02}	3.98×10^{-03}	**1.69×10^{-03}**
		Rank	2	4	6	5	3	**1**
		Mean	$3.88 \times 10^{+01}$	$3.34 \times 10^{+01}$	$5.12 \times 10^{+01}$	**4.40×10^{-11}**	$2.73 \times 10^{+01}$	$1.31 \times 10^{+02}$
f_6	1	S.D.	8.06	8.19	$1.62 \times 10^{+01}$	**6.58×10^{-11}**	$8.88 \times 10^{+00}$	$1.01 \times 10^{+01}$
		Rank	4	3	5	**1**	2	6
		Mean	3.81×10^{-01}	1.14×10^{-06}	4.32	3.15×10^{-03}	4.37×10^{-14}	**1.20×10^{-14}**
f_7	1	S.D.	6.47×10^{-01}	1.04×10^{-06}	1.16	2.18×10^{-02}	4.86×10^{-14}	**3.90×10^{-15}**
		Rank	5	3	6	4	2	**1**
		Mean	1.46×10^{-02}	1.38×10^{-02}	4.14×10^{-01}	**1.41×10^{-12}**	4.44×10^{-03}	1.15×10^{-02}
f_8	0	S.D.	1.70×10^{-02}	1.55×10^{-02}	3.12×10^{-01}	**2.52×10^{-12}**	4.99×10^{-03}	1.40×10^{-02}
		Rank	5	4	6	**1**	2	3
		Mean	1.39×10^{-01}	1.04×10^{-02}	3.87	2.32×10^{-14}	**1.08×10^{-26}**	2.96×10^{-02}
f_9	0	S.D.	2.62×10^{-01}	3.14×10^{-02}	2.66	5.72×10^{-14}	**2.24×10^{-26}**	4.74×10^{-02}
		Rank	4	3	6	2	1	4
		Mean	2.28×10^{-01}	1.10×10^{-02}	$2.16\times 10^{+01}$	1.03×10^{-11}	5.31×10^{-03}	**6.22×10^{-32}**
f_{10}	1	S.D.	6.63×10^{-01}	4.43×10^{-02}	$1.12\times 10^{+01}$	1.45×10^{-11}	4.13×10^{-04}	**7.45×10^{-32}**
		Rank	5	4	6	2	3	**1**
Ave. rank			3.6	3.9	5.9	2.7	2.5	2.3
Final rank			4	5	6	3	2	1

Table 8. Test results of the six algorithms for 50-D problems.

Function	k	Criteria	PSO-w	GPSO	LPSO	CLPSO	DMS-PSO	DCS-PSO
f_1	1	Mean	5.20×10^{-17}	3.68×10^{-04}	$1.30 \times 10^{+03}$	5.70×10^{-05}	1.26×10^{-06}	**3.93×10^{-18}**
		S.D.	1.91×10^{-16}	4.22×10^{-04}	$4.91 \times 10^{+02}$	6.50×10^{-05}	3.14×10^{-06}	**1.05×10^{-17}**
f_2	0	Mean	9.06×10^{-06}	2.01×10^{-01}	$2.26 \times 10^{+01}$	2.00×10^{-01}	1.48×10^{-04}	**5.87×10^{-07}**
		S.D.	5.83×10^{-05}	1.41	4.26	1.41	6.99×10^{-04}	**2.82×10^{-06}**
f_3	1	Mean	$4.52 \times 10^{+02}$	$4.38 \times 10^{+03}$	$7.71 \times 10^{+03}$	**1.55**	$1.90 \times 10^{+03}$	$1.33 \times 10^{+03}$
		S.D.	$1.68 \times 10^{+02}$	$1.88\times 10^{+03}$	$2.11 \times 10^{+03}$	**9.59×10^{-01}**	$2.30 \times 10^{+03}$	$5.06\times 10^{+02}$
f_4	1	Mean	$7.59 \times 10^{+01}$	$1.34\times 10^{+02}$	$3.19 \times 10^{+03}$	$1.69 \times 10^{+02}$	$1.46 \times 10^{+02}$	**4.52×10^{-02}**
		S.D.	$4.33 \times 10^{+01}$	$1.52\times 10^{+02}$	$1.65 \times 10^{+03}$	$2.75 \times 10^{+02}$	$1.41 \times 10^{+02}$	**2.98×10^{-02}**
f_5	1	Mean	3.75×10^{-02}	7.50×10^{-02}	9.04×10^{-01}	1.72×10^{-01}	4.52×10^{-02}	**1.85×10^{-02}**
		S.D.	1.12×10^{-02}	2.09×10^{-02}	3.67×10^{-01}	4.36×10^{-02}	1.67×10^{-02}	**6.22×10^{-03}**
f_6	1	Mean	$7.48 \times 10^{+01}$	$8.78\times 10^{+01}$	$1.16 \times 10^{+02}$	**1.32×10^{-03}**	$7.67 \times 10^{+01}$	$4.79 \times 10^{+01}$
		S.D.	$1.62 \times 10^{+01}$	$1.83\times 10^{+01}$	$2.04 \times 10^{+01}$	**6.87×10^{-03}**	$1.60 \times 10^{+01}$	3.18
f_7	0	Mean	1.38	**1.40×10^{-01}**	9.00	1.22	1.14	9.05×10^{-01}
		S.D.	8.89×10^{-01}	**3.52×10^{-01}**	1.08	5.83×10^{-01}	7.77×10^{-01}	8.47×10^{-01}
f_8	0	Mean	1.36×10^{-02}	8.18×10^{-03}	$1.22 \times 10^{+01}$	**3.18×10^{-04}**	3.79×10^{-02}	1.43×10^{-02}
		S.D.	2.61×10^{-02}	1.02×10^{-02}	4.71	**1.41×10^{-03}**	7.54×10^{-02}	1.76×10^{-02}
f_9	0	Mean	1.60×10^{-01}	1.04×10^{-01}	$2.08 \times 10^{+01}$	8.95×10^{-02}	**7.84×10^{-08}**	2.99×10^{-02}
		S.D.	2.65×10^{-01}	1.96×10^{-01}	8.57	1.44×10^{-01}	**4.22×10^{-07}**	4.91×10^{-02}
f_{10}	1	Mean	4.07	9.60×10^{-01}	$4.06 \times 10^{+02}$	1.35	8.78	**3.53×10^{-01}**
		S.D.	3.36	1.12	$3.10 \times 10^{+02}$	1.87	6.03	**3.69×10^{-01}**

