IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 21, 2022, accepted March 7, 2022, date of publication March 10, 2022, date of current version March 21, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3158666

CAPSO: Chaos Adaptive Particle Swarm Optimization Algorithm

YOUXIANG DUAN[1], NING CHEN[1], LUNJIE CHANG[2], YONGJING NI[3], S. V. N. SANTHOSH KUMAR[4], AND PEIYING ZHANG[1]

[1]College of Computer Science and Technology, China University of Petroleum (East China), Qingdao 266580, China
[2]Research Institute of Petroleum Exploration and Development, PetroChina Tarim Oilfield Company, Korla 841000, China
[3]School of Information Science and Engineering, Hebei University of Science and Technology, Shijiazhuang, Hebei 050000, China
[4]School of Information Technology and Engineering, Vellore Institute of Technology, Vellore 632014, India

Corresponding author: Yongjing Ni (23781315@qq.com)

This work was supported in part by the Shandong Provincial Natural Science Foundation, China, under Grant ZR2020MF006; in part by the Industry-University Research Innovation Foundation of Ministry of Education of China under Grant 2021FNA01001; in part by the Major Scientific and Technological Projects of China National Petroleum Corporation (CNPC) under Grant ZD2019-183-006; in part by the Fundamental Research Funds for the Central Universities of China University of Petroleum (East China) under Grant 20CX05017A; and in part by the Youth Fund for Science and Technology Research in Colleges and Universities of Hebei Province under Grant QN2021066.

ABSTRACT As an influential technology of swarm evolutionary computing (SEC), the particle swarm optimization (PSO) algorithm has attracted extensive attention from all walks of life. However, how to rationally and effectively utilize the population resources to equilibrate the exploration and utilization is still a key dispute to be resolved. In this paper, we propose a novel PSO algorithm called Chaos Adaptive Particle Swarm Optimization (CAPSO), which adaptively adjust the inertia weight parameter w and acceleration coefficients c_1, c_2, and introduces a controlling factor γ based on chaos theory to adaptively adjust the range of chaotic search. This makes the algorithm have favorable adaptability, and then the particles cannot only effectively prevent missing the global optimal solution, but also have a high probability of jumping out of the local optimal solution. To verify the stability, convergence speed, and accuracy of CAPSO, we conduct ample experiments on 6 test functions. In addition, to further verify the effectiveness and scalability of CAPSO, comparative experiments are carried out on the CEC2013 test suite. Finally, the results prove that CAPSO outperforms other peer algorithms to achieve satisfactory performance.

INDEX TERMS Swarm evolutionary computing, particle swarm optimization, chaos theory, function optimization.

I. INTRODUCTION

Most engineering optimization problems can be abstracted into the mathematical representation of multimodal functions with multiple minimum (maximum) values [1]. How to solve such problems has critically influenced academic dialogue [2]–[4]. The Particle Swarm Optimization (PSO) algorithm [5], which is researched and developed by Kennedy and Eberhart [6], is an important technology with uniqueness and effectiveness in optimization problems. Once it was published, it triggered a wave of research. For example, Iranmehr *et al.* [7] develop a new method based on PSO to extract audio features similar to human ears. Seo *et al.* [8] propose a multi-group particle swarm optimization (MGPSO) algorithm and further apply it to electromagnetic optimization problems. Lei [9] introduces the concave function form of the inertia weight into the PSO, and applies it to the model parameter optimization of the established Francis hydraulic turbine governing system. Marjani *et al.* [10] combine genetic algorithms and PSO to supervise neural networks. They track the connections between the applied operators and the layers through genetic algorithms and use PSO to check the values of all individual deviations and weights in the neural network to modify the best network topology. Bouzidi *et al.* [11] develop a new operator specifically used to solve combinatorial optimization problems, and embed it into the improved discrete particle swarm optimization algorithm (DPSO-CO), opening up a new horizon to solve the traveling salesman problem. Due to its excellent performance, PSO has been widely used in many fields [3], [4], [12] such as medicine, chemical industry, agriculture, finance, etc., and has successfully

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Ching Ying.

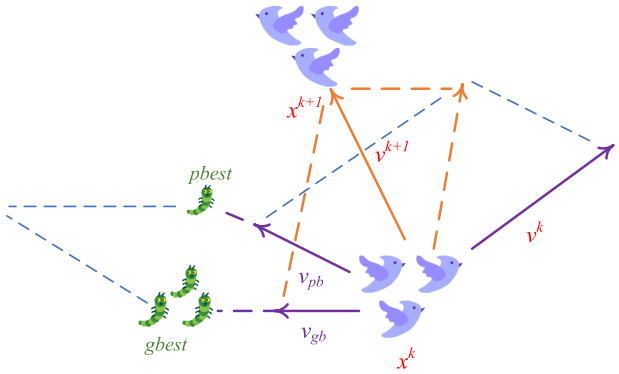


FIGURE 1. The influence of local best position *pbest* and global best position *gbest* on population flight. x^k and x^{k+1} represent the position of the particle before and after the particle is updated, respectively. v_k and v_{k+1} represent the running velocity of the particle before and after the particle is updated, respectively. v_{pb} represents the velocity when the particle moves towards *pbest*. v_{gb} represents the velocity when the particle moves towards *gbest*.

achieved satisfactory application results in resource scheduling, control system design, load problems, and power system optimization.

As the key algorithm for swarm intelligence optimization, the principle of PSO is to simulate the social behavior of biological communities [13], and use the evolution and iteration of the population to achieve the purpose of optimization. In the process of aggregation and predation, the position corresponds to the solution of the problem, and the velocity determines the direction and distance of the next search [14]. The movement direction of the particle is adjusted according to the position closest to the food found by itself and the entire group [5]. Therefore, the location of the food that the population is looking for can be abstracted as the best solution in the solution of the problem. Moreover, PSO judges whether the position of the particle is the best according to the value of the fitness function, which plays a guiding role for the flight of the population [15]. As shown in FIGURE 1, in each iteration, the direction and distance of the particle search will be adjusted in time under the joint influence of its local optimal position and the global optimal position of all particles. Iterate many times until the conditions are met.

However, in the optimization process of traditional PSO, problems such as low convergence accuracy and difficulty in finding the global optimum are prone to occur. Meanwhile, Chaos Optimization Algorithm (COA) [16] can provide search diversity in the optimization process, and has been successfully used in robot optimization control, parameter optimization in control systems, financial systems, and manufacturer scheduling [17], etc. In COA, chaotic mapping, as a simple and effective mapping method, can improve the exploration of meta-heuristic algorithms. Inspired by COA making full use of many characteristics of chaotic variables in a certain range of search space to improve the probability of searching for global optimal solutions, we propose the Chaos Adaptive Particle Swarm Optimization algorithm (CAPSO). The main *contributions* of CAPSO can be summarized as follows:

1) We combine linear and nonlinear inertia weight to adaptively adjust the local and global search ability of particles. A nonlinear method is adopted to adjust the acceleration coefficient adaptively so that the particles can quickly obtain the global optimal solution, avoid falling into the local optimal, and speed up the convergence speed.
2) The *Logistic* mapping is used to initialize the population, which increases the richness and convenience of the population and makes it easier for particles to jump out of the local optimum.
3) To replace the current global optimal point with a better point, the control factor γ is introduced to adaptively adjust the search range of the particles near the global optimal solution.
4) Finally, extensive experiments are carried out to verify the effectiveness of the algorithm. The results show that CAPSO can dynamically adjust the value of each parameter, but also has high convergence accuracy and strong stability.

The rest of this paper is organized as follows. The related work is reviewed in Section II. Section III manifests the problem definition of this paper. The proposed CAPSO algorithm is presented in Section IV. In Section V, the results of our experiment are provided. Finally, in Section VI, we summarize our work done.

II. RELATED WORK

At present, the related work of the improved PSO algorithm can be roughly summarized into the following aspects.

A. INERTIA WEIGHT AND ACCELERATION COEFFICIENT

Zhang *et al.* [18] introduce a fuzzy system into PSO for the breast cancer problem, and propose a fuzzy adaptive PSO algorithm to adjust the searchability of the algorithm. The application of fuzzy adaptive PSO to train the feedforward neural network is more accurate and stable, can converge to the best position faster, and reduce the risk of overfitting to a certain extent. Lynn and Suganthan propose a novel PSO, called Heterogeneous comprehensive learning particle swarm optimization (HCLPSO) [19]. It divides the population into two sub-populations for exploration and utilization respectively, and proposes a comprehensive learning strategy combined with particle experience for the search process, which made the algorithm obtain satisfactory convergence. Inspired by the activation function widely used in neural networks, Liu *et al.* [20] combine the Sigmoid function for weighting to adaptively adjust the acceleration coefficient, and propose Adaptive Weighted Particle Swarm Optimization (AWPSO), which significantly improved the convergence of the population. To solve the problem of slightly insufficient performance of the standard particle swarm optimization algorithm (SPSO) for high-dimensional complex optimization problems, Lin *et al.* [21] introduce a new parameter adjustment strategy of piecewise nonlinear

acceleration coefficient (PNAC), and propose an improved SPSO algorithm (P-SPSO) based on PNAC. At the same time, they also develop a mean difference mutation strategy (MDM) for the update mechanism of P-SPSO, which is called mean-differential-mutation-strategy embedded P-SPSO (MP-SPSO). This algorithm has significant effects in terms of solution quality and robustness and has been successfully applied in practical applications.

