Research Article

A Hybrid Dynamic Probability Mutation Particle Swarm Optimization for Engineering Structure Design

Qiuyu Li and Zhiteng Ma

School of Mechanical Electrical & Information Engineering, Shandong University, Weihai 264209, China

Correspondence should be addressed to Zhiteng Ma; 201600800304@mail.sdu.edu.cn

Received 10 October 2020; Revised 31 January 2021; Accepted 3 February 2021; Published 25 February 2021

Academic Editor: Juan C. Cano

Copyright © 2021 Qiuyu Li and Zhiteng Ma. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Particle swarm optimization (PSO) is a common metaheuristic algorithm. However, when dealing with practical engineering structure optimization problems, it is prone to premature convergence during the search process and falls into a local optimum. To strengthen its performance, combining several ideas of the differential evolution algorithm (DE), a dynamic probability mutation particle swarm optimization with chaotic inertia weight (CWDEPSO) is proposed. The main improvements are achieved by improving the parameters and algorithm mechanism in this paper. The former proposes a novel inverse tangent chaotic inertia weight and sine learning factors. Besides, the scaling factor and crossover probability are improved by random distributions, respectively. The latter introduces a monitoring mechanism. By monitoring the convergence of PSO, a developed mutation operator with a more reliable local search capability is adopted and increases population diversity to help PSO escape from the local optimum effectively. To evaluate the effectiveness of the CWDEPSO algorithm, 24 benchmark functions and two groups of engineering optimization experiments are used for numerical and engineering optimization, respectively. The results indicate CWDEPSO offers better convergence accuracy and speed compared with some well-known metaheuristic algorithms.

1. Introduction

Recently, an increasing number of fields, such as feature selection [1], artificial intelligence [2], and engineering structure design [3], have been faced with optimization problems due to the development of science and technology and industrialization level development. Among them, the engineering structure optimization problem is difficult to be solved by a specific algorithm because of extensive and complex constraints. Since the traditional optimization methods take too much time to solve these problems, thus making it difficult to find the optimal solution, people have started researching new optimization algorithms with a broader applicability. In the past two decades, physical or biological phenomena have provided inspiration. Researchers have consequently invented many optimization algorithms, like the whale optimization (WOA) algorithm [4], biogeography-based optimization (BBO) [5], the sine-cosine algorithm (SCA) [6], the moth-flame optimization

(MFO) algorithm [7], ant colony optimization (ACO) [8], krill herd (KH) algorithm [9, 10], the artificial bee colony (ABC) [11], the gravitational search algorithm (GSA) [12], monarch butterfly optimization (MBO) [13, 14], earthworm optimization algorithm (EWA) [15], elephant herding optimization (EHO) [16, 17], moth search (MS) algorithm [18], slime mould algorithm (SMA) [19], Harris hawks optimization (HHO) [20], differential evolution (DE) algorithm, [21] and particle swarm optimization (PSO) [22, 23]. These optimization algorithms are also known as metaheuristic algorithms and are now widely used in real life.

PSO is considered as a swarm intelligence algorithm put forward by the American computer intelligence researcher Eberhart and psychological researcher Kennedy in 1995 [22] and is derived from the foraging behavior of bird populations. PSO is not only robust and efficient but also easily implemented. Therefore, PSO is widely used in practical engineering optimization. However, it is susceptible to premature convergence and falls into local

optimal solutions when dealing with complicated multimodal optimization or multiconstraint conditions. To overcome these shortcomings, several scholars have proposed various PSO variants. Liang et al. made an introduction to comprehensive learning PSO (CLPSO) [24], which combined an original learning method. Liu and Gao proposed chaos PSO (CPSO) [25], which improved the performance of PSO by introducing chaos mechanism. Wang et al. proposed an adaptive granularity learning distributed particle swarm optimization (AGLDPSO) [26] with the help of machine-learning techniques, including clustering analysis based on locality-sensitive hashing (LSH) and adaptive granularity control based on logistic regression (LR). To avoid premature convergence, Jordehi proposed a time-varying acceleration coefficient PSO (TVACPSO) [27]. A kind of EC algorithm called local version particle swarm optimization (PSO) is adopted by Li et al. to implement a pipeline-based parallel PSO (PPPSO, i.e., P^3 SO) [28]. Fan and Jen attempted to develop the basic PSO effectiveness and convergence by adopting the thought of multiple cooperative swarms, and then, an enhanced partial search PSO (EPS-PSO) [29] was proposed. Ma et al. proposed the chaotic PSO-arctangent acceleration coefficient algorithm (CPSO-AT) [30], which strengthened the effectiveness of the original PSO through chaotic initialization and nonlinear optimization. Zhang et al. proposed a cooperative coevolutionary bare-bones particle swarm optimization (CCBBPSO) with function independent decomposition (FID), called CCBBPSO-FID [31]. Liu et al. proposed a coevolutionary particle swarm optimization with a bottleneck objective learning (BOL) strategy for many-objective optimization (CPSO) [32]. Xia et al. proposed a triple archives PSO (TAPSO) [33]. Although the convergence accuracy of these algorithms is higher compared with the basic PSO, their convergence and robustness are still unsatisfactory.

Similar to PSO, DE is also a typical swarm intelligence algorithm and possesses strong points of simple rules, excellent robustness, quick convergence speed, and easy implementation. Nevertheless, the individual convergence is too strong, which makes DE very easy to be attracted by local interference extreme values in the late evolution, leading to precocity. For the purpose of overcoming such issues, many scholars have done many works on adaptive aspects of DE, such as self-adaptive DE with developed mutation method (IMSaDE) [34], multipopulation ensemble DE (MPEDE) [35], self-adapting control parameters in DE (jDE) [36], and self-adaptive DE (SaDE) [37]. In addition, Zhao et al. proposed an LBP-based adaptive DE (LBPADE) algorithm [38]. Wang et al. proposed a new automatic niching technique based on the affinity propagation clustering (APC) and design a novel niching differential evolution (DE) algorithm, termed as automatic niching DE (ANDE) [39]. Zhan et al. developed a novel double-layered heterogeneous DE algorithm and realized it in cloud computing distributed environment (CloudDE) [40]. Wang et al. proposed a dynamic group learning distributed particle swarm optimization (DGLDPSO) [41] for large-scale

optimization and extended it for the large-scale cloud workflow scheduling. Liu et al. developed a novel variant of adaptive DE, utilizing both the historical experience and heuristic information for the adaptation (HHDE) [42]. Zhan et al. proposed an adaptive DDE (ADDE) [43] to relieve the sensitivity of strategies and parameters. Chen et al. developed a distributed individual differential evolution (DIDE) algorithm [44] based on a distributed individual for multiple peaks (DIMPs) framework and two novel mechanisms. The most classical one is adaptive DE with the optional external archive (JADE) [45] put forward by Zhang and Sanderson, which uses a Cauchy and Gaussian distributions to dynamically update and improve population convergent speed to the optimal value. Although these modifications are better than traditional DE, the parameter improvement depends too much on previous historical experience, and there are many problems such as slow convergence.

To overcome the limitations of a single algorithm, scholars have combined PSO and DE and conducted much research on this kind of hybrid algorithms, such as self-adaptive mutation DE based on PSO (DEPSO) [46], ensemble PSO and DE (EPSODE) [47], combining DE and PSO (DEPSO) [48], and hybridizing PSO with DE (PSO-DE) [49]. The existing hybrid algorithms can be roughly classified as follows: two algorithms are used in sequence, two algorithms solve and communicate with two different populations, respectively, and two algorithms evaluate and select optimal values. These hybrid algorithms inherit the advantages of each algorithm, which is able to promote basic PSO and DE optimization performance. However, the requirements for more computing resources and slower convergence speed are still the main problems. Based on the above, we combine some ideas of DE and propose a dynamic probability mutation particle swarm optimization with chaotic inertia weight (CWDEPSO), which improves the parameters of PSO and DE. It adds chaotic processing to the inertial weight. Once it is found that PSO cannot successfully update the optimal value, a different mutation operator is adopted for helping PSO run away from local extrema, thereby avoiding the premature maturity of the algorithm and enhancing the population distribution entropy that aims at achieving the overall balance between the local search capability and the global search capability. Among them, the distribution entropy of the population represents the disorder degree of the population's species and the population diversity [50]. Generally, individuals are scattered in the early stage of iteration, so the degree of disorder is high, and the population diversity is high. At the end of the iteration, individuals are concentrated in the area with strong fitness, so the degree of disorder is low, and the population diversity is low. The population diversity can be improved by adding a mutation operator [51]. The contributions of this paper are as follows:

- (1) The paper proposes a different mutation operator DE/current-to-gbest/1 and a novel dynamic probability mutation strategy, and the former is applied to the latter.