3) Comparison of convergence speed: Figure 6 shows the convergence characteristics in terms of the error between the best fitness value found and the actual best function value of the median run of each algorithm for each test function with 30 dimensions. Since the convergence characteristics for the 10-D and 50-D cases are similar to those in the 30-D case, they are not presented.

Combining the numerical results and convergence plots, it can be concluded that all algorithms perform well in solving f_1 and f_2 with three different dimensionalities. However, DCS-PSO is faster than other algorithms. PSO exhibits similar convergence characteristics as DCS-PSO on f_2. The convergence characteristics of each algorithm on f_1 and f_2 are shown in Figure 6(a) and (b). Figure 6(c) indicates that CLPSO has better convergence speed and accuracy than other algorithms on f_3. Although DCS-PSO has slightly lower convergence accuracy than CLPSO on f_4, it has the fastest convergence speed as illustrated in Figure 6(d). All algorithms perform similarly on the noise problem (f_5). Figure 6(e) displays that these algorithms show a slow decrease in error, but DCS-PSO has the fastest convergence speed and the lowest error.



(a) f_1 (b) f_2

(c) f_3 (d) f_4

(e) f_5 (f) f_6

(g) f_7 (h) f_8

(i) f_9 (j) f_{10}

Figure 6. Convergence curves of the six algorithms in 30-D benchmark functions.

The advantages of CLPSO and DMS-PSO are revealed through some multimodal problems. As shown in Figure 6(f), although the convergence speed is slow, CLPSO has the best search accuracy on f_6. In contrast, other algorithms cannot continue to search effectively after reaching a certain accuracy. From the convergence characteristics in Figure 6(g), the performances of DCS-PSO and DMS-PSO are also excellent on f_7, but DCS-PSO has the fastest convergence speed. Although DCS-PSO is not as good as CLPSO and DMS-PSO in terms of average errors on f_8 and f_9, there is no statistically significant difference according to the Wilcoxon tests, as verified in Figure 6(h, i). It can be seen from Figure 6(j) that DCS-PSO has the best search efficiency on f_{10}. Considering the numerical results and the convergence speeds, DCS-PSO performs best among all tested algorithms.

4) Discussion: The nonparametric Wilcoxon rank-sum tests indicate that DCS-PSO is significantly better than or equivalent to the other algorithms on most problems with three different dimensionality cases. When the problems are 10-dimensional, DCS-PSO is significantly better than other algorithms on f_2, f_3, f_5 and f_{10}, and is equivalent to CLPSO and DMS-PSO on f_6 and f_7, respectively. For f_8 and f_9, with 30 and 50 dimensionalities, DCS-PSO performs equivalently to CLPSO and DMS-PSO, respectively. All statistical results demonstrate that DCS-PSO performs very well overall.

In the above experiments, DCS-PSO shows better convergence speed and accuracy on most unimodal problems, indicating that it is more effective in solving such problems. Simultaneously, the performance of DCS-PSO on most multimodal problems is comparable to that of CLPSO and DMS-PSO, demonstrating its ideal global search performance and ability to escape local optima.