B. VELOCITY AND POSITION FORMULA

Combining the characteristics of wireless sensor networks, Nagireddy *et al.* [3] propose a PSO algorithm based on velocity adaptation (VAPSO), which improves the traditional velocity update formula by introducing partial derivatives of local and global optimums to time. It improves the convergence and minimizes the positioning error, thereby helping to improve the positioning accuracy and network life.

C. INTRODUCE NEW RULES OR PARAMETERS

To balance the convergence and diversity of the population, Zhang *et al.* [22] propose an adaptive bare-bones particle swarm optimization (ABPSO) algorithm. It adds disturbance value to each particle through convergence and population diversity, and introduces a mutation operator to adaptively adjust the global search process. Moreover, to solve the problem of resource-constrained project scheduling (RCPSP), Kumar and Vidyarthi [23] embed the valid particle generator (VPG) operator into the PSO, and propose an adaptive particle swarm optimization algorithm (A-PSO). This algorithm can convert the invalid particles caused by the dependent behavior of RCPSP into effective particles, and adjust the inertia weight through three parameters of fitness value, previous inertia weight, and the iteration counter to speed up the convergence speed of the algorithm. Due to the large shrinkage factor of traditional PSO in the initial iteration process, its global distribution in the solution space cannot accurately track the local optimal solution, which leads to the problem of difficulty in convergence. Acharya and Kumar [24] propose a new shrinkage factor (ECF) and apply it to channel equalization. The simulation results prove that this algorithm achieves a perfect balance between local and global search, and has better performance. Yan *et al.* [25] introduce the constraint factor into the velocity update of the SPSO, and dynamically adjust the inertia weight according to the exponential decay mode. This makes it possible not only to obtain enjoyable global searchability in the early stage of the optimization process, but also to obtain a higher local search performance in the later stage. The results show that it has more advantages than other algorithms in terms of convergence speed and stability.

D. HYBRID APPROACH

Chuang *et al.* [26] propose an accelerated chaotic particle swarm optimization (ACPSO) algorithm by combining chaotic mapping and acceleration strategies. This algorithm searches for the appropriate cluster center through any data set, and can efficiently find an ideal alternative solution to the data clustering problem. Wang *et al.* [27] combine PSO and Chaos Search Technology (CST) to solve the problem of nonlinear bipolar programming. This method can greatly increase the search diversity of the population and avoid the algorithm capturing local particles. Gong *et al.* [28] propose a novel Genetic Learning PSO Algorithm (GLPSO) by combining genetic evolution techniques. Specifically, it trains particles through the genetic algorithm and uses the experience of particles to guide the evolution process, which makes GLPSO have a significant improvement in searchability and efficiency. On this basis, Xia *et al.* [29] combine the genetic algorithm and propose Triple Archive Particle Swarm Optimization (TAPSO) to improve search efficiency through the collaboration of three different roles particles. Wei *et al.* [30] propose Multiple Adaptive Particle Swarm Optimization (MAPSO), which divides multiple clusters in each iteration, adjusts the cluster distribution according to fitness, and breeds particles using differential evolution. GOLCHI [31] proposed a hybrid algorithm of firefly and improved particle swarm optimization (IPSO) to optimize load balancing in a cloud environment to achieve a better average load and to improve important indicators such as effective resource utilization and task response time. This algorithm not only has obvious advantages in convergence speed and response speed, but also has better flexibility than other methods in minimizing average load through different goals.

III. PROBLEM DEFINITION

A. NOTATION

The definitions of related notations used in this paper are shown in the Table 1.

TABLE 1. Related notations.

Notation	Description
k	Current iteration number
T	Maximum iteration number
w	Inertia weight
c_1, c_2	Acceleration coefficient
M	Population size (number of particles)
N	Search space dimensions
v	Particle search velocity
x	Particle location
l	Solution space minimum
u	Solution space maximum

B. PROBLEM DEFINITION

Due to PSO being prone to premature convergence and difficult to accurately search for the global optimum, Shi and Eberhart [32] try to introduce a new parameter-inertia weight parameter w based on the original parameters to make more fine adjustments to the algorithm. Such PSO algorithms with w parameters are collectively referred to as standard particle swarm optimization algorithms (SPSO).

Suppose a total of M particles are searched in an $N-$dimensional space, where the local optima of particles

and the global optima of all particles can be broadly expressed as $pbest_i = (pbest_{i,1}, pbest_{i,2}, \ldots, pbest_{i,N})$ and $gbest = (gbest_1, gbest_2, \ldots, gbest_N)$. During each iteration of SPSO, the velocity v_{ij} and position x_{ij} of the particle i in the j dimension of the search space are adjusted by Eq.1 and Eq.2.

$$v_{ij}^{k+1} = wv_{ij}^{k} + c_1 r_1 \left(pbest_{ij} - x_{ij}^{k}\right) + c_2 r_2 \left(gbest_j - x_{ij}^{k}\right), \tag{1}$$

$$x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1}, \tag{2}$$

where $i = 1, 2, \ldots, M$. $j = 1, 2, \ldots, N$. r_1, r_2 are random numbers belonging to [0, 1].

C. INFLUENCE OF PARAMETERS

The SPSO algorithm is simple, efficient, and easy to understand and implement. However, in the optimization process, the selection and adjustment of various parameters are closely related to the performance of the algorithm, as follows:

1) INERTIAL WEIGHT w

In SPSO, the inertia weight can be used to adjust the particle's ability to explore the solution space, and its value determines the ability to adjust the current velocity during the velocity update process. When the value of w is large ($w > 1.2$), the particles tend to search globally, and constantly try to search in new areas. There is a high probability that the global optimal solution will be missed, and more iterations are needed to find it. When the value of w is average ($0.8 < w < 1.2$), the particles' global search ability is the best. When the value of w is small ($0.4 < w < 0.8$), the particles tend to search for local areas. At this time, if the particles search near the global optimal solution, the possibility of finding it is higher. Shi and Eberhart [32] suggest that the inertia weight w is dynamically adjusted during the optimization process, starting from a larger value of 0.9 (more inclined to global search), and dynamically reduced to 0.4 (more inclined to local search).

2) ACCELERATION COEFFICIENT c_1, c_2

The acceleration coefficient is also called the learning factor, which can adjust the particle and population cognitive ability. When its value is large, it can make the particles search quickly outside the target area, and the search range is wide, but it is easy to miss the global optimal solution. When its value is small, it can make the particles search within the target area. The particle search range is small, but it is difficult to jump out of the local optimum. For example, when $c_1 = c_2 = 0$, the particles can only move along the initial direction, the search range of the particles is small, and to a large extent, the global optimal solution cannot be found. When $c_1 = 0, c_2 \neq 0$, the particles can only search based on population experience, and it is also difficult to find the global optimal solution. When $c_1 \neq 0, c_2 = 0$, the population experience cannot be relied on. At this time, the particle itself cannot be effectively searched, and the search range is also small, making it difficult to find the global optimal solution. Therefore, to avoid affecting the information exchange and optimization capabilities between particles, it is necessary to set appropriate learning factors. Furthermore, the setting strategy is generally divided into static and dynamic strategies. Static strategy refers to setting the learning factor to a constant, usually set to 2 [32], but some scholars believe that it can be set to 1.494 [33], 2.05, 2.5 [34] etc. The dynamic strategy means that the value of the learning factor changes dynamically with the optimization process. For example, the values of c_1 and c_2 are continuously increased [35], c_1 keeps decreasing while c_2 keeps increasing [36], etc.

3) POPULATION SIZE M

For SPSO, the larger the value of M means the more particles need to be searched. At the same time, the searchability of the algorithm is stronger, and the easier it is to search for the global optimal solution, the corresponding search for the solution will take the longer. The smaller the value of M, the fewer particles need to be searched, the more difficult it is to search for the global optimal solution, but the corresponding search for the solution will be shorter. For different optimization problems, it is generally set according to experience and the difficulty of the problem to be optimized.

4) MAXIMUM VELOCITY v_{max}

The purpose of maximum velocity is to control the change of particle velocity. The larger its value, the greater the amplitude of particle movement and the higher the possibility of missing the global optimal solution. The lower the value, the smaller the amplitude of particle movement, it may take a long time to find the required solution and it is difficult to jump out of the local optimum. Related studies have shown that the effect of setting the maximum velocity and adjusting the inertia weight w is the same, so it is generally not adjusted further.