- (2) In the selection stage, to fully exploit the individual value in the population, an original selection strategy is proposed, which is to introduce a candidate population to reuse the culled individuals.
- (3) The paper introduces a method of using logistic chaotic maps, inverse tangent, and cosine trigonometric functions and random distributions to optimize the parameters, respectively.
- (4) CWDEPSO is applied to optimize the actual engineering structure. CWDEPSO proposed in this paper is used to optimize the spring tension structure and the pressure vessel structure.

24 benchmark test functions and two groups of engineering optimization experiments are adopted for testing the algorithm to evaluate its performance. The results suggest that the performance of CWDEPSO is better compared with PSO [22], DE [21], AIWCPSO [52], MPSPSO-ST [53], JADE [45], SinDE [54], and DEPSO [55] and five common metaheuristic algorithms GSA [12], ABC [11], MFO [7], SCA [6], and BBO [5]. Besides, it achieves the best performance in practical engineering optimization problems.

The remaining arrangements of the article are shown below. The second section is the introduction to the basic DE and PSO. The third section introduces the improvement of the parameters, and the fourth section presents the monitoring mechanisms and specific processes of CWDEPSO. The experimental part is shown in the fifth section. The sixth section uses CWDEPSO to solve the engineering structure optimization problem. Furthermore, the final section is the conclusion.

2. PSO and DE Algorithm

Two types of algorithms, including DE and PSO, are introduced in this section.

2.1. Particle Swarm Optimizer (PSO). As a metaheuristic algorithm, the position of each particle is the representative of a candidate solution within the PSO algorithm, and the fitness of solution relies on the fitness function. The search method is completed by iteration. During the iterative process, the individual particles have a memory function, which can record the best position X_{pbest} they have passed in the search procedure and combine the optimal position X_{gbest} found by the entire population for refining their position X_i^G and speed V_i^G , $i = 1, 2, \dots, NP$. The NP particles are searched for the optimal solution in D -dimensional space by iteration. Within the i -th particle of the G -th generation, the position and velocity have been represented by vectors $X_i^G = (x_{i1}^G, x_{i2}^G, \dots, x_{iD}^G)$ and $V_i^G = (v_{i1}^G, v_{i2}^G, \dots, v_{iD}^G)$, respectively. In the i -th particle of the current G generation, the greatest position that has undergone is $pbest_{i,d}^G$. In addition, the best position explored by the whole particle swarm is $gbest_d^G$. At the $(G + 1)$ -th iteration, the D -dimensional vector of the i -th particle velocity and position is able to be updated as follows:

$$V_{i,d}^{G+1} = \omega \cdot V_{i,d}^G + c_1 \cdot r_1 \cdot (gbest_d^G - X_{i,d}^G) + c_2 \cdot r_2 \cdot (pbest_d^G - X_{i,d}^G), \quad (1)$$

$$X_{i,d}^{G+1} = X_{i,d}^G + V_{i,d}^G, \quad (2)$$

where ω is the inertial weight; r_1 and r_2 are the random values in the interval $[0, 1]$; and c_1 and c_2 are the learning factors, with $c_1 = c_2 = 2$.

Shi and Eberhart [56] proposed a well-known PSO variant, which introduced a linearly decreasing inertial vector to balance well global and local searches, as shown in Figure 1. The definition of the update mechanism can be shown as follows:

$$\omega = \omega_{\max} - \frac{G}{G_{\max}} \cdot (\omega_{\max} - \omega_{\min}), \quad (3)$$

where $G_{\max}$ and G have been the iterations' maximum number and the current iterations' number, respectively.

2.2. Differential Evolution (DE) Optimizer. Being a random search method based on population, DE is divided into three steps: mutation, crossover, and selection to solve optimization problems. DE is adopted worldwide within various fields due to its simple principle, strong robustness, and few control parameters. This part mainly introduces the specific steps of the DE.

2.2.1. Population Initialization. Suppose there is a population containing NP individuals, and all the individuals can be represented by the vectors $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$, and D is the representative of the solution's dimension. In the process of evolution, the evolution times and the maximum evolution times are, respectively, represented by G and $G_{\max}$. The G -th generation is represented by a vector $X_i^G = (x_{i1}^G, x_{i2}^G, \dots, x_{iD}^G)$. Every dimensional lower and upper limits are represented by the vectors $X_l = (x_{l1}, x_{l2}, \dots, x_{lD})$ and $X_u = (x_{u1}, x_{u2}, \dots, x_{uD})$. The initial population is presented as follows:

$$x_{ij}^0 = x_{lj} + \text{rand}(0, 1) \times (x_{uj} - x_{lj}), \quad (4)$$

where $\text{rand}(0, 1)$ is a random number between the interval $[0, 1]$.

2.2.2. Mutation. In the G -th generation, each individual, the target vector $X_i^G = (x_{i1}^G, x_{i2}^G, \dots, x_{iD}^G)$, uses the mutation operator to produce the mutation vector $V_i^G = (v_{i1}^G, v_{i2}^G, \dots, v_{iD}^G)$. Figure 2 shows the classic mutation operator DE/rand/1. Selecting different mutation operators will influence the algorithm's performance greatly. Common mutation operators are as follows.

DE/rand/1:

$$V_i^G = X_{r1}^G + F \cdot (X_{r2}^G - X_{r3}^G). \quad (5)$$

FIGURE 1: Schematic diagram of PSO. ★ represents the optimal value.

FIGURE 2: Schematic diagram of DE/rand/1. ★ represents the optimal value.

DE/rand/2:

$$V_i^G = X_{r1}^G + F \cdot (X_{r2}^G - X_{r3}^G) + F \cdot (X_{r4}^G - X_{r5}^G). \quad (6)$$

$$u_{ij}^G = \begin{cases} v_{ij}^G, & \text{if } (\text{rand}(0, 1) < \text{CR or } j = j_{\text{rand}}), \\ x_{ij}^G, & \text{otherwise,} \end{cases} \quad (10)$$

DE/current/1:

$$V_i^G = X_i^G + F \cdot (X_{r1}^G - X_{r2}^G). \quad (7)$$

DE/best/1:

$$V_i^G = X_{\text{best}}^G + F \cdot (X_{r1}^G - X_{r2}^G). \quad (8)$$

DE/current-to-best/1:

$$V_i^G = X_i^G + F \cdot (X_{\text{best}}^G - X_i^G) + F \cdot (X_{r1}^G - X_{r2}^G), \quad (9)$$

where $r1, r2, r3, r4$, and $r5$ are integers obtained in the interval $[1, \text{NP}]$ randomly and $r1 \neq r2 \neq r3 \neq r4 \neq r5 \neq i$. The parameter F is the scaling factor, which is used to control the scale of variation, usually $F \in (0, 1]$. X_{best}^G is the most suitable individual in the current population.

2.2.3. Crossover. After mutation, the target vector of each generation crosses with its corresponding mutation vector to generate an experimental vector $U_i^G = (u_{i1}^G, u_{i2}^G, \dots, u_{iD}^G)$. Each dimension of the experimental vector is gained from the mutation vector V_i^G or the target vector X_i^G . The specific operation of crossover looks as follows:

where $i = 1, 2, \dots, \text{NP}$, $j = 1, 2, \dots, D$, crossover probability $\text{CR} \in [0, 1]$. CR is used to control the probability that the experimental vector inherits the mutation vector. The parameter $j_{\text{rand}} \in [1, D]$, which is used to make sure that one or more dimensions in the experimental vector, inherits the value of the target vector.