4.3. Experiments on CEC2017

This section further uses a widely recognized CEC2017 test suite to confirm the performance of DCS-PSO with PSO variants. In CEC2017, there are 29 functions (F2 has been excluded because it shows unstable behavior, especially for higher dimensions) which can be classified into four categories: unimodal functions (f_1, f_3), multimodal functions ($f_4 - f_{10}$), hybrid functions ($f_{11} - f_{20}$) and composition functions ($f_{21} - f_{30}$). Due to space constraints, we only select D = 30 for analysis and discussion. The best value for each function is shown in bold.

4.3.1. Results comparison

Table 9 shows that CLPSO and DMS-PSO obtain the best result in unimodal functions, followed by DCS-PSO. In multimodal functions, DCS-PSO achieves the best performance on f_4, f_5, f_8 and f_9, 4 of 7 multimodal functions in terms of mean value, followed by GPSO, CLPSO, and DMS-PSO, which all obtain the best solutions on one function.

For the 10 hybrid functions, DCS-PSO reaches the best solutions on f_{13}, f_{14}, f_{15} and f_{18}, 4 of 10 hybrid functions. DMS-PSO also performs remarkably since it reaches the best solutions on 3 out of 10 functions, followed by GPSO, LPSO and CLPSO, which all obtain the best solutions on one function. In the ten composition functions, DMS-PSO attains remarkable results on f_{21}, f_{22}, f_{23}, f_{24}, f_{26} and f_{27}, 6 out of 10 composition functions. The comparison results indicate that DCS-PSO works well in complex and multimodal situations.

Table 9. Results of comparison algorithms on CEC2017 test suite (D = 30).