Given the influence of different parameters, CAPSO optimizes the adjustment and change process of the parameters on the basis of the SPSO, which will be introduced in detail in Section IV.

IV. THE PROPOSED APPROACH CAPSO

A. ADAPTIVE INERTIA WEIGHT w

The inertia weight w determines the extent to which the current velocity is affected by the previous velocity, and its value seriously affects the accuracy and convergence speed of the SPSO. In the early stage of the iteration, using a larger w can increase the particle's movement velocity and global search capability. In the later stage of the iteration, using a smaller w can reduce the moving velocity of the particles and make them focus on the local search to improve the accuracy of the optimal solution. Generally, as the iteration progresses, w decreases linearly. Whereas the optimization process of SPSO is very complicated, simple linear adjustment can no longer meet the needs of the algorithm. We combine

linear and nonlinear w to adaptively adjust the particle's local and global searchability. The specific adjustment method is shown in Eq.3.

$$w = \begin{cases} w_{\max}, & 0 \le \dfrac{k}{T} \le 0.1, \\ w_{\min} + \dfrac{(w_{\max} + w_{\min})}{e^{-w_{\max}} + e^{-1.2+\frac{20k}{T}}}, & \text{else}, \end{cases} \tag{3}$$

where w_{max} and w_{min} are the predefined maximum and minimum values of inertia weight, respectively.

B. ADAPTIVE ACCELERATION COEFFICIENT c_1, c_2

The acceleration coefficients c_1 and c_2 are designed to enable the algorithm to quickly cover the entire search space in the early stage, and improve the accuracy and convergence speed of the algorithm in the later stage. In most existing work, it is usually set to a constant, but on the whole, adaptive adjustment can increase the ability of particles to find the optimal solution and make the algorithm have better performance. We combine the non-linear strategy to make the value of c_1 gradually decrease from 2.5 while adjusting the value of c_2 to gradually increase from 0.5. In this way, the particles are more inclined to find the optimal solution based on their own experience in the early iteration process, increasing the diversity of the search while quickly obtaining the global optimal solution, avoiding falling into the local optimal. In the later iterative process, the particles are more inclined to find the optimal solution based on the experience of the population and have a strong local search ability, which can adjust the accuracy of the global optimal solution in more detail and speed up the convergence speed. The specific adjustment strategies for acceleration coefficients c_1 and c_2 are shown in Eq.4 and Eq.5.

$$c_1 = 0.5 * (c_{\max} - c_{\min}) \left(\frac{k}{T}\right)^2 + c_{\min}, \tag{4}$$

$$c_2 = (c_{\max} - c_{\min}) \left(\frac{k}{T}\right)^2 + c_{\max}, \tag{5}$$

where c_{max} and c_{min} are the maximum and minimum values of the predefined acceleration coefficient, respectively.

C. INITIAL POPULATION

In the SPSO, initializing the population is the first and critical step. The stronger the ergodicity of the initial population, the richer the diversity of the population, and the easier it is to overcome the obstacles of the local optimum to find the global optimum so that the performance of the algorithm is superior. In general, the most commonly used is to initialize particles randomly, but to some extent, it is difficult to ensure the ergodicity of the population, which affects the final result.

In our work, *Logistic* mapping is used to generate a series of chaotic variables to initialize the population, which is shown in Eq.6.

$$z_{k+1} = a z_k (1 - z_k), \tag{6}$$

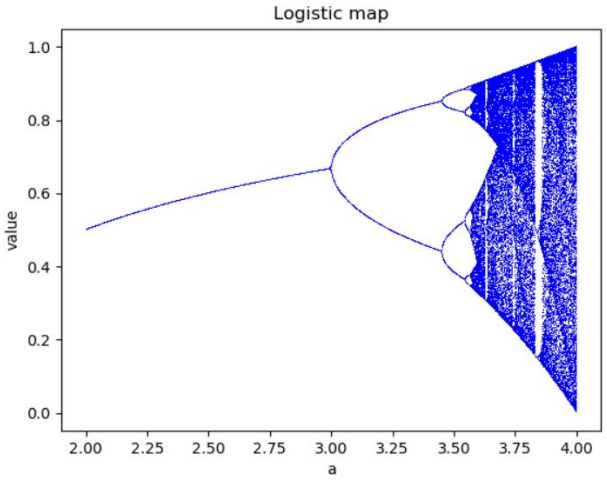


FIGURE 2. ***Logistic*** **mapping.**

where z_{k+1} is the value after mapping, z_k is the value before mapping, and a is a random variable. As illustrated in the FIGURE 2, when a changes from 3.569945672 to 4, the chaotic state of the system gradually changes from the initial state to the complete state. When its value exceeds 4, the system will become unstable. Therefore, the value of a is generally set to 4. To better illustrate the superiority of *Logistic* chaotic mapping over randomly initialized populations, we conduct 1000 iterations of experiments, and the results are shown in FIGURE 3. It can be seen that the ergodicity of generating points using *Logistic* chaotic mapping is better than random function.

D. CHAOS OPTIMIZATION

When some particles are searching near the global optimal solution, if they move at the previous velocity, they may miss the position of the solution. When the particle moves to the vicinity of the local optimal value, as it continues to iterate, the remaining particles will move in the direction of this particle, thereby causing the particles to fall into the local optimal value. To solve this problem, we introduce the control factor γ shown in Eq.7, which not only can effectively prevent the particles from missing the global optimal solution, but also makes them have a great probability of jumping out of the local optimal situation.

$$\gamma = \xi + \frac{1}{e^{0.1+\frac{5k}{T}}}, \tag{7}$$

where $\xi \in [0, 1]$ is the adjustment variable that can be adjusted according to the actual situation.

Use γ to control chaotic search, and the process of optimizing *gbest* is shown in the Algorithm 1.

In the process of iterative search, the value of the control factor γ decreases nonlinearly, so that the search range near the global optimal solution is gradually reduced, and the current optimal point is replaced with a better point. In the early iterative search process, the value of γ is large, so that the population roughly searches for the larger area around the current global optimal solution. In the middle and later

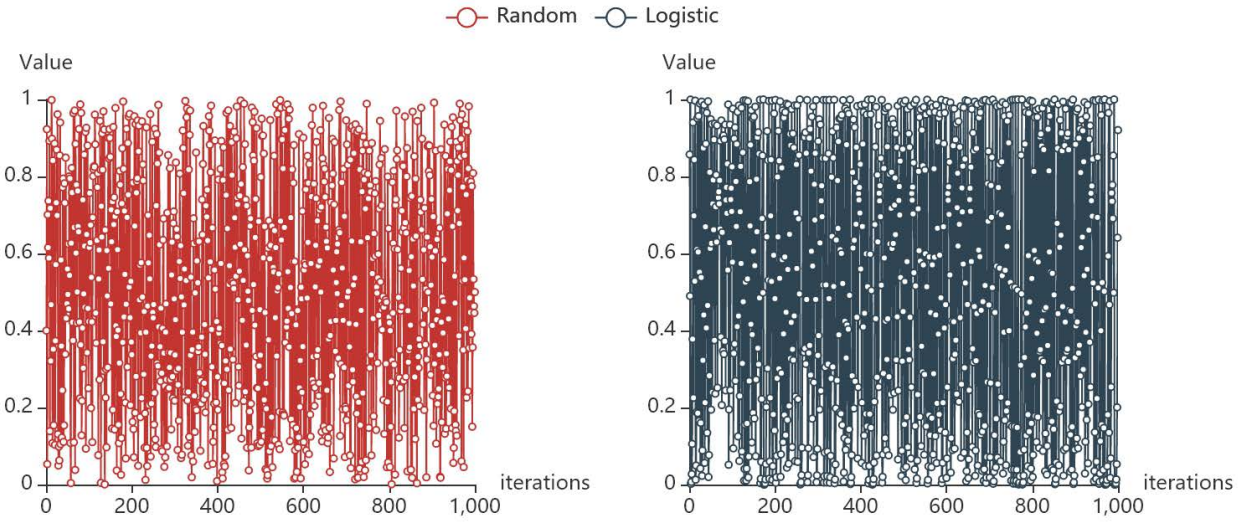


FIGURE 3. Comparison chart of *Logistic* chaotic mapping and random function.