To prevent values from exceeding the upper and lower bounds, they must also be bounded. The specific operation is as follows:

$$u_{ij}^G = \begin{cases} x_{uj}^G, & \text{if } u_{ij}^G > x_{uj}^G, \\ x_{lj}^G, & \text{if } u_{ij}^G < x_{lj}^G, \\ u_{uj}^G, & \text{otherwise,} \end{cases} \quad (11)$$

where $i = 1, 2, \dots, \text{NP}$, $j = 1, 2, \dots, D$.

2.2.4. Selection. Use the greedy strategy between the target vector and the experimental vector: select the best function value to enter the next iteration. Better individuals can be inherited to the following generation via the greedy method, making sure the algorithm convergence. The specific operation process is described as follows:

FIGURE 3: Schematic diagram of logistic chaotic mapping.

$$X_i^G = \begin{cases} U_i^G, & \text{if } f(U_i^G) \leq f(X_i^G), \\ X_i^G, & \text{otherwise,} \end{cases} \quad (12)$$

where $i = 1, 2, \dots, NP$ and $f(x)$ is the representatives of the function value of the vector x .

3. The Parameters of CWDEPSO

This section describes the improvements of CWDEPSO parameters. We introduce chaos theory, the improvement of parameters in PSO, and that of parameters in DE, in turn.

3.1. Chaos Theory. Chaotic motion is a common nonlinear phenomenon in nature. It looks cluttered on the outside but has subtle regularity on the inside. It possesses the features of regularity, randomness, as well as ergodicity. Chaotic ergodicity is able to experience every state within a definite interval with no repetition. Therefore, the optimized search using chaotic variables is superior to the blind random search. There are many types of chaotic dynamic models. In this paper, logistic mapping [57, 58] is used to design chaotic operations. The iterative equation is defined as follows:

$$y^{k+1} = \mu y^k \cdot (1 - y^k), \quad (13)$$

where $y^0 = \text{rand}(0, 1)$ and $y^0 \neq \{0.25, 0.5, 0.75\}$. It is completely chaotic when the control parameter $3.57 < \mu < 4$. The best results are obtained when $\mu = 4$, as shown in Figure 3, so let $\mu = 4$ in this article.

3.2. Improvement of PSO Parameters. The traditional PSO algorithm uses fixed parameter settings during the evolution process. However, many pieces of literatures indicate that PSO is very sensitive to parameter settings [30, 58–61]. Among them, Zhan et al. [61] proposed APSO, which is a very famous PSO variant that well deals with the parameters. Also, Peng et al. [60] used probabilistic methods to prove that the inertia weight on PSO's convergence is an order of magnitude higher than two learning factors, so the inertia weight can be more necessary to be adjusted in the evolution process of PSO. Among them, for the inertia

coefficient ω , Shi and Eberhart [56] proposed a linearly decreasing ω . The reason for this improvement is to increase the global search capability in the early iterations and the local search capability in the late iterations. Correspondingly, a larger inertial weight is helpful for global exploration, and with small inertia weight, it is conducive to local exploration. Afterwards, after many numerous studies [53, 58, 59], it was found that the nonlinear ω is more helpful for algorithm enhancement. In addition, in this paper, we choose to use nonlinear ω and perform chaos treatment, aiming to enhance the disorder during iteration to enhance the population diversity of the algorithm in the late iteration. In general, we hope that the global exploration competence of the algorithm is strong in the primary period, and the local search competence is improved later to explore the optimal value more accurately. In this paper, we use the arctangent acceleration coefficient (AT) to propose a nonlinearly changing inertial weight and introduce a chaotic mechanism within to avoid a local optimum by increasing the population diversity and thereby improving the search performance of PSO. The update of the inertia weight is presented as follows: where φ , τ , and q are constants ($\varphi = 0.9$, $\tau = 0.65$, $q = 0.05$). $q \cdot \alpha \cdot y^k$ is the chaos term, which enables the algorithm to increase chaos characteristics while maintaining the original change trend, and α is the direction factor. The introduction of the direction factor can make ω fluctuate up and down, and in each iteration, the value of α will choose to be equal to 1 or -1 . In each generation, y^k is updated by equation (13) and substituted into equation (14). The specific image is shown in Figures 4(a) and 4(b).

$$\omega = \frac{\varphi}{\pi} \cdot \arctan\left(2 \cdot \left(\frac{G-1}{G_{\max}}\right) - 1\right) + \tau + q \cdot \alpha \cdot y^k, \quad (14)$$

Similarly, the learning factor also influences the performance of PSO greatly. c_1 and c_2 are called cognitive and social components, respectively. Within the traditional PSO algorithm, the cognitive component is always equal to the social component, that is, $c_1 = c_2$. However, many literatures [30, 53, 59] have demonstrated that the focus should be on the global exploration ability at the beginning of the iteration; i.e., the social component c_1 is larger, and the focus

FIGURE 4: Improved inertia weight: (a) without adding chaotic terms; (b) with adding chaotic terms.

FIGURE 5: Improved learning factors.

should be on the local exploration ability at the later part of the iteration; i.e., the cognitive component c_2 is larger. In order to better balance the global exploration and local exploration ability in the iteration, in this paper, we use the sine function to optimize it so that c_1 gradually decreases and c_2 gradually increases. The specific equation is as follows: where ∂ , δ_1 , and δ_2 are constants ($\partial = 2$, $\delta_1 = 2.5$, and $\delta_2 = 0.5$).

The change in curves of c_1 and c_2 is presented in Figure 5.

$$c_1 = -\partial \cdot \sin\left(\frac{2 \cdot G}{\pi \cdot G_{\max}}\right) + \delta_1, \quad (15)$$

$$c_2 = \partial \cdot \sin\left(\frac{2 \cdot G}{\pi \cdot G_{\max}}\right) + \delta_2, \quad (16)$$

3.3. Improvement of DE Parameters. Similar to PSO, the parameters of DE also have a significant impact on the algorithm [21, 24, 40, 41]. The JADE first proposed the use of Cauchy and Gaussian distributions to update parameters and was widely cited in the literature [36, 62, 63], where the parameters μF and μCR are proposed to be updated by a power function in the literature [62]. Inspired by that, this paper adjusts the scaling factor and the crossover probability by using the parameter control methods below.

Similar to JADE, in each generation of evolution, the scaling factors of all the individuals in the population are independently and randomly created by the Cauchy distribution, as follows:

$$F_i = \text{Cauchy}(\mu F, 0.1), \quad (17)$$

where $\text{Cauchy}(\mu F, 0.1)$ is a random number generated by the Cauchy distribution with a scaling factor of 0.1 and a position parameter of μF . If $F_i > 1$, make it equal to 1, and if $F_i < 0$, regenerate. In the current generation, all successful scaling factors F_i are stored in set S_F . The position parameter μF is initialized to 0.5. At the end of all the generations, μF is updated as follows:

$$\mu F = (1 - c) \cdot \mu F + c \cdot \text{mean}_L(S_F), \quad (18)$$

where c is a constant between the interval $[0, 1]$ and mean_L represents the Lehmer average:

$$\text{mean}_L(S_F) = \frac{\sum_{F_i \in S_F} F_i^2}{\sum_{F_i \in S_F} F_i}. \quad (19)$$

In each evolutionary generation, the crossover probability CR of all the individuals within the population is generated independently on the basis of the Gaussian distribution parameters as follows:

$$CR_i = \text{Gaussian}(\mu CR, 0.1), \quad (20)$$

where Gaussian(μCR , 0.1) is a random number generated by the Gaussian distribution with a standard deviation of 0.1 and a mean value of μCR . If $\text{CR}_i < 0$, let it be equal to 1, and if $\text{CR}_i > 0$, let it be equal to 0. In the current generation, all successful crossover probabilities CR_i are stored in the set S_{CR} . The position parameter μCR is initialized to 0.5. Unlike JADE, after the end of each generation, the update of μCR adopts the power average, as follows:

$$\mu\text{CR} = (1 - c) \cdot \mu\text{CR} + c \cdot \text{mean}_{\text{pow}}(S_{\text{CR}}), \quad (21)$$

where mean_{pow} represents the power-mean:

$$\text{mean}_{\text{pow}}(S_{\text{CR}}) = \left(\frac{\sum_{\text{CR}_i \in S_{\text{CR}}} (\text{CR}_i)^n}{|S_{\text{CR}}|} \right)^{(1/n)}, \quad (22)$$

where $|S_{\text{CR}}|$ represents the cardinality of the set S_{CR} and n is a constant ($n = 2$).