Function	Criteria	PSO-w	GPSO	LPSO	CLPSO	DMS-PSO	DCS-PSO
F1	Mean	$5.73 \times 10^{+08}$	$9.15 \times 10^{+09}$	$5.24 \times 10^{+09}$	**$1.01 \times 10^{+02}$**	$3.26 \times 10^{+03}$	$6.44 \times 10^{+03}$
	S.D.	$2.36 \times 10^{+09}$	$1.29 \times 10^{+10}$	$9.65 \times 10^{+09}$	**5.12×10^{-01}**	$4.53 \times 10^{+03}$	$1.93 \times 10^{+03}$
F3	Mean	$2.52 \times 10^{+03}$	$5.01 \times 10^{+03}$	$9.58 \times 10^{+03}$	$4.90 \times 10^{+02}$	**$2.16 \times 10^{+02}$**	$5.45 \times 10^{+02}$
	S.D.	$1.23 \times 10^{+03}$	$2.38 \times 10^{+03}$	$5.11 \times 10^{+03}$	$1.39 \times 10^{+01}$	$3.43 \times 10^{+02}$	**3.90×10^{-13}**
F4	Mean	$1.03 \times 10^{+02}$	$1.73 \times 10^{+02}$	$2.15 \times 10^{+02}$	$5.53 \times 10^{+02}$	$7.39 \times 10^{+01}$	**1.22**
	S.D.	$3.19 \times 10^{+01}$	$1.02 \times 10^{+02}$	$1.92 \times 10^{+02}$	7.54	$2.39 \times 10^{+01}$	**1.75**
F5	Mean	$1.43 \times 10^{+02}$	$7.61 \times 10^{+01}$	$1.42 \times 10^{+02}$	$6.00 \times 10^{+02}$	$5.88 \times 10^{+01}$	**$2.09 \times 10^{+01}$**
	S.D.	$3.11 \times 10^{+01}$	$2.08 \times 10^{+01}$	$3.78 \times 10^{+01}$	**0.00**	$1.30 \times 10^{+01}$	7.01
F6	Mean	$4.01 \times 10^{+01}$	$1.43 \times 10^{+01}$	$5.14 \times 10^{+01}$	$7.87 \times 10^{+02}$	**9.86×10^{-04}**	3.75
	S.D.	$1.49 \times 10^{+01}$	5.47	$1.39 \times 10^{+01}$	6.43	**2.47×10^{-03}**	3.17
F7	Mean	$1.29 \times 10^{+02}$	**$1.12 \times 10^{+01}$**	$1.91 \times 10^{+02}$	$8.49 \times 10^{+02}$	$9.76 \times 10^{+01}$	$1.28 \times 10^{+02}$
	S.D.	$3.46 \times 10^{+01}$	$1.03 \times 10^{+01}$	$4.86 \times 10^{+01}$	$1.03 \times 10^{+01}$	$1.99 \times 10^{+01}$	**9.23**
F8	Mean	$1.20 \times 10^{+02}$	$6.83 \times 10^{+01}$	$1.24 \times 10^{+02}$	$9.09 \times 10^{+02}$	$5.94 \times 10^{+01}$	**$1.62 \times 10^{+01}$**
	S.D.	$3.18 \times 10^{+01}$	$1.66 \times 10^{+01}$	$2.95 \times 10^{+01}$	**3.05**	$1.68 \times 10^{+01}$	5.52
F9	Mean	$1.38 \times 10^{+03}$	$1.53 \times 10^{+02}$	$2.26 \times 10^{+03}$	$3.27 \times 10^{+03}$	$2.33 \times 10^{+01}$	**6.09×10^{-01}**
	S.D.	$1.18 \times 10^{+03}$	$1.64 \times 10^{+02}$	$9.36 \times 10^{+02}$	$2.89 \times 10^{+02}$	$5.63 \times 10^{+01}$	**1.83**
F10	Mean	$3.61 \times 10^{+03}$	$3.19 \times 10^{+03}$	$3.91 \times 10^{+03}$	**$1.18 \times 10^{+03}$**	$2.97 \times 10^{+03}$	$1.99 \times 10^{+03}$
	S.D.	$5.67 \times 10^{+02}$	$6.28 \times 10^{+02}$	$7.81 \times 10^{+02}$	**4.51**	$5.89 \times 10^{+02}$	$2.87 \times 10^{+02}$
F11	Mean	$1.30 \times 10^{+02}$	$1.71 \times 10^{+02}$	$1.76 \times 10^{+02}$	$5.97 \times 10^{+05}$	**$4.74 \times 10^{+01}$**	$1.12 \times 10^{+02}$
	S.D.	$4.18 \times 10^{+01}$	$5.41 \times 10^{+01}$	$5.11 \times 10^{+01}$	$3.96 \times 10^{+05}$	$3.12 \times 10^{+01}$	**$2.44 \times 10^{+01}$**
F12	Mean	$4.56 \times 10^{+07}$	$6.40 \times 10^{+07}$	$1.76 \times 10^{+08}$	**$2.62 \times 10^{+03}$**	$8.25 \times 10^{+04}$	$2.79 \times 10^{+03}$
	S.D.	$2.83 \times 10^{+08}$	$1.12 \times 10^{+08}$	$5.00 \times 10^{+08}$	**$1.55 \times 10^{+03}$**	$2.16 \times 10^{+04}$	$2.07 \times 10^{+03}$
F13	Mean	$1.77 \times 10^{+06}$	$5.01 \times 10^{+07}$	$1.88 \times 10^{+07}$	$1.75 \times 10^{+04}$	$1.55 \times 10^{+04}$	**$1.27 \times 10^{+04}$**
	S.D.	$8.67 \times 10^{+06}$	$1.71 \times 10^{+08}$	$1.02 \times 10^{+08}$	$9.65 \times 10^{+03}$	$1.65 \times 10^{+04}$	**$3.86 \times 10^{+03}$**
F14	Mean	$1.80 \times 10^{+04}$	$3.70 \times 10^{+04}$	$1.10 \times 10^{+04}$	$1.61 \times 10^{+03}$	$1.32 \times 10^{+04}$	**$1.02 \times 10^{+02}$**
	S.D.	$2.36 \times 10^{+04}$	$4.40 \times 10^{+04}$	$1.30 \times 10^{+04}$	$2.70 \times 10^{+01}$	$1.23 \times 10^{+04}$	**$2.45 \times 10^{+01}$**
F15	Mean	$3.99 \times 10^{+03}$	$1.97 \times 10^{+04}$	$1.07 \times 10^{+04}$	$2.16 \times 10^{+03}$	$6.90 \times 10^{+03}$	**$1.09 \times 10^{+02}$**
	S.D.	$3.96 \times 10^{+03}$	$3.05 \times 10^{+04}$	$1.25 \times 10^{+04}$	$1.32 \times 10^{+02}$	$9.16 \times 10^{+03}$	**$1.14 \times 10^{+02}$**
F16	Mean	$1.04 \times 10^{+03}$	**$7.61 \times 10^{+02}$**	$1.13 \times 10^{+03}$	$1.84 \times 10^{+03}$	$7.95 \times 10^{+02}$	$1.79 \times 10^{+03}$
	S.D.	$2.73 \times 10^{+02}$	$2.47 \times 10^{+02}$	$2.95 \times 10^{+02}$	**$4.18 \times 10^{+01}$**	$2.10 \times 10^{+02}$	$1.25 \times 10^{+02}$
F17	Mean	$4.00 \times 10^{+04}$	$1.57 \times 10^{+04}$	$1.21 \times 10^{+04}$	$2.16 \times 10^{+05}$	**$1.89 \times 10^{+02}$**	$4.62 \times 10^{+03}$
	S.D.	$6.54 \times 10^{+02}$	$1.54 \times 10^{+03}$	$2.56 \times 10^{+03}$	$1.61 \times 10^{+05}$	**$1.14 \times 10^{+02}$**	$6.54 \times 10^{+02}$
F18	Mean	$1.41 \times 10^{+05}$	$4.45 \times 10^{+05}$	$1.12 \times 10^{+05}$	$1.95 \times 10^{+03}$	$1.27 \times 10^{+05}$	**$5.56 \times 10^{+02}$**
	S.D.	$1.04 \times 10^{+05}$	$3.62 \times 10^{+05}$	$1.15 \times 10^{+05}$	**$1.88 \times 10^{+01}$**	$1.06 \times 10^{+05}$	$8.59 \times 10^{+02}$
F19	Mean	$6.20 \times 10^{+03}$	$4.47 \times 10^{+05}$	**$6.10 \times 10^{+03}$**	$6.30 \times 10^{+03}$	$9.39 \times 10^{+03}$	$7.78 \times 10^{+03}$
	S.D.	$6.09 \times 10^{+03}$	$2.71 \times 10^{+06}$	$6.37 \times 10^{+03}$	**$2.64 \times 10^{+01}$**	$1.28 \times 10^{+04}$	$1.08 \times 10^{+03}$
F20	Mean	$1.70 \times 10^{+04}$	$1.45 \times 10^{+04}$	$1.92 \times 10^{+04}$	$2.36 \times 10^{+03}$	**$2.73 \times 10^{+02}$**	$6.00 \times 10^{+03}$
	S.D.	$3.26 \times 10^{+03}$	$3.44 \times 10^{+02}$	$2.75 \times 10^{+03}$	**5.04**	$1.41 \times 10^{+02}$	$5.32 \times 10^{+02}$
F21	Mean	$3.39 \times 10^{+02}$	$2.80 \times 10^{+02}$	$3.34 \times 10^{+02}$	$2.65 \times 10^{+03}$	$2.62 \times 10^{+02}$	**$1.03 \times 10^{+02}$**
	S.D.	$5.45 \times 10^{+01}$	$4.25 \times 10^{+01}$	$4.90 \times 10^{+01}$	$5.40 \times 10^{+02}$	$2.11 \times 10^{+01}$	**8.18×10^{-01}**