Algorithm 1 Chaos Optimization *gbest*

1. Scale the value of each dimension of $gbest = (gbest_1, gbest_2, \ldots, gbest_N)$ to the range of $[0, 1]$ according to $gbest^{0t1} = \frac{gbest-u}{u-l}$, where $l = (l_1, l_2, \ldots, l_N)$ is the minimum value of the solution space, $u = (u_1, u_2, \ldots, u_N)$ is the maximum value of the solution space;
2. Use the Eq.6 to map $gbest^{0t1}$ to generate chaotic variables $gbest^c$;
3. Use linear mapping $gbest^* = l + gbest^c \times (u - l)$ to map the chaotic variable $gbest_c$ back to the original value range, and get $gbest^*$;
4. According to $x_i = \gamma \times gbest_i^*, i = 1, 2, \ldots, N$ control the search of the neighborhood range and calculate the corresponding fitness function value $f(x_i)$;
5. If there is $f(x_i) < gbest$, then let $gbest = f(x_i)$.

iterative search process, the value of γ is small, so that a more refined search can be performed to find the global optimal point. In this case, the search can quickly converge, thereby reducing running time.

E. CAPSO ALGORITHM FLOW

Function optimization problems include minimum and maximum optimization. Without loss of generality, in this paper, we carry out the corresponding research with the minimum optimization of multimodal functions. Based on our work, the maximum problem is also easy to derive.

Suppose the constrained optimization problem is expressed as: $minf_i(x)$, $X = (x_1, x_2, \ldots, x_i, \ldots, x_M)$ is the particle population, and the position of each particle is $x_i = (x_{i1}, x_{i2}, \ldots, x_{ij}, \ldots, x_{iN})$ where $x_{ij} \in [l_j, u_j]$. The algorithm steps of CAPSO are shown in the Algorithm 2.

It is assumed that the i-th iteration of the population of particles takes C_i time, the maximum number of iterations is T, M is the number of particles in the population, M_i is the number of particles in the i-th iteration of the population, and N is the dimension of the search space. Then the computational complexity of the proposed CAPSO is:

$$complexity = \sum_{i=1}^{T} N \times M_i \times C_i. \tag{8}$$

V. EXPERIMENT

A. TEST FUNCTION

This work selects 6 commonly used test functions to test the performance of CAPSO, which are:

1) DROP-WAVE FUNCTION

The *DROP-WAVE* function is shown in the Eq.9, and its image is illustrated in FIGURE 4(a). It can be seen that this function is multi-model and complex, and the value of the global optimal solution is -1.

$$f_1(x_1, x_2) = -\frac{1 + \cos\left(12\sqrt{x_1^2 + x_2^2}\right)}{0.5\left(x_1^2 + x_2^2\right) + 2}, \tag{9}$$

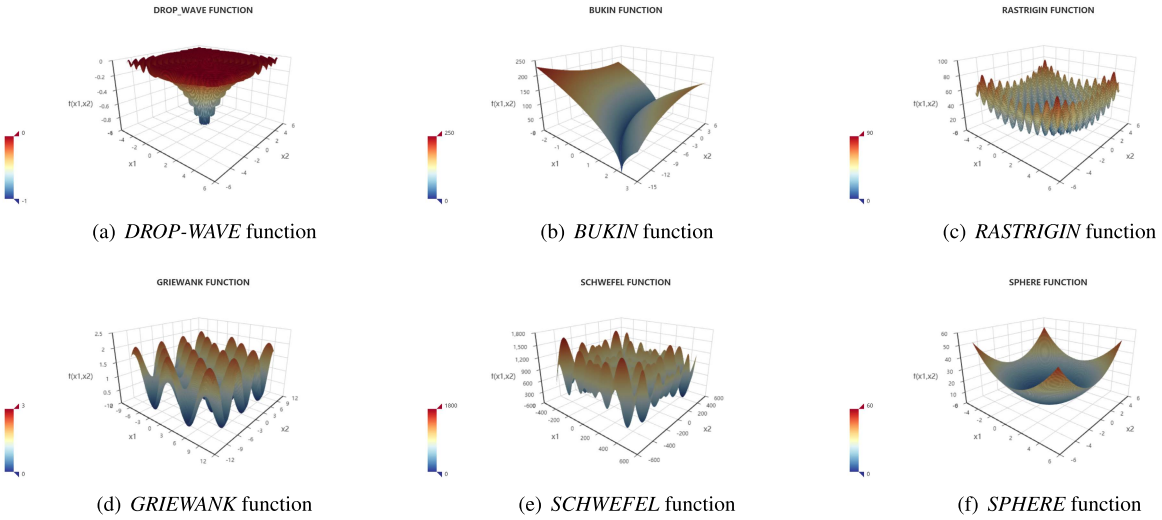
where the domain of definition is $x_i \in [-5.12, 5.12], i = 1, 2$.

2) BUKIN FUNCTION

The *BUKIN* function is shown in Eq.10, and its image is illustrated in FIGURE 4(b). It can be seen that this function has many local minima, the value of its global optimal solution is 0, and these minima are all located in a ridge.

$$f_2(x_1, x_2) = 100\sqrt{\left|x_2 - 0.01\, x_1^2\right|} + 0.01\,|x_1 + 10|, \tag{10}$$

where the domain of definition is $x_1 \in [-15, -5]$ and $x_2 \in [-3, 3]$.



(a) *DROP-WAVE* function (b) *BUKIN* function (c) *RASTRIGIN* function

(d) *GRIEWANK* function (e) *SCHWEFEL* function (f) *SPHERE* function

FIGURE 4. Test function.

Algorithm 2 The Algorithm Steps of CAPSO

Initialization parameters: $w_{max}, w_{min}, c_{max}, c_{min}, \xi, T, l, u, k = 0$.

Initialize the particle population X**:**

1. Randomly initialize each dimension x_{ij} of each particle x_i, where $0 \le x_{ij} \le 1$, and $x_{ij} \notin (0, 0.25, 0.5, 0.75, 1)$;

2. Use Eq.6 to map the population X to generate the chaotic variable X^c;

3. Use linear mapping $X^k = l + X^c \times (u - l)$ to map X^c to the original value, and then obtain the initial population $X^k = (x_1^k, x_2^k, \ldots, x_i^k, \ldots, x_M^k)$.

repeat

1. Calculate the fitness value $f_i(x_i^k)$ of each particle x_i for the fitness function $f_i(x)$.

2. if $f_i(x_i^k) < pbest_i$ **then**

$pbest_i = f_i(x_i^k)$

end

3. if $f_i(x_i^k) < gbest_i$ **then**

$gbest_i = f_i(x_i^k)$

end

4. Update the inertia weight w and acceleration coefficient c_1, c_2 according to Eq.3, Eq.4 and Eq.5;

5. Update the velocity v_{ij}^k and the position x_{ij}^k of each particle in the population X^k according to Eq.1 and Eq.2;

6. According to the algorithm 1, the control factor γ is used to control the chaotic search and optimize *gbest*;

7. k++.

until the cycle epoch iterates T times or the fitness function value $f_i(X^k)$ no longer changes.

Output: global optimal solution *gbest*.

3) RASTRIGIN FUNCTION

The *RASTRIGIN* function is shown in Eq.11, and its image is illustrated in FIGURE 4(c). It can be seen that this multi-modal function has a global optimal solution with a value of 0 and multiple local optimal solutions.

$$f_3(x_i) = 10N + \sum_{i=1}^{N} \left[x_i^2 - 10\cos(2\pi x_i) \right] \tag{11}$$

where the domain of definition is $x_i \in [-5.12, 5.12]$, $i = 1, 2, \ldots, N$.

TABLE 2. Comparison of parameter settings.

Algorithm	Parameter settings
SPSO	$w = 0.7298, c_1 = c_2 = 2$
LDWPSO	$w_{max} = 0.9, w_{min} = 0.4, c_1 = c_2 = 2$
ChPSO	$w_{max} = 0.9, w_{min} = 0.4$
CMPSO	$w_{max} = 0.9, w_{min} = 0.2, c_1 = c_2 = 2, H = 0.003$
CAPSO	$w_{max} = 0.9, w_{min} = 0.4, c_{max} = 2.5, c_{min} = 0.5, \xi = 0.2$

4) GRIEWANK FUNCTION

The *GRIEWANK* function is shown in Eq.12, and its image is illustrated in FIGURE 4(d). It can be seen that this function has a global minimum value of 0 and multiple local minimums.

$$f_4(x_i) = \sum_{i=1}^{N} \frac{x_i^2}{4000} - \prod_{i=1}^{d} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{12}$$

where the domain of definition is $x_i \in [-10, 10]$, $i = 1, 2, \ldots, N$.

5) SCHWEFEL FUNCTION

The *SCHWEFEL* function is shown in Eq.13, and its image is illustrated in FIGURE 4(e). It can be seen that this function has multiple local minima and the value of the global optimal solution is 0.

$$f_5(x_i) = 418.9829\, N - \sum_{i=1}^{N} x_i \sin\left(\sqrt{|x_i|}\right) \tag{13}$$

where the domain of definition is $x_i \in [-500, 500]$, $i = 1, 2, \ldots, N$.

TABLE 3. Test function results of each algorithm when $N = 2$.