4. Dynamic Probabilistic Mutation PSO

Under the premise of ensuring population diversity, PSO can effectively explore the solution area and has a fast convergence speed. However, the search can easily stall due to the lack of diversity during the search procedure. Once trapped in premature convergence, it is difficult to escape from local optimization [64–67]. To solve this problem, we have to enhance the population diversity in the late iteration in order to avoid falling into local optimum. According to [68], in the paper, we learn that DE effectively helps PSO jump out of the local optimum and improve the population diversity when the PSO algorithm falls into local optimum in the late iteration, and the literature [48, 68, 69] all shows that the DE algorithm combined with the PSO algorithm can effectively improve the algorithm performance and avoid premature convergence. In addition, some algorithms such as DEPSO [48] substantially increase the computational complexity in each iteration by comparing two different trial vectors when adapting. Therefore, in this paper, we choose to help the algorithm improve its diversity and at the same time avoid adding too much computational complexity by observing the results of the previous iterations to predict whether to join the DE algorithm next time. This paper proposes a novel dynamic probabilistic mutation PSO to improve local search in the later iteration. As shown in Figure 6, the improved mutation operator is introduced in PSO for increasing the population diversity and helping the algorithm get away from local convergence. The specific improvements can be seen below.

4.1. Monitor Global Optimum $g\text{best}^G$ Update. First, the global optimum $g\text{best}^G$ of each generation is monitored. Once the i -th particle of G -th generation fails to update the global optimum, the flag bit γ_i corresponding to the particle is increased by one. When the flag bit γ_i of all particles is greater than one, that is, all particles of the G -th generation fail to search the optimal value, it is considered to be trapped within a local optimum, which might affect the effect of population convergence. We will use methods increasing the

probability of mutation according to the change in the flag bit γ_i to aid the population jump out of local convergence.

4.2. Improvement of Mutation Operator. In addition, an appropriate mutation operator needs to be selected. The DE/current-to-best/1 mutation operator possesses great local development capabilities and fast convergence speed, which can reflect the current particle state, but in comparison, the global development ability is poor. The DE/rand/1 and DE/rand/2 mutation operators depend on other random individuals, which are conducive to improving the population diversity. The global convergence competence is strong, but the convergence speed is relatively slow. Even if PSO updates the global optimum $g\text{best}^G$ unsuccessfully in the current population, the corresponding current particles still have a high development value. Considering that the algorithm is mostly at the end of evolution when it falls into local convergence, it should pay more attention to the local development ability; we thus take the DE/current-to-best/1 mutation operator as the basis and improve it. The new mutation operator DE/current-to- $g\text{best}$ /1 is as follows:

$$V_i^G = \alpha \cdot X_i^G + F \cdot (g\text{best}^G - X_i^G) + K \cdot (X_{r1}^G - X_{r2}^G), \quad (23)$$

where K is a variable independent of the scaling factor and α is a variable that changes with the number of iterations. Making the variable α gradually increase with the number of iterations can reduce the influence of local convergence of the PSO in the later stage and improve the ability to explore the surrounding area. Mathematically, the variable α is given as follows:

$$\alpha = 0.8 + 0.2 \cdot \left(\frac{G}{G_{\text{max}}} \right). \quad (24)$$

In contrast, the population diversity decreases as the number of iterations increases, and the variable K affecting the population diversity should increase as the iterations increase. Otherwise, it will still fall into local convergence. K is defined as follows:

$$K = 0.1 + 0.2 \cdot \left(\frac{G}{G_{\text{max}}} \right). \quad (25)$$

By strengthening the exploration of the global situation in the early period and focusing on the exploration of the current local area in the later stage, the balance between local and global convergences is effectively maintained.

4.3. Processing of Monitoring Data. When all particle flags $\gamma_i > 1$, the count variable count increases by one. With the increase in count, it indicates that the number of convergence failures gradually increases, which means that the algorithm is more likely to fall into the state of local convergence. The initial increase in the variable count has contingency, but the more the count increases, the more likely it falls into premature convergence. Therefore, we define a probability p , which represents the probability of the algorithm quoting mutation ideas. The larger the count, the larger the probability p . Inspired by the exponential

FIGURE 6: Schematic diagram of CWDEPSO. ★ represents the optimal value.

formula update probability in the literature [49], the paper uses an exponential formula to control the probability p of introducing mutation ideas. The probability p is set as follows:

$$p = e^{-(G_{\max} - 10 \cdot \text{count} / G_{\max})}, \quad (26)$$

where $G_{\max}$ is the evolutions' maximum number.

If the flag bit γ_i corresponding to the i -th particle is greater than the warning value θ , it means that the individual has a high probability of falling into local convergence that is not easy to escape. As a result, the reference value of the particle would be low. This paper sets the warning value $\theta = (G_{\max}/5)$ in this article and gives it a proper probability to reinitialize it. The specific operation is as follows:

$$x_{ij} = \begin{cases} x_{lj} + \text{rand}(0, 1) \times (x_{uj} - x_{lj}), & \text{if } \text{rand}(0, 1) < 0.01, \\ x_{ij}, & \text{otherwise,} \end{cases} \quad (27)$$

where $j = 1, 2, \dots, D$.

The complete monitoring mechanism is shown in Figure 7.

4.4. Modification of Selection Method. After the mutation operation, the algorithm will proceed according to the crossover and selection operation of DE. For the select operation, the fundamental DE and a lot of DE variants adopt the greedy selection method, while greedy selection often leads to the abandonment of many good individuals falling into local convergence [63]. In order to fully mine the effective information of these excellent individuals, we have added a temporary population to save these abandoned individuals temporarily as follows:

$$\text{temp}_i^G = \begin{cases} U_i^G, & \text{if } f(V_i^G) < f(U_i^G), \\ V_i^G, & \text{otherwise,} \end{cases} \quad (28)$$

where temp_i^G holds all the individuals who have been eliminated in the contemporary era. Some of the outstanding individuals in the temp_i^G have a higher probability

than the worst individuals of the current updated population. If the worst individual within the brand-new population is worse than the best individual within the temporary population, it is replaced. It is worth noting that if too many individuals are replaced, it will cause local convergence.

On the basis of the analysis before, the pseudocode of CWDEPSO can be seen below (Algorithm 1).

5. Experimental Results and Discussion

In this section, the performance of CWDEPSO is evaluated by 24 benchmark functions that have been adopted widely in the field of numerical optimization methods [58, 70, 71], most of which belong to CEC2017 [72, 73], as shown in Tables 1–3. The experiment can be classified into two parts. The first part compares CWDEPSO with PSO, DE, and their variants AIWCPSO, MPSPSO-ST, JADE, SinDE, and DEPSO. The second part compares CWDEPSO with several other well-known optimization algorithms (GSA, ABC, SCA, MFO, and BBO). All of the algorithms are coded in MATLAB R2018a and run on a PC with an Intel(R) Core(TM) i7-7700HQ CPU@2.80 GHz/8GB RAM.

5.1. Test Function. The benchmark functions adopted within the article are divided into three categories, and $f_1 \sim f_5$ are unimodal functions, where f_5 contains noise interference; $f_6 \sim f_{22}$ are multimodal function. Among them, the local optimum of f_7 is large, and the suboptimal value differs significantly from the global optimum. The characteristic of f_8 is that it has a very narrow global basin, so it can be rather easy to fall into a local optimum. $f_{23} \sim f_{25}$ are multimodal benchmark functions with fixed dimensions. The three types are listed in Tables 1–3. Dim and Scope, respectively, represent the solution space dimension and range of the solution, and type represents the type of the current function.

5.2. Comparison of CWDEPSO with PSO, DE, JADE, AIWCPSO, MPSPSO-ST, JADE, SinDE, and DEPSO.

FIGURE 7: Process of monitoring mechanism.