Continued on next page

Function	Criteria	PSO-w	GPSO	LPSO	CLPSO	DMS-PSO	DCS-PSO
F22	Mean	$4.15 \times 10^{+02}$	$6.72 \times 10^{+02}$	$1.00 \times 10^{+03}$	$2.72 \times 10^{+03}$	$2.38 \times 10^{+02}$	$\mathbf{1.09 \times 10^{+02}}$
	S.D.	$9.94 \times 10^{+02}$	$7.48 \times 10^{+02}$	$1.50 \times 10^{+03}$	$1.15 \times 10^{+01}$	$5.92 \times 10^{+02}$	**1.44**
F23	Mean	$6.42 \times 10^{+02}$	$5.82 \times 10^{+02}$	$6.73 \times 10^{+02}$	$2.92 \times 10^{+03}$	$4.30 \times 10^{+02}$	$\mathbf{3.25 \times 10^{+02}}$
	S.D.	$1.01 \times 10^{+02}$	$9.36 \times 10^{+01}$	$1.10 \times 10^{+02}$	$1.21 \times 10^{+01}$	$2.10 \times 10^{+01}$	**7.91**
F24	Mean	$6.61 \times 10^{+02}$	$6.34 \times 10^{+02}$	$7.14 \times 10^{+02}$	$2.89 \times 10^{+03}$	$4.94 \times 10^{+02}$	$\mathbf{3.53 \times 10^{+02}}$
	S.D.	$7.19 \times 10^{+01}$	$8.33 \times 10^{+01}$	$8.07 \times 10^{+01}$	$\mathbf{2.23 \times 10^{-01}}$	$2.60 \times 10^{+01}$	$1.09 \times 10^{+01}$
F25	Mean	$\mathbf{3.95 \times 10^{+02}}$	$4.13 \times 10^{+02}$	$4.47 \times 10^{+02}$	$3.47 \times 10^{+03}$	$3.97 \times 10^{+02}$	$4.26 \times 10^{+02}$
	S.D.	$1.50 \times 10^{+01}$	$3.64 \times 10^{+01}$	$3.05 \times 10^{+01}$	$7.89 \times 10^{+02}$	**1.53**	$2.34 \times 10^{+01}$
F26	Mean	$1.24 \times 10^{+03}$	$1.51 \times 10^{+03}$	$2.45 \times 10^{+03}$	$3.21 \times 10^{+03}$	$1.49 \times 10^{+03}$	$\mathbf{4.02 \times 10^{+02}}$
	S.D.	$1.29 \times 10^{+03}$	$7.71 \times 10^{+02}$	$1.36 \times 10^{+03}$	**6.70**	$5.18 \times 10^{+02}$	$5.94 \times 10^{+01}$
F27	Mean	$5.88 \times 10^{+02}$	$5.80 \times 10^{+02}$	$6.21 \times 10^{+02}$	$3.21 \times 10^{+03}$	$5.38 \times 10^{+02}$	$\mathbf{4.03 \times 10^{+02}}$
	S.D.	$5.87 \times 10^{+01}$	$5.54 \times 10^{+01}$	$5.36 \times 10^{+01}$	5.65	$1.25 \times 10^{+01}$	**4.45**
F28	Mean	$4.34 \times 10^{+02}$	$5.05 \times 10^{+02}$	$5.49 \times 10^{+02}$	$3.45 \times 10^{+03}$	$\mathbf{3.40 \times 10^{+02}}$	$3.88 \times 10^{+02}$
	S.D.	$4.76 \times 10^{+01}$	$7.23 \times 10^{+01}$	$1.46 \times 10^{+02}$	**4.85**	$5.79 \times 10^{+01}$	8.36
F29	Mean	$6.25 \times 10^{+02}$	$8.71 \times 10^{+02}$	$6.24 \times 10^{+02}$	$8.91 \times 10^{+02}$	$\mathbf{6.03 \times 10^{+02}}$	$6.38 \times 10^{+02}$
	S.D.	$\mathbf{4.87 \times 10^{+01}}$	$6.31 \times 10^{+01}$	$2.17 \times 10^{+02}$	$7.46 \times 10^{+01}$	$1.55 \times 10^{+02}$	$7.88 \times 10^{+01}$
F30	Mean	$4.52 \times 10^{+04}$	$4.50 \times 10^{+05}$	$3.78 \times 10^{+05}$	$\mathbf{1.12 \times 10^{+03}}$	$5.81 \times 10^{+03}$	$3.14 \times 10^{+05}$
	S.D.	$1.10 \times 10^{+05}$	$1.41 \times 10^{+06}$	$4.93 \times 10^{+05}$	$\mathbf{2.22 \times 10^{+03}}$	$2.34 \times 10^{+03}$	$5.14 \times 10^{+05}$