Algorithm	Metric	f_1	f_2	f_3	f_4	f_5	f_6
SPSO	*Mean*	0.0074	0.0712	0.0934	0.0015	0.0085	1.085e-34
	Variance	3.846e-4	3.882e-3	0.0777	2.128e-5	8.02e-5	3.177e-67
	Standard deviation	0.0199	0.0634	0.2834	0.0047	0.0091	5.732e-34
LDWPSO	*Mean*	0.0149	0.0401	0.0332	2.229e-06	0.0037	1.112e-69
	Variance	7.271e-4	**2.666e-4**	0.0319	3.162e-11	1.854e-5	3.564e-137
	Standard deviation	0.0274	**0.0166**	0.1817	5.719e-6	0.0044	6.072e-69
ChPSO	*Mean*	0.0106	0.0332	0.0663	1.304e-10	0.0058	1.911e-112
	Variance	5.645e-4	4.066e-4	0.0616	1.334e-19	6.595e-5	1.030e-222
	Standard deviation	0.2524	0.0205	0.2524	3.715e-10	0.0083	1.032e-111
CMPSO	*Mean*	0.0242	0.2941	0.1974	0.2771	0.0110	3.181e-10
	Variance	9.231e-4	0.4981	0.2027	0.8434	0.0001	2.871e-18
	Standard deviation	0.0309	0.7178	0.4579	0.9341	0.0104	1.724e-9
CAPSO	*Mean*	**0.0018**	**0.0327**	**0.0143**	**1.036e-15**	**0.0024**	**7.275e-156**
	Variance	**9.200e-5**	3.573e-4	**0.0042**	**1.753e-30**	**9.163e-6**	**6.660e-310**
	Standard deviation	**0.0098**	0.0192	**0.0663**	**1.347e-15**	**0.0031**	**2.625e-155**

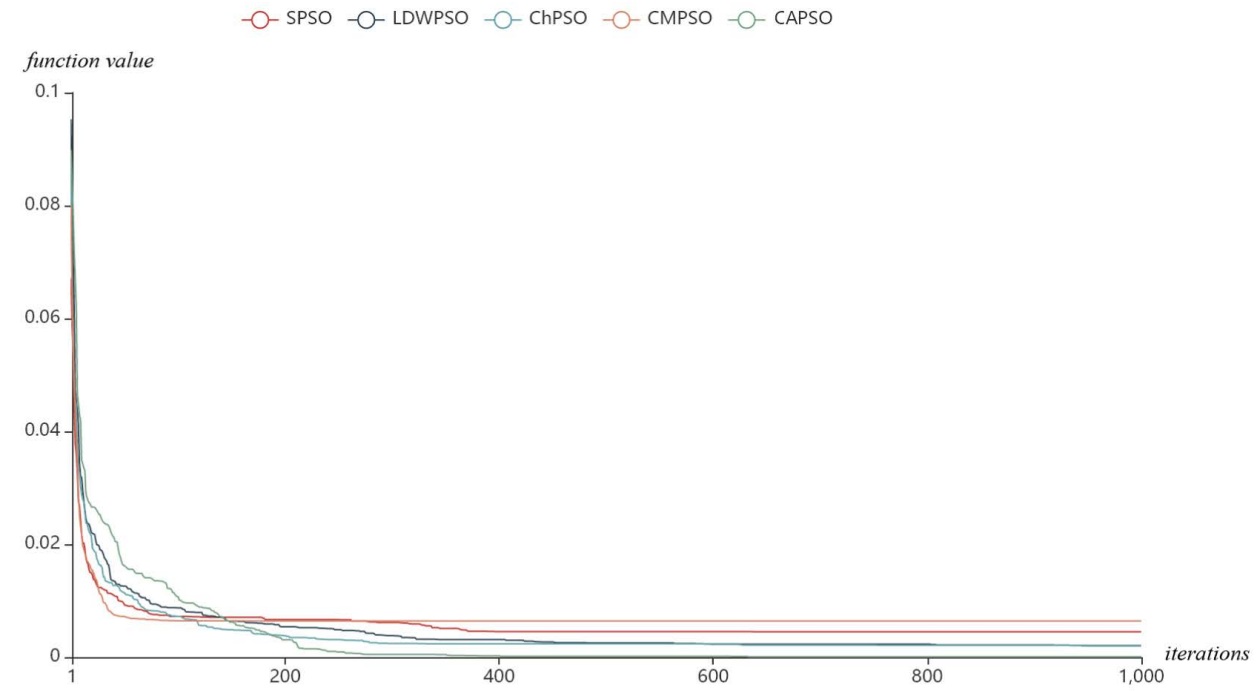


FIGURE 5. The change of average fitness value of each algorithm on *GRIEWANK* function.

6) SPHERE FUNCTION

The *SPHERE* function is shown in Eq.14, and its image is illustrated in FIGURE 4(f). It can be seen that this function is unimodal and convex, with N local minima, and the value of the global optimal solution is 0.

$$f_6(x_i) = \sum_{i=1}^{N} x_i^2 \tag{14}$$

where the domain of definition is $x_i \in [-5.12, 5.12]$, $i = 1, 2, \ldots, N$.

B. BASELINE

We compare CAPSO with Standard Particle Swarm Optimization algorithm (SPSO) [32], Linearly Decreasing inertia Weight Particle Swarm Optimization algorithm (LDWPSO) [37], ChPSO (a PSO algorithm improved by Cheng and Han) [38], and Chaos Modified Particle Swarm Optimization algorithm (CMPSO) [39] in terms of convergence speed and accuracy. The parameters of each algorithm are adjusted and set for each test function, as shown in Table 2. Taking into account the influence of randomness, for all algorithms, the dimensions N are selected as 2 and 10 in the test process to repeat the experiment 30 times.

C. PERFORMANCE

When $N = 2$, the results of the mean, variance, and standard deviation of the difference between the fitness value of the repeated experiment on the test function $f_1 - f_6$ and the

TABLE 4. Test function results of each algorithm when $N = 10$.

Algorithm	Metric	f_3	f_4	f_5	f_6
SPSO	*Mean*	0.1742	1169.3825	145.8350	0.0044
	Variance	0.0107	154675.2166	593674.6614	3.267e-5
	Standard deviation	0.1051	400.0111	783.6748	0.0058
LDWPSO	*Mean*	0.2445	1046.7057	298.155	0.0003
	Variance	0.0142	151446.8204	1143092.9167	1.026e-6
	Standard deviation	0.1211	395.8145	1087.4327	0.0010
ChPSO	*Mean*	0.1987	1169.1536	2.065	6.587e-23
	Variance	0.0128	234712.1146	118.7755	1.982e-8
	Standard deviation	0.1151	492.7531	11.0847	0.0001
CMPSO	*Mean*	0.0261	1412.3984	858.993	1.0122
	Variance	0.0261	230606.9130	4181261.9900	12.0941
	Standard deviation	0.1644	488.4249	2079.7700	3.5371
CAPSO	*Mean*	**0.0751**	**541.2722**	**1.877e-8**	**6.425e-9**
	Variance	**0.0008**	**182994.1853**	**4.707e-15**	**9.248e-16**
	Standard deviation	**0.0293**	**435.0912**	**6.978e-8**	**3.093e-8**

actual global optimal solution of each algorithm are shown in Table 3, where the bolded data represent the smallest value of the algorithm on the corresponding function.

It can be concluded that although the variance and standard deviation results of CAPSO on the *BUKIN* function are not optimal, they are not much different from the results of LDWPSO. In other test functions, CAPSO's indicators are better than other algorithms. On the whole, the convergence accuracy of CAPSO is significantly better than other algorithms. In addition, although CAPSO has not reached the global optimal solution on the *SPHERE* function and *GRIEWANK* function, it has been infinitely close to the global optimal solution, indicating that CAPSO is more stable and has a lower probability of falling into the local optimal solution.

To evaluate the convergence, we take the *GRIEWANK* function as an example. The average change of the best fitness value obtained by each algorithm repeated 30 times on this function is shown in the curve drawn in FIGURE 5. According to the curve change, CAPSO is slower in the early stage and faster in the later stage. It is better than the SPSO and LDWPSO. Generally speaking, it is still within the acceptable range, and the most important thing is that CAPSO has the highest convergence accuracy.

When $N = 10$, since f_1 and f_2 are two-dimensional functions, we repeat the experiment 30 times on the $f_3 - f_6$ test function. The results of the mean, variance, and standard deviation of the difference between the fitness value of the repeated experiment and the actual global optimal solution of each algorithm are shown in Table 4, where the bolded data represent the smallest value of the algorithm on the corresponding function.

It can be seen from Table 4 that CAPSO has obvious advantages over other algorithms in terms of mean, variance, and standard deviation. Although the effect of CAPSO on the *GRIEWANK* function is not satisfactory, the results on the *SPHERE* function and the *SCHWEFEL* function are approximately 0, indicating that CAPSO is infinitely close to or reaches the global optimal solution in 30 repeated experiments. On the whole, the CAPSO algorithm we proposed has higher convergence accuracy, stronger stability, and it is easier to search for the global optimal solution.