CWDEPSO is compared with PSO, DE, and their variants AIWCPSO, MPSPSO-ST, JADE, SinDE, and DEPSO in this part. To ensure a fair comparison, we set the parameters in both sets of experiments all according to the original paper, strictly following the steps in the literature [58, 59], and the population size is set to 500. It is worth noting that the algorithm CWDEPSO proposed in this paper has the same number of function evaluations as PSO and DE in each iteration, so it can ensure fairness in the case of the same number of iterations. Also, the initial populations of all algorithms are generated with random numbers, and each group is run twenty times independently. The specific parameters are set as follows:

PSO setting: $c_1 = c_2 = 2$, $\omega = 0.9 \sim 0.4$, $V_{\max} = 0.2 \times \text{Bound}$.

AIWCPSO setting: $c_1 = c_2 = 2$, $\omega = 0.9 \sim 0.4$, $V_{\max} = 0.2 \times \text{Bound}$.

MPSPSO-ST setting: $\partial = 0.2$, $\theta = 1.5$, $\phi = 0.9$, $\tau = 0$, $\rho = 0.1$, $V_{\max} = 0.2 \times \text{Bound}$

DE setting: $F = 0.5$, $CR = 0.5$.

JADE setting: $p = 0.05$, $c = 0.1$.

SinDE settings: $\text{freq} = 0.25$.

DEPSO settings: $k = 4$.

CWDEPSO setting: $c_1: 2.5 \sim 0.5$, $c_2: 0.5 \sim 2.5$, $\omega =$

```

(1) Step 1: initialization
(2) Parameter initialization: NP, D, Gmax, G = 1, μF = μCR = 0.5, γ = 0
(3) Generate initial population: pop0
(4) Find the initial pbest and gbest of the population
(5) Step 2: update parameters
(6) While iter < itermax
(7)   Update the inertia weights ω adopting equation (14)
(8)   Update the social component c2 and the cognitive component c1 using equations (15) and (16)
(9)   Calculate Fi and CRi of each particle using equations (17) and (20)
(10)  Update the probability P using equation (26)
(11) Step 3: finding the optimal value through iteration
(12) For i = 1: NP do
(13)   Update the velocity Vi adopting equation (1)
(14)   Using equation (2) to update the position of the particle Xi
(15) End For
(16) If rand(0, 1) < P
(17)   For i = 1: NP do
(18)     Mutation operation using equation (23)
(19)     Crossover and boundary processing using equations (10) and (11), respectively
(20)     Using equations (12) and (28) to get Xi and tempi
(21)     Update parameters μFi and μCRi using equations (18) and (21), respectively
(22)   End For
(23)     Select the better one between the worst of tempi and the best of Xi
(24)     Update pbest and gbest
(25) Else
(26)   Calculate the fitness of particles in the population and update pbest and gbest
(27) End If
(28) If failed to update gbestG
(29)   flag bit γi = γi + 1
(30) Else
(31)   γi = 0
(32) End If
(33) If flag bit γi > warning value θ
(34)   Choose whether to initialize the particle using equation (27)
(35) End If
(36) If flag bits of all particles γ > 0
(37)   number of failures count = count + 1;
(38) End If
(39) G = G + 1
(40) Step 4: output result
(41) End While

```

ALGORITHM 1: Pseudocode of the CWDEPSO algorithm.

TABLE 1: Unimodal benchmark functions.

Function	Scope	Dim	Name
$f_1(x) = \sum_{i=1}^D x_i^2$	$[-100, 100]$	30	Sphere
$f_2(x) = x_1^2 + 10^6 \sum_{i=2}^D x_i^2$	$[-10, 10]$	30	Bent Cigar
$f_3(x) = \sum_{i=1}^D x_i ^{i+1}$	$[-100, 100]$	30	Sum of different powers
$f_4(x) = \sum_{i=1}^D x_i^2 + (\sum_{i=1}^D 0.5x_i)^2 + (\sum_{i=1}^D 0.5x_i)^4$	$[-5, 10]$	30	Zakharov
$f_5(x) = \sum_{i=1}^D ix_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]$	30	Noise

0.924 ~ 0.375, $c = 0.1$, $V_{\max} = 0.2 \times \text{Bound}$.

The benchmark functions adopted in this section have been presented within Tables 1–3. For the purpose of reducing statistical mistakes, all the test functions are

independently evaluated 20 times. The obtained best value (Best), the worst value (Worst), the mean results (Mean), and the standard deviation (S.D.) by statistics are presented in Table 4, with the optimal values identified in bold. For the

TABLE 2: Multimodal benchmark functions.

Function	Scope	Dim	Name
$f_6(x) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	$[-10, 10]$	30	Rosenbrock
$f_7(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10)$	$[100, 100]$	30	Rastrigin
$f_8(x) = -20 \exp(-0.2\sqrt{(1/n) \sum_{i=1}^D x_i^2}) - \exp((1/n) \sum_{i=1}^D \cos(2\pi x_i)) + 20 + e$	$[-32, 30]$	30	Ackley
$f_9(x) = \sin^2(\pi w_1) + \sum_{i=1}^{D-1} (w_i - 1)^2 [1 + 10 \sin^2(\pi w_i + 1)] + (w_D - 1)^2 [1 + \sin^2(2\pi w_D)]$ $w_i = 1 + ((x_i - 1)/4)$	$[-10, 10]$	30	Levy
$f_{10}(x) = \sum_{i=1}^D (10^6)^{(i-1/D-1)} x_i^2$	$[-10, 10]$	30	High conditioned elliptic
$f_{11}(x) = 10^6 x_i^2 + \sum_{i=2}^D x_i^2$	$[-10, 10]$	30	Discus
$f_{12}(x) = (\sum_{i=1}^D x_i^2)^2 - (\sum_{i=1}^D x_i)^2 ^{(1/2)} + (0.5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i)/D + 0.5$	$[-10, 10]$	30	HGBat
$f_{13}(x) = (1/4000) \sum_{i=1}^D x_i^2 - \prod_{i=1}^D \cos(x_i/\sqrt{i}) + 1$	$[-600, 600]$	30	Griewank
$f_{14}(x) = \sum_{i=1}^D (\sum_{k=1}^{k_{\max}} [a^k \cos(x_i + 0.5)]) - D \sum_{k=1}^{k_{\max}} [a^k \cos(2\pi b^k \cdot 0.5)]$ $a = 0.25, b = 3, k_{\max} = 20$	$[-10, 10]$	30	Weierstrass
$f_{15}(x) = (10/D^2) \prod_{i=1}^D (1 + i \sum_{j=1}^{32} (2^j x_i - \text{round}(2^j x_i) /2^j))^{(10/D^{1.2})} - (10/D^2)$	$[-10, 10]$	30	Katsuura
$f_{16}(x) = \sum_{i=1}^D x_i^2 - D ^{(1/4)} + (0.5 \sum_{i=1}^D x_i^2 + \sum_{i=1}^D x_i)/D + 0.5$	$[-10, 10]$	30	HappyCat
$f_{17} = [(1/D - 1) \sum_{i=1}^{D-1} (\sqrt{s_i} \cdot (\sin(50.0)s_i^{0.2} + 1))]^2$ $s_i = \sqrt{x_i^2 + x_{i+1}^2}$	$[-10, 10]$	30	Schaffer's F7
$f_{18}(x) = \sum_{i=1}^D (x_i^2 - 10 \cos(2\pi x_i) + 10) + f_{13}^*$	$[100, 100]$	30	Noncontinuous rotated Rastrigin
$f_{19}(x) = (\pi/D) \left\{ 10 \sin^2(\pi y_1) + \sum_{i=1}^{D-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_D - 1)^2 \right\} + \sum_{i=1}^n u(x_i, 10, 100, 4), y_i = 1 + (x_i + 1/4)$	$[-50, 50]$	30	Penalized 1
$u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a, \\ k(-x_i - a)^m, & x_i < -a, \end{cases}$ $f_{20}(x) = (1/10) \sin^2(3\pi x_1) + \sum_{i=1}^{D-1} (x_i - 1)^2 + \sin^2(3\pi x_{i+1}) + (x_D - 1)^2 (1 + \sin^2(2\pi x_D)) + \sum_{i=1}^n u(x_i, 5, 100, 4)$	$[-50, 50]$	30	Penalized 2
$f_{21}(x) = \sum_{i=1}^D x_i \sin(x_i) + 0.1 x_i $	$[-10, 10]$	30	Alpine

purpose of facilitating the comparison of the mean and the S.D., we represent the data with a radar chart, as shown in Figure 8. Figure 9 represents the trajectory of the average of the algorithm on the current test function.