4.3.2. Friedman test results

Friedman test is used in this section to compare the performance of DCS-PSO and other PSO variants. The results of the Friedman test are shown in Table 10. It can be seen from Table 10 that DCS-PSO achieves a remarkable result on the CEC2017, followed by DMS-PSO, PSO-w, GPSO, CLPSO and LPSO. The overall performance in CEC2017 reveals that DCS-PSO has a distinct advantage over other compared algorithms.

Table 10. Friedman test of all compared algorithms on the CEC2017 test suite.

Average rank	Algorithm	Ranking
1	DCS-PSO	2.03
2	DMS-PSO	2.17
3	PSO-w	3.69
4	GPSO	4.07
5	CLPSO	4.45
6	LPSO	4.59

5. Conclusions and future works

In this paper, we propose an improved particle swarm optimization combined with double-chaos search mechanism, namely DCS-PSO. The standard PSO is simple and efficient, but it is difficult to maintain population diversity when dealing with complex problems with a large search space, leading to premature convergence and getting trapped in local optima. The chaos search mechanism has the advantages of global traversal and avoiding falling into local optima.

In the first stage of DCS-PSO, we adopt the double-chaos search mechanism to narrow the search space to the vicinity of the optimal solution, effectively reducing the risk of PSO getting trapped in local optima. In the second stage, to enhance the population diversity, the logistic map is utilized to perform a global search in the narrowed search space and the best solution obtained from both chaos search and population search will guide the population to converge. Experimental studies show that the algorithm can effectively narrow the search space and has better convergence accuracy and speed for most functions.

Of course, according to the no free lunch theorem, DCS-PSO cannot optimally solve every kind of global optimization problem. Although we use the chaos search mechanism to improve the convergence speed and ability to escape the local optima of DCS-PSO, it also brings about a problem worth further investigation: how to ensure that the optimal solution must exist in the narrowed search space. In the future, we will continue conducting in-depth research on chaos theory, optimize our DCS-PSO algorithm and try to apply DCS-PSO to solving these real-world problems.

Use of AI tools declaration

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

Acknowledgments

This research was supported by the National Natural Science Foundation of China (Grant No 82260849); the National Natural Science Foundation of China (Grant No 61562045); and Jiangxi University of Chinese Medicine Science and Technology Innovation Team Development Program (Grant No CXTD22015).