TABLE 5. Parameter configuration information of Peer algorithms.

Peer Algorithms	Configuration Information
HCLPSO [19]	$w = 0.99$, $c_1 = 2.5$, $c_2 = 0.5$, $a = 0$, $b = 0.25$
GLPSO [28]	$w = 0.7298$, $c_1 = c_2 = 1.49618$, $sg = 7$, $pm = 0.01$
TAPSO [29]	$w = 0.7298$, $p_m = 0.02$, $p_c = 0.5$
AWPSO [20]	$w = 0.9$, $a = 3.5 \times 10^{-5}$, $b = 0.5$, $c = 0$, $d = 1.5$
MAPSO [30]	$N_P = 3$, $A_E = N_S$, $IG = 0$, $SG = 0$

D. CEC2013 TEST SUITE

To further verify the effectiveness of CAPSO, we conduct experiments on the CEC2013 test suite to verify the performance of the proposed algorithm in different environments. It should be noted that in CEC2013, f_1-f_5 are unimodal functions, f_6-f_{20} are multimodal functions, and $f_{21} - f_{28}$ are combined functions. Moreover, we set dimensions $N = 10$ and $N = 50$ respectively to verify the scalability of CAPSO.

1) PEER ALGORITHMS

We selected 5 PSO variants that were widely used in CEC2013. To ensure the rigor and fairness of the comparative experiments, all relevant parameters of peer algorithms are set according to the recommendations in the original paper. In addition, we ensure that all algorithms are experimented with in the same environment to remove the effects of any random errors. All peer algorithms and corresponding configuration information are recorded in Table 5.

2) PERFORMANCE ON CEC2013

In Table 6 and Table 7, the mean and standard deviation of the peer algorithms on the CEC2013 suite when $N = 10$ and $N = 50$ are recorded respectively. By comparison, we can know the effectiveness of CAPSO. It can be found that CAPSO almost all shows the best performance on unimodal

TABLE 6. Test results of CAPSO and peer algorithms on CEC2013 suite ($N = 10$).

Function	Metric	Peer Algorithms					
		HCLPSO	GLPSO	TAPSO	AWPSO	MAPSO	CAPSO
f_1	Mean	0	0	0	1.29e-13	0	**0**
	Standard deviation	0	0	0	1.12e-13	0	**0**
f_2	Mean	1.59e5	2.20e4	1.73e5	1.33e5	3.17e4	**2.03e4**
	Standard deviation	1.05e5	1.07e4	1.68e5	1.14e5	2.22e4	**9.54e3**
f_3	Mean	1.69e5	4.79e6	1.10e6	8.03e8	1.88e4	**1.25e4**
	Standard deviation	2.18e5	6.89e6	1.99e6	1.34e9	3.38e4	**1.68e4**
f_4	Mean	1.31e3	1.94e3	360	732	52.2	**26.7**
	Standard deviation	738	1.35e3	353	931	48.3	**23.9**
f_5	Mean	5.80e-14	0	7.58e-15	30.5	0	**0**
	Standard deviation	5.68e-14	0	1.41e-14	44.7	0	**0**
f_6	Mean	0.815	6.17	5.38	21.4	7.13	**3.87e-4**
	Standard deviation	0.12	4.57	5.13	19.9	3.89	**3.31e-4**
f_7	Mean	1.77	4.12	2.32	20.4	1.11	**0.733**
	Standard deviation	1.03	4.08	2.65	18.2	1.32	**0.628**
f_8	Mean	20.3	20.4	20.4	20.3	20.4	**20.3**
	Standard deviation	0.0639	0.0781	0.0406	0.0503	0.0531	**0.0406**
f_9	Mean	2.42	2.71	2.51	3.48	1.73	**1.48**
	Standard deviation	0.822	0.984	0.891	1.13	0.87	**0.672**
f_{10}	Mean	0.308	0.231	0.256	23.2	**0.151**	0.234
	Standard deviation	0.149	0.0826	0.0877	27.4	**0.0677**	0.161
f_{11}	Mean	0.117	0.195	2.85	3.68	0.507	**8.94e-4**
	Standard deviation	0.211	0.321	1.07	1.81	0.617	**7.15e-4**
f_{12}	Mean	8.53	11.5	9.17	16.2	10.8	**4.55**
	Standard deviation	2.76	4.6	3.3	5.32	3.54	**2.07**
f_{13}	Mean	17.5	26.7	14	21.1	15.8	**10.8**
	Standard deviation	5.36	8.18	4.88	4.49	6.47	**2.83**
f_{14}	Mean	44.1	**0.818**	290	309	6.27	4.18
	Standard deviation	55.5	**1.03**	120	123	5.16	2.73
f_{15}	Mean	525	671	422	614	617	**382**
	Standard deviation	148	255	179	163	170	**136**
f_{16}	Mean	0.402	0.309	1.13	0.902	0.458	**0.104**
	Standard deviation	0.14	0.158	0.15	0.218	0.163	**0.08**
f_{17}	Mean	9.81	10.8	16.3	12.8	11	**5.2**
	Standard deviation	1.1	0.304	2.23	2.44	0.387	**0.211**
f_{18}	Mean	20.8	21.3	29.6	24.5	16.5	**14.33**
	Standard deviation	3.91	4.41	5.99	5.88	1.95	**0.81**
f_{19}	Mean	0.359	0.6	0.608	8.35	0.692	**0.199**
	Standard deviation	0.0834	0.121	0.175	14.4	0.174	**0.0438**
f_{20}	Mean	2.31	2.97	2.23	2.73	2.07	**1.94**
	Standard deviation	0.4	0.451	0.581	0.697	0.461	**0.317**
f_{21}	Mean	261	290	300	300	300	**257**
	Standard deviation	63.1	18.8	0	3.41e-14	0	**0**
f_{22}	Mean	59.1	66.5	319	360	75.9	**46.9**
	Standard deviation	58	56.5	117	188	50.3	**43.1**
f_{23}	Mean	665	932	609	857	566	**531**
	Standard deviation	186	325	310	234	250	**178**
f_{24}	Mean	140	210	204	218	202	**108**
	Standard deviation	29.9	4.36	8.84	4.33	6.24	**2.15**
f_{25}	Mean	181	204	204	206	203	**153**
	Standard deviation	30.8	6.81	8.39	3.84	3.41	**2.34**
f_{26}	Mean	136	166	129	188	186	**118**
	Standard deviation	35.3	61.1	57.6	64.6	**23.2**	43.7
f_{27}	Mean	323	359	302	456	304	**281**
	Standard deviation	17.1	48.5	74.3	63.6	7.09	**6.15**
f_{28}	Mean	265	385	241	301	292	**230**
	Standard deviation	58.1	117	57.7	54.7	15.1	**10.9**

functions, multimodal functions, and combined functions with high optimization difficulty. Furthermore, CAPSO also displays better reliability when the test function is extended to higher dimensions. It is mainly since CAPSO adaptively adjusts the inertia weight and acceleration coefficient, and adaptively adjusts the search range through the control factor

TABLE 7. Test results of CAPSO and peer algorithms on CEC2013 suite ($N = 50$).