As shown in Table 4, the single-peaked functions $f_1 \sim f_5$ are significantly more accurate for the optimization of CWDEPSO compared with the other seven algorithms, achieving the best results in each index. Among the

TABLE 3: Multimodal benchmark functions with fixed dimensions.

Function	Scope	Dim	Name
$f_{22}(x) = \sum_{i=1}^{11} [a_i - (x_1(b_i^2 + b_i x_2)/b_i^2 + b_i x_3 + x_4)]^2$	$[-5, 5]$	4	Kowalik
$f_{23}(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2)$	$[0, 1]$	6	Hartman 6
$f_{24}(x) = \sum_{i=1}^{10} [(X - a_i)(X - a_i)^T + c_i]^{-1}$	$[0, 10]$	4	Shekel_10

TABLE 4: Experiments were carried out on 24 classical test functions with CWDEPSO, PSO, DE, AIWCPSO, MPSPSO-ST, JADE, SinDE, and DEPSO.

f	Criteria	CWDEPSO	PSO	DE	AIWCPSO	MPSPSO-ST	JADE	SinDE	DEPSO
f_1	Best	7.59E-28	3.04E-01	7.13E-04	5.10E-06	9.63E-15	2.85E-13	1.18E-03	6.92E-10
	Worst	3.98E-24	1.42E+00	2.27E-03	7.84E-05	1.64E-06	5.21E-12	3.49E-03	1.45E-09
	Mean	7.88E-25	7.12E-01	1.33E-03	3.37E-05	1.65E-07	2.23E-12	2.42E-03	1.11E-09
	S.D.	4.03E-24	1.40E+00	1.92E-03	8.74E-05	1.56E-06	4.75E-12	2.42E-03	9.27E-10
f_2	Best	1.69E-24	1.27E+03	3.46E+00	1.64E+00	2.93E-09	1.11E-09	7.58E+00	5.09E-04
	Worst	3.48E-20	7.25E+03	1.54E+01	1.33E+02	2.02E+02	9.61E-08	1.87E+01	1.21E-03
	Mean	4.93E-21	4.11E+03	9.12E+00	2.04E+01	2.02E+01	4.42E-08	1.24E+01	8.38E-04
	S.D.	3.33E-20	5.15E+03	1.20E+01	1.20E+02	1.92E+02	1.01E-07	1.13E+01	5.89E-04
f_3	Best	3.39E-72	1.20E+07	3.61E+00	1.91E-04	8.90E-24	4.42E-19	2.37E-01	1.33E-16
	Worst	3.26E-63	3.11E+11	2.87E+10	1.83E+01	8.22E-17	1.43E-14	3.01E+26	1.53E-15
	Mean	3.37E-64	3.21E+10	2.87E+09	2.04E+00	8.31E-18	3.15E-15	3.01E+25	7.05E-16
	S.D.	3.08E-63	2.94E+11	2.73E+10	1.73E+01	7.79E-17	1.53E-14	2.85E+26	1.47E-15
f_4	Best	4.13E-18	8.03E+00	2.01E+02	4.03E+01	1.82E+01	1.12E-01	1.25E+02	1.15E-08
	Worst	3.93E-15	4.37E+01	3.08E+02	1.30E+02	3.82E+02	1.60E+01	2.59E+02	4.20E-08
	Mean	4.40E-16	2.31E+01	2.63E+02	6.78E+01	2.04E+02	4.10E+00	1.93E+02	2.50E-08
	S.D.	3.68E-15	3.20E+01	1.05E+02	8.17E+01	3.07E+02	1.74E+01	1.17E+02	2.63E-08
f_5	Best	1.26E-03	3.39E-02	1.80E-01	4.25E-02	2.50E-02	2.08E-01	1.75E-01	7.76E-03
	Worst	2.26E-03	7.37E-02	5.95E-01	1.22E-01	8.35E-02	4.99E-01	6.87E-01	2.24E-02
	Mean	1.74E-03	5.45E-02	3.79E-01	7.76E-02	4.74E-02	3.50E-01	3.62E-01	1.32E-02
	S.D.	1.12E-03	4.55E-02	4.23E-01	8.31E-02	5.41E-02	2.88E-01	4.54E-01	1.56E-02
f_6	Best	5.45E-01	2.91E+01	2.71E+01	2.06E+01	2.76E+01	6.92E+00	2.65E+01	5.94E-04
	Worst	2.55E+01	1.44E+02	6.69E+01	1.03E+02	1.05E+02	7.70E+01	9.40E+01	1.35E-03
	Mean	1.93E+01	6.59E+01	3.34E+01	6.45E+01	7.04E+01	2.63E+01	3.45E+01	7.97E-04
	S.D.	2.97E+01	1.16E+02	3.62E+01	8.90E+01	9.01E+01	5.61E+01	6.28E+01	6.40E-04
f_7	Best	5.33E-15	4.10E+01	1.28E+02	3.78E+01	5.67E+01	1.71E+01	9.92E+01	9.95E-01
	Worst	2.98E+01	6.62E+01	1.60E+02	7.16E+01	8.76E+01	3.09E+01	1.23E+02	3.38E+01
	Mean	3.58E+00	5.06E+01	1.52E+02	4.94E+01	7.34E+01	2.29E+01	1.14E+02	1.39E+01
	S.D.	2.81E+01	2.84E+01	2.78E+01	3.29E+01	3.15E+01	1.44E+01	2.43E+01	3.11E+01
f_8	Best	7.55E-14	6.76E-02	6.65E-03	7.19E-04	9.31E-01	7.80E-08	1.14E-02	2.03E-05
	Worst	3.63E-12	2.15E+00	1.96E-02	2.74E+00	6.35E+00	9.31E-01	2.54E-02	2.72E-05
	Mean	6.84E-13	1.24E+00	1.12E-02	3.94E-01	3.31E+00	9.31E-02	1.59E-02	2.28E-05
	S.D.	3.60E-12	2.17E+00	1.27E-02	2.70E+00	4.85E+00	8.84E-01	1.15E-02	7.19E-06
f_9	Best	1.06E-07	5.77E-03	1.31E-04	2.07E-03	3.36E+00	2.27E-14	1.10E-04	3.29E-10
	Worst	4.48E-01	7.61E-01	1.24E-03	4.58E+00	2.01E+01	4.54E-01	1.53E-03	1.57E-09
	Mean	1.16E-01	1.62E-01	3.45E-04	2.41E+00	8.14E+00	8.12E-02	4.50E-04	8.25E-10
	S.D.	4.57E-01	7.53E-01	9.90E-04	5.26E+00	1.45E+01	4.15E-01	1.24E-03	1.03E-09
f_{10}	Best	7.93E-27	2.50E+00	8.35E-03	2.90E-03	3.25E-10	3.76E-11	7.74E-03	2.00E-05
	Worst	3.04E-21	2.34E+01	2.90E-02	6.66E-02	7.57E+01	3.85E-07	2.29E-02	6.03E-05
	Mean	3.90E-22	1.09E+01	1.59E-02	3.16E-02	8.90E+00	3.88E-08	1.56E-02	4.01E-05
	S.D.	2.85E-21	1.94E+01	2.11E-02	6.61E-02	7.09E+01	3.65E-07	1.41E-02	3.89E-05
f_{11}	Best	4.52E-28	1.26E-03	1.15E-05	1.04E-05	2.04E-14	2.16E-14	1.81E-05	6.28E-07
	Worst	3.87E-24	2.79E-02	1.04E-04	2.71E-04	3.40E-06	7.56E-13	1.08E-04	9.80E-07
	Mean	4.78E-25	1.13E-02	2.93E-05	6.34E-05	5.20E-07	1.65E-13	4.73E-05	8.27E-07
	S.D.	3.62E-24	2.75E-02	8.45E-05	2.29E-04	3.48E-06	6.75E-13	8.58E-05	3.83E-07