References

1. J. Kennedy, R. Eberhart, Particle swarm optimization, in *Proceedings of ICNN'95-international conference on neural networks*, **4** (1995), 1942–1948. https://doi.org/10.1109/ICNN.1995.488968
2. J. H. Holland, Genetic algorithms, *Sci. Am.*, **21** (1992), 66–73. https://doi.org/10.1038/scientificamerican0792-66
3. G. G. Wang, S. Deb, Z. Cui, Monarch butterfly optimization, *Neural Comput. Appl.*, **31** (2019), 1995–2014. https://doi.org/10.1007/s00521-015-1923-y
4. S. Li, H. Chen, M. Wang, A. A. Heidari, S. Mirjalili, Slime mould algorithm: A new method for stochastic optimization, *Future Gener. Comput. Syst.*, **111** (2020), 300–323. https://doi.org/10.1016/j.future.2020.03.055
5. G. G. Wang, Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems, *Memetic Comput.*, **10** (2018), 151–164. https://doi.org/10.1007/s12293-016-0212-3
6. Y. Yang, H. Chen, A. A. Heidari, A. H. Gandomi, Hunger games search: Visions, conception, implementation, deep analysis, perspectives, and towards performance shifts, *Expert Syst. Appl.*, **177** (2021), 114864. https://doi.org/10.1016/j.eswa.2021.114864

7. I. Ahmadianfar, A. A. Heidari, A. H. Gandomi, X. Chu, H. Chen, RUN beyond the metaphor: An efficient optimization algorithm based on Runge Kutta method, *Expert Syst. Appl.*, **181** (2021), 115079. https://doi.org/10.1016/j.eswa.2021.115079
8. J. Tu, H. Chen, M. Wang, A. H. Gandomi, The colony predation algorithm, *J. Bionic. Eng.*, **18** (2021), 674–710. https://doi.org/10.1007/s42235-021-0050-y
9. I. Ahmadianfar, A. A. Heidari, S. Noshadian, H. Chen, A. H. Gandomi, INFO: An efficient optimization algorithm based on weighted mean of vectors, *Expert Syst. Appl.*, **195** (2022), 116516. https://doi.org/10.1016/j.eswa.2022.116516
10. A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, H. Chen, Harris hawks optimization: Algorithm and applications, *Future Gener. Comput. Syst.*, **97** (2019), 849–872. https://doi.org/10.1016/j.future.2019.02.028
11. H. Su, D. Zhao, A. A. Heidari, L. Liu, X. Zhang, M. Mafarja, et al., RIME: A physics-based optimization, *Neurocomputing*, **532** (2023), 183–214. https://doi.org/10.1016/j.neucom.2023.02.010
12. Y. Li, G. Wang, H. Chen, L. Shi, L. Qin, An ant colony optimization based dimension reduction method for high-dimensional datasets, *J. Bionic. Eng.*, **10** (2013), 231–241. https://doi.org/10.1016/S1672-6529(13)60219-X
13. S. Thawkar, S. Sharma, M. Khanna, L. K. Singh, Breast cancer prediction using a hybrid method based on Butterfly Optimization Algorithm and Ant Lion Optimizer, *Comput. Biol. Med.*, **139** (2021), 104968. https://doi.org/10.1016/j.compbiomed.2021.104968
14. S. Chakraborty, A. K. Saha, S. Nama, S. Debnath, COVID-19 X-ray image segmentation by modified whale optimization algorithm with population reduction, *Comput. Biol. Med.*, **139** (2021), 104984. https://doi.org/10.1016/j.compbiomed.2021.104984
15. X. Gao, L. Wang, X. Yu, X. Su, Y. Ding, C. Lu, et al., Conditional probability based multi-objective cooperative task assignment for heterogeneous UAVs, *Eng. Appl. Artif. Intell.*, **123** (2023), 106404. https://doi.org/10.1016/j.engappai.2023.106404
16. X. Yu, X. Gao, L. Wang, X. Wang, Y. Ding, C. Lu, et al., Cooperative multi-uav task assignment in cross-regional joint operations considering ammunition inventory, *Drones*, **6** (2022), 77. https://doi.org/10.3390/drones6030077
17. C. Li, Y. Zhang, X. Su, X. Wang, An improved optimization algorithm for aeronautical maintenance and repair task scheduling problem, *Mathematics*, **10** (2022), 3777. https://doi.org/10.3390/math10203777
18. F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, X. L. Shen, A hybrid particle swarm optimization algorithm using adaptive learning strategy, *Inf. Sci.*, **436** (2018), 162–177. https://doi.org/10.1016/j.ins.2018.01.027
19. A. Ratnaweera, S. K. Halgamuge, H. C. Watson, Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients, *IEEE Trans. Evol. Comput.*, **8** (2004), 240–255. https://doi.org/10.1109/TEVC.2004.826071
20. B. Y. Qu, P. N. Suganthan, S. Das, A distance-based locally informed particle swarm model for multimodal optimization, *IEEE Trans. Evol. Comput.*, **17** (2013), 387–402. https://doi.org/10.1109/TEVC.2012.2203138
21. R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, *IEEE Trans. Evol. Comput.*, **8** (2004), 204–210. https://doi.org/10.1109/TEVC.2004.826074