Function	Metric	Peer Algorithms					
		HCLPSO	GLPSO	TAPSO	AWPSO	MAPSO	CAPSO
f_1	Mean	7.04e-13	1.96e-13	2.03e-13	4.12e3	8.47e-14	**2.81e-14**
	Standard deviation	8.04e-14	5.38e-14	7.28e-14	1.84e3	1.06e-13	**8.15e-15**
f_2	Mean	2.65e6	9.03e5	1.22e7	1.97e7	2.31e6	**7.39e5**
	Standard deviation	6.49e5	2.63e5	6.74e6	1.27e7	1.20e6	**1.08e5**
f_3	Mean	2.32e8	8.75e7	3.79e7	2.95e10	3.67e7	**1.34e7**
	Standard deviation	1.67e8	8.56e7	6.50e7	1.79e10	2.39e7	**8.68e6**
f_4	Mean	2.13e3	2.17e3	687	5.58e3	27.3	**12.1**
	Standard deviation	628	1.38e3	162	4.96e3	23.9	**10.7**
f_5	Mean	8.02e-13	1.34e-13	3.06e-13	1.92e3	1.20e-13	**1.03e-13**
	Standard deviation	1.25e-13	3.30e-14	1.79e-13	906	1.26e-14	**9.37e-15**
f_6	Mean	44.6	49.1	44.1	241	43.7	**40.6**
	Standard deviation	0.607	9.3	5.02	93.5	0.431	**0.354**
f_7	Mean	42.4	45.9	26.8	104	36.5	**22.2**
	Standard deviation	8.86	9.94	6.33	25.3	10.6	**5.79**
f_8	Mean	21.1	21.1	21.1	21.1	21.1	**21.1**
	Standard deviation	0.0286	0.0549	0.0338	0.0354	0.027	**0.0236**
f_9	Mean	41.6	31.6	26.6	41.2	28.3	**23.7**
	Standard deviation	5.35	3.36	4.34	**2.49**	3.87	3.15
f_{10}	Mean	0.244	0.179	0.0905	977	0.0987	**0.0733**
	Standard deviation	0.117	0.0574	0.0427	518	0.0449	**0.0406**
f_{11}	Mean	2.31e-12	4.68e-14	64.7	155	13.2	**8.65**
	Standard deviation	2.73e-12	1.65e-14	10.6	41.7	3.14	**2.39**
f_{12}	Mean	127	108	114	220	85.4	**61**
	Standard deviation	16.6	21.5	40.4	38.4	15.6	**12.8**
f_{13}	Mean	248	238	211	356	162	**132**
	Standard deviation	40.1	40	48.9	43.6	26.8	**16.6**
f_{14}	Mean	3.24	4.64	2.20e3	3.41e3	74.8	**0.49**
	Standard deviation	1.14	1.99	484	515	46.6	**0.181**
f_{15}	Mean	6.99e3	7.34e3	6.75e3	8.07e3	7.01e3	**5.94e3**
	Standard deviation	663	692	2.34e3	1.50e3	702	**628**
f_{16}	Mean	1.8	**0.691**	3.28	2.73	1.65	1.17
	Standard deviation	0.285	0.302	0.261	0.308	0.401	**0.239**
f_{17}	Mean	51.3	52.8	141	175	78.1	**45.7**
	Standard deviation	0.107	0.577	15	43	6.99	**0.0893**
f_{18}	Mean	173	144	340	365	137	**135**
	Standard deviation	28	16.5	64.6	89.6	45.5	**11.8**
f_{19}	Mean	2.13	2.8	7.34	1.92e4	4.72	**1.59**
	Standard deviation	0.353	0.388	1.13	2.73e4	0.837	**0.33**
f_{20}	Mean	20	20.6	20.9	23.2	18.8	**15.6**
	Standard deviation	0.833	0.759	0.758	1.75	0.768	**0.732**
f_{21}	Mean	**239**	851	800	1.18e3	890	308
	Standard deviation	**47**	228	239	259	252	106
f_{22}	Mean	34.7	42.5	2.94e3	4.59e3	94.4	**20.8**
	Standard deviation	21.9	37.2	488	784	43.1	**15.9**
f_{23}	Mean	8.23e3	8.21e3	8.13e3	8.82e3	6.69e3	**3.84e3**
	Standard deviation	809	**113**	1.67e3	1.17e3	778	450
f_{24}	Mean	281	283	312	341	274	**132**
	Standard deviation	13.4	9.54	16.3	11.4	9.56	**7.81**
f_{25}	Mean	357	323	336	380	329	**315**
	Standard deviation	12.2	10.1	15.2	9.91	17.1	**6.16**
f_{26}	Mean	200	352	361	379	323	**153**
	Standard deviation	**0.0646**	47.5	427	55.9	67.2	15.8
f_{27}	Mean	1.22e3	1.17e3	1.29e3	1.53e3	1.01e3	**831**
	Standard deviation	152	87.1	90	93	127	**72.9**
f_{28}	Mean	400	641	817	2.29e3	400	**400**
	Standard deviation	2.02e-11	445	723	890	**4.55e-13**	6.88e-12

of chaos theory. Therefore, CAPSO has better adaptability, and can easily jump out of the local optimal solution and approach the global optimal solution infinitely. To sum up, our proposed CAPSO algorithm is effective.

VI. CONCLUSION AND FUTURE

Based on the in-depth research and analysis of traditional particle swarm optimization algorithms, this paper aims to deal with complex function optimization problems and practical

applications that are prone to poor convergence accuracy and the inability to effectively obtain global optimization. On the basis of chaos theory, we propose a chaotic adaptive particle swarm optimization (CAPSO) algorithm. To prove the stability, convergence speed, and accuracy of CAPSO, experiments are performed on 6 test functions with other algorithms. The comparative analysis results show that although CAPSO has a slight deficiency in the convergence speed, its convergence accuracy is higher, the stability is stronger, and it is not easy to fall into the local optimum. To further prove the effectiveness and scalability of CAPSO, extensive experiments are performed on the CEC2013 test suite. All results comprehensive prove CAPSO has achieved satisfactory performance.

Furthermore, CAPSO achieves advanced retrieval accuracy due to a series of adaptive computations. However, its convergence speed, although within an acceptable range, is still slightly slower. In the future, we hope to further simplify the search process of the algorithm based on the adaptive adjustment strategy to improve the convergence speed. Moreover, due to the advantages of CAPAO in terms of convergence, stability, and accuracy, we believe that it will play a role in resource scheduling, load problems, system optimization, and other fields. In the follow-up work, we also expect to work with other researchers to further explore and make breakthroughs in parameter sensitivity, high-dimensional solution space, multi-objective optimization, etc.

REFERENCES

[1] H. Han, X. Bai, H. Han, Y. Hou, and J. Qiao, "Self-adjusting multitask particle swarm optimization," *IEEE Trans. Evol. Comput.*, vol. 26, no. 1, pp. 145–158, Feb. 2022.

[2] Z.-M. Gao, J. Zhao, Y.-R. Hu, and H.-F. Chen, "The challenge for the nature-inspired global optimization algorithms: Non-symmetric benchmark functions," *IEEE Access*, vol. 9, pp. 106317–106339, 2021.

[3] V. Nagireddy, P. Parwekar, and T. K. Mishra, "Velocity adaptation based PSO for localization in wireless sensor networks," *Evol. Intell.*, vol. 14, no. 2, pp. 243–251, Jun. 2021.

[4] C. Shang, J. Gao, H. Liu, and F. Liu, "Short-term load forecasting based on PSO-KFCM daily load curve clustering and CNN-LSTM model," *IEEE Access*, vol. 9, pp. 50344–50357, 2021.

[5] X. Liu, P. Zhang, H. Fang, and Y. Zhou, "Multi-objective reactive power optimization based on improved particle swarm optimization with ϵ-greedy strategy and Pareto archive algorithm," *IEEE Access*, vol. 9, pp. 65650–65659, 2021.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE ICNN*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.

[7] E. Iranmehr, S. B. Shouraki, and M. M. Faraji, "Unsupervised feature selection for phoneme sound classification using particle swarm optimization," in *Proc. 5th Iranian Joint Congr. Fuzzy Intell. Syst. (CFIS)*, Mar. 2017, pp. 86–90.

[8] J.-H. Seo, C.-H. Im, C.-G. Heo, J.-K. Kim, H.-K. Jung, and C.-G. Lee, "Multimodal function optimization based on particle swarm optimization," *IEEE Trans. Magn.*, vol. 42, no. 4, pp. 1095–1098, Apr. 2006.

[9] G. Lei, "Application improved particle swarm algorithm in parameter optimization of hydraulic turbine governing systems," in *Proc. IEEE 3rd Inf. Technol. Mechatronics Eng. Conf. (ITOEC)*, Oct. 2017, pp. 1135–1138.

[10] A. Marjani, S. Shirazian, and M. Asadollahzadeh, "Topology optimization of neural networks based on a coupled genetic algorithm and particle swarm optimization techniques (c-GA–PSO-NN)," *Neural Comput. Appl.*, vol. 29, no. 11, pp. 1073–1076, Jun. 2018.

[11] M. Bouzidi, M. E. Riffi, and A. Serhir, "Discrete particle swarm optimization for travelling salesman problems: New combinatorial operators," in *Proc. Int. Conf. Soft Comput. Pattern Recognit.* Cham, Switzerland: Springer, 2017, pp. 141–150.

[12] S. N. Ghorpade, M. Zennaro, B. S. Chaudhari, R. A. Saeed, H. Alhumyani, and S. Abdel-Khalek, "A novel enhanced quantum PSO for optimal network configuration in heterogeneous industrial IoT," *IEEE Access*, vol. 9, pp. 134022–134036, 2021.

[13] T. M. Shami, A. A. El-Saleh, M. Alswaitti, Q. Al-Tashi, M. A. Summakieh, and S. Mirjalili, "Particle swarm optimization: A comprehensive survey," *IEEE Access*, vol. 10, pp. 10031–10061, 2022.

[14] X. Ji, Y. Zhang, D. Gong, and X. Sun, "Dual-surrogate-assisted cooperative particle swarm optimization for expensive multimodal problems," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 794–808, Aug. 2021.

[15] W. Zhang, J. Ma, L. Wang, and F. Jiang, "Particle-swarm-optimization-based 2D output feedback robust constraint model predictive control for batch processes," *IEEE Access*, vol. 10, pp. 8409–8423, 2022.