TABLE 4: Continued.

f	Criteria	CWDEPSO	PSO	DE	AIWCPSO	MPSPSO-ST	JADE	SinDE	DEPSO
f_{12}	Best	$3.85E-01$	$2.11E-01$	$2.65E-01$	$3.12E-01$	$2.60E-01$	$3.12E-01$	$3.14E-01$	$5.82E-02$
	Worst	$6.90E-01$	$1.04E+00$	$9.84E-01$	$1.13E+00$	$1.06E+00$	$7.60E-01$	$6.65E-01$	$1.46E-01$
	Mean	$5.33E-01$	$5.42E-01$	$4.47E-01$	$6.81E-01$	$6.48E-01$	$4.57E-01$	$4.03E-01$	$8.68E-02$
	S.D.	$3.87E-01$	$1.00E+0$	$0.661E-01$	$1.06E+00$	$9.56E-01$	$5.38E-01$	$2.91E-01$	$8.62E-02$
f_{13}	Best	$0.00E+00$	$5.30E-01$	$1.50E-03$	$1.71E-05$	$1.89E-13$	$2.42E-13$	$1.92E-03$	$7.49E-11$
	Worst	$2.95E-02$	$8.95E-01$	$1.20E-01$	$1.43E-02$	$9.34E-02$	$1.49E-02$	$5.72E-02$	$1.97E-02$
	Mean	$1.01E-02$	$7.05E-01$	$2.72E-02$	$5.24E-03$	$3.60E-02$	$2.23E-03$	$1.13E-02$	$1.03E-02$
	S.D.	$3.55E-02$	$3.78E-01$	$1.42E-01$	$1.19E-02$	$1.14E-01$	$1.49E-02$	$5.00E-02$	$2.05E-02$
f_{14}	Best	$-8.70E+02$	$-8.02E+02$	$-8.70E+02$	$-8.69E+02$	$-8.68E+02$	$-7.21E+02$	$-8.70E+02$	$-8.70E+02$
	Worst	$-8.49E+02$	$-7.72E+02$	$-8.70E+02$	$-8.43E+02$	$-8.34E+02$	$-6.87E+02$	$-8.70E+02$	$-8.70E+02$
	Mean	$-8.66E+02$	$-7.87E+02$	$-8.70E+02$	$-8.62E+02$	$-8.52E+02$	$-7.08E+02$	$-8.70E+02$	$-8.70E+02$
	S.D.	$2.21E+01$	$2.18E+01$	$0.00E+00$	$2.37E+01$	$4.04E+01$	$2.91E+01$	$0.00E+00$	$0.00E+00$
f_{15}	Best	$0.00E+00$	$4.30E-11$	$0.00E+00$	$1.98E-11$	$1.40E-11$	$5.49E-12$	$0.00E+00$	$0.00E+00$
	Worst	$5.87E-13$	$5.13E-11$	$0.00E+00$	$5.15E-11$	$4.20E-11$	$8.24E-12$	$0.00E+00$	$4.95E-11$
	Mean	$9.69E-14$	$4.71E-11$	$0.00E+00$	$4.64E-11$	$2.16E-11$	$7.15E-12$	$0.00E+00$	$2.47E-11$
	S.D.	$5.43E-13$	$5.88E-12$	$0.00E+00$	$2.86E-11$	$2.56E-11$	$2.48E-12$	$0.00E+00$	$3.50E-11$
f_{16}	Best	$1.31E-01$	$2.83E-01$	$4.74E-01$	$2.69E-01$	$2.17E-01$	$2.83E-01$	$4.42E-01$	$1.46E-01$
	Worst	$5.26E-01$	$5.68E-01$	$6.24E-01$	$6.56E-01$	$4.44E-01$	$4.69E-01$	$6.00E-01$	$1.87E-01$
	Mean	$3.41E-01$	$4.26E-01$	$5.69E-01$	$4.32E-01$	$3.26E-01$	$3.47E-01$	$5.38E-01$	$3.66E-01$
	S.D.	$3.88E-01$	$2.02E-01$	$1.56E-01$	$3.76E-01$	$2.65E-01$	$2.00E-01$	$1.33E-01$	$2.88E-02$
f_{17}	Best	$0.00E+00$	$2.96E-01$	$0.00E+00$	$1.12E-02$	$3.21E-01$	$2.31E-01$	$0.00E+00$	$4.08E-02$
	Worst	$0.00E+00$	$3.16E-01$	$1.67E-08$	$7.92E-02$	$5.76E-01$	$2.84E-01$	$7.46E-08$	$1.96E-01$
	Mean	$0.00E+00$	$3.06E-01$	$3.55E-09$	$3.06E-02$	$4.22E-01$	$2.55E-01$	$7.67E-09$	$1.17E-01$
	S.D.	$0.00E+00$	$1.44E-02$	$1.59E-08$	$7.14E-02$	$2.26E-01$	$5.51E-02$	$7.06E-08$	$1.10E-01$
f_{18}	Best	$0.00E+00$	$4.24E+01$	$9.99E+01$	$2.50E+01$	$1.60E+01$	$2.26E+01$	$6.94E+01$	$6.71E-05$
	Worst	$7.81E+00$	$7.96E+01$	$1.30E+02$	$6.31E+01$	$7.00E+01$	$2.81E+01$	$8.65E+01$	$3.59E-04$
	Mean	$8.48E-01$	$6.10E+01$	$1.13E+02$	$4.71E+01$	$3.41E+01$	$2.49E+01$	$7.58E+01$	$2.27E-04$
	S.D.	$7.36E+00$	$2.63E+01$	$2.90E+01$	$3.76E+01$	$4.57E+01$	$4.65E+00$	$1.33E+01$	$2.32E-04$
f_{19}	Best	$1.84E-12$	$2.11E-04$	$1.22E-02$	$1.23E-10$	$4.09E-19$	$6.65E-17$	$7.19E-03$	$5.00E-09$
	Worst	$6.01E-11$	$2.10E-01$	$3.75E-02$	$1.04E-01$	$6.23E-01$	$6.56E-11$	$4.84E-02$	$6.69E-09$
	Mean	$1.81E-11$	$1.05E-01$	$2.19E-02$	$2.07E-02$	$1.87E-01$	$1.58E-11$	$2.28E-02$	$5.82E-09$
	S.D.	$5.37E-11$	$1.48E-01$	$3.05E-02$	$1.31E-01$	$7.01E-01$	$6.22E-11$	$3.74E-02$	$1.20E-09$
f_{20}	Best	$0.00E+00$	$3.49E+01$	$1.18E+02$	$2.38E+01$	$3.70E+01$	$2.04E+01$	$6.89E+01$	$6.71E-05$
	Worst	$3.60E+00$	$9.91E+01$	$1.33E+02$	$6.62E+01$	$8.90E+01$	$2.61E+01$	$9.04E+01$	$3.59E-04$
	Mean	$7.17E-01$	$6.55E+01$	$1.24E+02$	$3.87E+01$	$5.88E+01$	$2.35E+01$	$7.67E+01$	$2.27E-04$
	S.D.	$4.24E+00$	$5.81E+01$	$1.74E+01$	$4.11E+01$	$5.53E+01$	$5.42E+00$	$2.11E+01$	$2.32E-04$
f_{21}	Best	$4.81E-10$	$5.44E-03$	$2.21E-01$	$1.89E-05$	$4.66E-11$	$1.96E-02$	$1.63E-01$	$1.88E-02$
	Worst	$1.12E-05$	$8.17E-03$	$3.49E+00$	$1.86E-04$	$1.72E-02$	$3.12E-02$	$3.07E+00$	$6.25E-02$
	Mean	$2.81E-06$	$6.81E-03$	$1.36E+00$	$4.94E-05$	$1.73E-03$	$2.53E-02$	$1.44E+00$	$4.67E-02$
	S.D.	$1.15E-05$	$1.93E-03$	$3.11E+00$	$1.51E-04$	$1.64E-02$	$1.08E-02$	$3.08E+00$	$3.42E-02$
f_{22}	Best	$3.07E-04$	$3.07E-04$	$3.07E-04$	$3.07E-04$	$3.07E-04$	$3.07E-04$	$3.07E-04$	$3.07E-04$
	Worst	$3.07E-04$	$3.08E-04$	$1.22E-03$	$1.11E-03$	$1.03E-03$	$3.07E-04$	$3.07E-04$	$3.07E-04$
	Mean	$3.07E-04$	$3.08E-04$	$3.99E-04$	$6.77E-04$	$5.89E-04$	$3.07E-04$	$3.07E-04$	$3.07E-04$
	S.D.	$3.15E-18$	$2.29E-07$	$8.69E-04$	$9.82E-04$	$1.09E-03$	$3.07E-19$	$2.34E-11$	$7.71E-10$
f_{23}	Best	$-3.32E+00$	$-3.32E+00$	$-3.32E+00$	$-3.32E+00$	$-3.32E+00$	$-3.32E+00$	$-3.32E+00$	$-3.32E+00$
	Worst	$-3.20E+00$	$-3.20E+00$	$-3.20E+00$	$-3.20E+00$	$-3.20E+00$	$-3.32E+00$	$-3.20E+00$	$3.20E+00$
	Mean	$-3.31E+00$	$-3.30E+00$	$-3.25E+00$	$-3.25E+00$	$-3.29E+00$	$-3.32E+00$	$-3.29E+00$	$-3.26E+00$
	S.D.	$1.13E-01$	$1.50E-01$	$1.73E-01$	$1.84E-01$	$1.72E-01$	$1.40E-15$	$1.72E-01$	$1.88E-01$
f_{24}	Best	$-1.05E+01$	$-1.05E+01$	$-1.05E+01$	$-1.05E+01$	$-1.05E+01$	$-1.05E+01$	$-1.05E+01$	$-1.05E+01$
	Worst	$-1.05E+01$	$-2.43E+00$	$-3.84E+00$	$-5.13E+00$	$-2.43E+00$	$-1.05E+01$	$-1.05E+01$	$-5.18E+00$
	Mean	$-1.05E+01$	$-8.19E+00$	$-9.20E+00$	$-8.92E+00$	$-7.07E+00$	$-1.05E+01$	$-1.05E+01$	$-1.00E+01$
	S.D.	$6.15E-15$	$1.14E+01$	$8.48E+00$	$7.79E+00$	$1.13E+01$	$6.70E-15$	$6.35E-15$	$5.09E+00$