22. Y. Shi, R. Eberhart, A modified particle swarm optimizer, in *1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360)*, (1998), 69–73. https://doi.org/10.1109/ICEC.1998.699146
23. Y. Shi, R. C. Eberhart, Empirical study of particle swarm optimization, in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, **3** (1999), 1945–1950. https://doi.org/10.1109/CEC.1999.785511
24. H. Liu, X. W. Zhang, L. P. Tu, A modified particle swarm optimization using adaptive strategy, *Expert Syst. Appl.*, **152** (2020), 113353. https://doi.org/10.1016/j.eswa.2020.113353
25. K. Chen, F. Zhou, L. Yin, S. Wang, Y. Wang, F. Wan, A hybrid particle swarm optimizer with sine cosine acceleration coefficients, *Inf. Sci.*, **422** (2018), 218–241. https://doi.org/10.1016/j.ins.2017.09.015
26. J. Kennedy, R. Mendes, Population structure and particle swarm performance, in *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, **2** (2002), 1671–1676. https://doi.org/10.1109/CEC.2002.1004493
27. J. J. Liang, P. N. Suganthan, Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism, in *2006 IEEE International Conference on Evolutionary Computation*, (2006), 9–16. https://doi.org/10.1109/CEC.2006.1688284
28. J. J. Liang, A. K. Qin, P. N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, *IEEE Trans. Evol. Comput.*, **103** (2006), 281–295. https://doi.org/10.1109/TEVC.2005.857610
29. Y. Wang, B. Wang, Z. Li, C. Xu, A novel particle swarm optimization based on hybrid-learning model, *Math. Biosci. Eng.*, **20** (2023), 7056–7087. https://doi.org/10.3934/mbe.2023305
30. X. Zhou, S. Zhou, Y. Han, S. Zhu, Lévy flight-based inverse adaptive comprehensive learning particle swarm optimization, *Math. Biosci. Eng.*, **19** (2022), 5241–5268. https://doi.org/10.3934/mbe.2022246
31. K. T. Alligood, T. D. Sauer, J. A. Yorke, D. Chillingworth, Chaos: an introduction to dynamical systems, *Phys. Today*, **50** (1997), 67–68. https://doi.org/10.1063/1.882006
32. B. Li, W. S. Jiang, Chaotic optimization method and its application, *Control Theory Appl.*, **14** (1997), 613–615.
33. M. Ji, H. Tang, Application of chaos in simulated annealing, *Chaos Solitons Fractals*, **21** (2004), 933–941. https://doi.org/10.1016/j.chaos.2003.12.032
34. K. Aihara, T. Takabe, M. Toyoda, Chaotic neural networks, *Phys. Lett. A*, **144** (1990), 333–340. https://doi.org/10.1016/0375-9601(90)90136-C
35. B. L. W. Jiang, Optimizing complex functions by chaos search, *Cybern. Syst.*, **29** (1998), 409–419. https://doi.org/10.1080/019697298125678
36. R. M. May, Simple mathematical models with very complicated dynamics, *Nature*, **261** (1976), 459–467. https://doi.org/10.1038/261459a0
37. C. B. Xiu, X. D. Liu, Y. H. Zhang, Optimization algorithm using two kinds of chaos and its application, *Control Decis.*, **6** (2003), 724–726.
38. H. Y. Liang, X. S. Gu, A novel chaos optimization algorithm based on parallel computing, *J. East China Univ Sci. Technol.*, **4** (2004), 450–453.
39. B. Liu, L. Wang, Y. H. Jin, F. Tang, D. X. Huang, Improved particle swarm optimization combined with chaos, *Chaos Solitons Fractals*, **25** (2005), 1261–1271. https://doi.org/10.1016/j.chaos.2004.11.095

40. P. J. Angeline, Evolutionary optimization versus particle swarm optimization: Philosophy and performance differences, in *International conference on evolutionary programming*, (1998), 601–610. https://doi.org/10.1007/BFb0040811
41. Y. Yu, S. Gao, S. Cheng, Y. Wang, S. Song, F. Yuan, CBSO: a memetic brain storm optimization with chaotic local search, *Memetic Comput.*, **10** (2018), 353–367. https://doi.org/10.1007/s12293-017-0247-0
42. L. Wang, *Intelligent Optimization Algorithms with Applications*, Tsinghua University Press, Beijing, 2001.
43. Z. Tu, Y. Lu, A robust stochastic genetic algorithm (StGA) for global numerical optimization, *IEEE Trans. Evol. Comput.*, **8** (2004), 456–470. https://doi.org/10.1109/TEVC.2004.831258
44. C. Y. Lee, X. Yao, Evolutionary programming using mutations based on the Levy probability distribution, *IEEE Trans. Evol. Comput.*, **8** (2004), 1–13. https://doi.org/10.1109/TEVC.2003.816583

AIMS Press