[16] Z. Zhang, H. Chen, Y. Yu, F. Jiang, and Q. S. Cheng, "Yield-constrained optimization design using polynomial chaos for microwave filters," *IEEE Access*, vol. 9, pp. 22408–22416, 2021.

[17] Y. Hu, F. Zhu, L. Zhang, Y. Lui, and Z. Wang, "Scheduling of manufacturers based on chaos optimization algorithm in cloud manufacturing," *Robot. Comput.-Integr. Manuf.*, vol. 58, pp. 13–20, Aug. 2019.

[18] L. Zhang, H. Wang, J. Liang, and J. Wang, "Decision support in cancer base on fuzzy adaptive PSO for feedforward neural network training," in *Proc. Int. Symp. Comput. Sci. Comput. Technol.*, vol. 1, 2008, pp. 220–223.

[19] N. Lynn and P. N. Suganthan, "Heterogeneous comprehensive learning particle swarm optimization with enhanced exploration and exploitation," *Swarm Evol. Comput.*, vol. 24, pp. 11–24, Oct. 2015.

[20] W. Liu, Z. Wang, Y. Yuan, N. Zeng, K. Hone, and X. Liu, "A novel sigmoid-function-based adaptive weighted particle swarm optimizer," *IEEE Trans. Cybern.*, vol. 51, no. 2, pp. 1085–1093, Feb. 2021.

[21] M. Lin, Z. Wang, F. Wang, and D. Chen, "Improved simplified particle swarm optimization based on piecewise nonlinear acceleration coefficients and mean differential mutation strategy," *IEEE Access*, vol. 8, pp. 92842–92860, 2020.

[22] Y. Zhang, D.-W. Gong, X.-Y. Sun, and N. Geng, "Adaptive bare-bones particle swarm optimization algorithm and its convergence analysis," *Soft Comput.*, vol. 18, no. 7, pp. 1337–1352, Jul. 2014.

[23] N. Kumar and D. P. Vidyarthi, "A model for resource-constrained project scheduling using adaptive PSO," *Soft Comput.*, vol. 20, no. 4, pp. 1565–1580, 2016.

[24] U. K. Acharya and S. Kumar, "Particle swarm optimization exponential constriction factor (PSO-ECF) based channel equalization," in *Proc. 6th Int. Conf. Comput. Sustain. Global Develop.*, 2019, pp. 94–97.

[25] C.-M. Yan, G.-Y. Lu, Y.-T. Liu, and X.-Y. Deng, "A modified PSO algorithm with exponential decay weight," in *Proc. 13th Int. Conf. Natural Comput., Fuzzy Syst. Knowl. Discovery (ICNC-FSKD)*, Jul. 2017, pp. 239–242.

[26] L.-Y. Chuang, C.-J. Hsiao, and C.-H. Yang, "Chaotic particle swarm optimization for data clustering," *Expert Syst. Appl.*, vol. 38, no. 12, pp. 14555–14563, Nov. 2011.

[27] Z. Wan, G. Wang, and B. Sun, "A hybrid intelligent algorithm by combining particle swarm optimization with chaos searching technique for solving nonlinear bilevel programming problems," *Swarm Evol. Comput.*, vol. 8, pp. 26–32, Feb. 2013.

[28] Y.-J. Gong, J.-J. Li, Y. Zhou, Y. Li, H. S.-H. Chung, Y.-H. Shi, and J. Zhang, "Genetic learning particle swarm optimization," *IEEE Trans. Cybern.*, vol. 46, no. 10, pp. 2277–2290, Oct. 2016.

[29] X. Xia, L. Gui, F. Yu, H. Wu, B. Wei, Y.-L. Zhang, and Z.-H. Zhan, "Triple archives particle swarm optimization," *IEEE Trans. Cybern.*, vol. 50, no. 12, pp. 4862–4875, Dec. 2020.

[30] B. Wei, X. Xia, F. Yu, Y. Zhang, X. Xu, H. Wu, L. Gui, and G. He, "Multiple adaptive strategies based particle swarm optimization algorithm," *Swarm Evol. Comput.*, vol. 57, Sep. 2020, Art. no. 100731. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2210650220303849

[31] M. M. Golchi, S. Saraeian, and M. Heydari, "A hybrid of firefly and improved particle swarm optimization algorithms for load balancing in cloud environments: Performance evaluation," *Comput. Netw.*, vol. 162, Oct. 2019, Art. no. 106860.

[32] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Proc. Int. Conf. Evol. Program.* Berlin, Germany: Springer, 1998, pp. 591–600.

[33] R. C. Eberhart and Y. Shi, "Comparing inertia weights and constriction factors in particle swarm optimization," in *Proc. Congr. Evol. Comput.*, vol. 1, Jul. 2000, pp. 84–88.

[34] M. Clerc, "The swarm and the queen: Towards a deterministic and adaptive particle swarm optimization," in *Proc. Congr. Evol. Comput.*, vol. 3, Jul. 1999, pp. 1951–1957.
[35] P. N. Suganthan, "Particle swarm optimiser with neighbourhood operator," in *Proc. Congr. Evol. Comput.*, vol. 3, Jul. 1999, pp. 1958–1962.
[36] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, Jun. 2004.
[37] Y. Shi and R. C. Eberhart, "Empirical study of particle swarm optimization," in *Proc. Congr. Evol. Comput.*, vol. 3, Jul. 1999, pp. 1945–1950.
[38] M. Cheng and Y. Han, "Application of a modified CES production function model based on improved PSO algorithm," *Appl. Math. Comput.*, vol. 387, Dec. 2020, Art. no. 125178.
[39] X. Liu, Y. Gu, S. He, Z. Xu, and Z. Zhang, "A robust reliability prediction method using weighted least square support vector machine equipped with chaos modified particle swarm optimization and online correcting strategy," *Appl. Soft Comput.*, vol. 85, Dec. 2019, Art. no. 105873.

YOUXIANG DUAN received the B.S. degree from the College of Computer and System Science, Nankai University, in 1986, and the Ph.D. degree from the School of Geosciences, China University of Petroleum (East China), Qingdao, China, in 2017. He is currently a Professor with the College of Computer Science and Technology, China University of Petroleum (East China). His research interests include service computing, intelligent control, and machine learning.

NING CHEN is currently pursuing the master's degree with the College of Computer Science and Technology, China University of Petroleum (East China). His research interests include cross-modal retrieval, deep learning, particle swarm optimization, network virtualization, blockchain, and microservice management.

LUNJIE CHANG is currently a Professor-Level Senior Engineer at the Petroleum Exploration and Development Research Institute, PetroChina Tarim Oilfield Company. He has published more than 40 papers in journals or conferences, such as *Xinjiang Petroleum Geology*, *Natural Gas Geoscience, Geophysical and Geochemical Exploration Computing Technology*, *Petroleum Exploration and Development*, *Earth Science Frontiers*, and *Oil and Gas Geology*. His research interests mainly include geology, mining engineering technology, oil and gas field well development engineering, composite films, computer architecture, fracture, and other directions.

YONGJING NI was born in 1981. She is currently pursuing the Ph.D. degree with Yanshan University. She is a Lecturer at the Hebei University of Science and Technology. Her research interests include computer networks and signal processing.

S. V. N. SANTHOSH KUMAR received the B.E. degree in computer science and engineering and the M.E. degree in software engineering from Anna University, Chennai, India, in 2011 and 2013, respectively. He is currently an Assistant Professor with VIT, Vellore Campus, India. He works in the areas of security and data dissemination in wireless sensor networks. He does research in information systems (business informatics), computer communications (networks), and computer security and reliability. He has published 20 articles in reputed international journals and conferences. His research interests include wireless sensor networks, the Internet of Things, and mobile computing. He is a Peer Reviewer of *Peer-to-Peer Networking and Applications* (Springer), IEEE TRANSACTIONS ON COMPUTATIONAL SOCIAL SYSTEMS, and *Wireless Personal Communications*.

PEIYING ZHANG is currently pursuing the Ph.D. degree in information and communication engineering with the State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing, China. He is an Associate Professor with the College of Computer and Communication Engineering, China University of Petroleum (East China), Qingdao, China. Since 2016, he has been publishing multiple IEEE/ACM TRANSACTIONS/journal/magazine articles, such as IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS, IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY, IEEE TRANSACTIONS ON NETWORK SCIENCE AND ENGINEERING, IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTING, *IEEE Network*, IEEE ACCESS, IEEE INTERNET OF THINGS JOURNAL, *ACM TALLIP, COMPUT COMMUN*, and *IEEE Communications Magazine*. His current research interests include semantic computing, deep learning, network virtualization, and future network architecture. He has served the Technical Program Committee for ISCIT 2016, ISCIT 2017, ISCIT 2018, ISCIT 2019, GLOBECOM 2019, COMNETSAT 2020, SoftIoT 2021, CBIoT 2021, DPRR 2021, IWCMC-Satellite 2019, and IWCMC-Satellite 2020.