multi-peaked functions, as for f_6 , the best results are obtained by DEPSO, followed by CWDEPSO. For the f_8 functions, CWDEPSO performs well and converges with the best accuracy. For the multimodal function f_9 , the best value

is obtained by JADE, while DEPSO performs best among the worst, mean, and S.D., which obtains more accurate results than others. In the multimodal functions f_{10} and f_{11} , CWDEPSO performs the best. For the f_{12} functions,

FIGURE 8: Continued.

FIGURE 8: Radar chart of eight algorithms, mean and S.D.

FIGURE 9: Continued.

(c)

(d)

(e)

(f)

FIGURE 9: Continued.

(g)

(h)

(i)

(j)

FIGURE 9: Continued.

FIGURE 9: Compare eight algorithms' convergence curves for different benchmark functions: (a) f_2 ; (b) f_3 ; (c) f_4 ; (d) f_5 ; (e) f_6 ; (f) f_7 ; (g) f_8 ; (h) f_9 ; (i) f_{10} ; (j) f_{13} ; (k) f_{17} ; (l) f_{22} .

TABLE 5: The results of Wilcoxon's rank-sum test for the seven algorithms on different test functions.

	PSO	DE	AIWCPSO	MPSPSO-ST	JADE	SinDE	DEPSO
f_1	1	1	1	1	1	1	1
f_2	1	1	1	1	1	1	1
f_3	1	1	1	1	1	1	1
f_4	1	1	1	1	1	1	1
f_5	1	1	1	1	1	1	1
f_6	1	1	1	1	0	1	1
f_7	1	1	1	1	1	1	1
f_8	1	1	1	1	1	1	1
f_9	0	0	1	1	0	0	1
f_{10}	1	1	1	1	1	1	1
f_{11}	1	1	1	1	1	1	1
f_{12}	0	0	0	0	0	1	1
f_{13}	1	0	0	0	0	0	0
f_{14}	1	1	1	1	1	1	0
f_{15}	1	1	1	1	1	1	0
f_{16}	0	1	0	0	0	1	0
f_{17}	1	1	1	1	1	0	0
f_{18}	1	1	1	1	1	1	0
f_{19}	1	1	1	0	1	1	0
f_{20}	1	1	0	1	1	1	0
f_{21}	1	1	1	1	1	1	0
f_{22}	1	1	1	0	1	1	0
f_{23}	0	1	0	0	0	0	1
f_{24}	0	1	0	0	1	1	1

DEPSO achieves the best performance. In f_{13} , CWDEPSO succeeds in achieving the theoretical optimum, but WORST and S.D. are slightly worse than AIWCPSO, and the mean value is inferior to JADE. For f_{16} , f_{17} , f_{19} , and f_{21} , CWDEPSO shows the highest convergence accuracy

compared to other comparing algorithms, finding the optimal result and providing the best performance. For f_{14} and f_{15} , DE can search for the optimal value, and CWDEPSO performs the same as DE in terms of the best value. For f_{18} and f_{20} , the search for the optimal solution is

TABLE 6: The running time of the eight algorithms on different test functions.

	CWDEPSO	MPSPSO-ST (s)	JADE (s)	PSO (s)	DE	AIWCPSO	SinDE	DEPSO
f_1	1.0653s	0.7287	0.9723	0.1651	0.3861s	0.1721s	0.3552s	11.5504s
f_2	1.0995s	0.9503	1.0811	0.1796	0.4291s	0.2171s	0.4515s	21.9234s
f_3	1.3302s	1.3954	1.3192	0.2940	0.6151s	0.2938s	0.6389s	14.7475s
f_4	1.2175s	1.0766	1.1902	0.2430	0.5144s	0.2323s	0.5097s	22.8226s
f_5	1.1963s	1.1725	1.1746	0.2579	0.5292s	0.2528s	0.5412s	32.0585s
f_6	1.1079s	0.9551	1.1744	0.2043	0.4396s	0.2085s	0.4317s	23.2556s
f_7	1.0881s	0.8567	1.1179	0.2091	0.4183s	0.1911s	0.4090s	21.2581s
f_8	1.4138s	1.1077	1.3764	0.2626	0.5914s	0.2577s	0.5376s	28.5731s
f_9	1.2172s	1.0532	1.2247	0.2400	0.4730s	0.2285s	0.4757s	20.5048s
f_{10}	1.3385s	1.5277	1.4469	0.3153	0.6731s	0.3209s	0.6608s	41.6464s
f_{11}	1.1542s	0.9217	1.1914	0.2010	0.4477s	0.2228s	0.4591s	22.5763s
f_{12}	1.2787s	1.1229	1.2366	0.2419	0.5173s	0.2373s	0.5246s	28.8664s
f_{13}	1.2617s	1.3655	1.3003	0.2950	0.6161s	0.2943s	0.6128s	32.3458s
f_{14}	33.2016s	80.7301	34.8477	16.0725	33.4909s	16.0757s	33.8106s	826.263s
f_{15}	39.6449s	96.0686	40.4186	18.7720	38.1136s	18.8152s	36.9709s	989.603s
f_{16}	3.6889s	4.2173	3.9664	0.7780	1.6266s	0.8859s	1.5547s	74.861s
f_{17}	4.7480s	6.9420	6.2873	1.4950	2.5856s	1.4324s	3.2057s	209.717
f_{18}	5.0012s	6.7874	5.4142	1.4410	2.8880s	1.4071s	2.9385s	92.6985s
f_{19}	5.6041s	2.6887	6.1948	1.6221	3.3629s	1.5848s	3.3518s	126.543s
f_{20}	8.2152s	11.0186	8.0453	2.1772	4.5801s	2.2404s	4.2489s	227.856s
f_{21}	3.5906s	2.7018	3.6265	0.6090	1.3123s	0.5891s	1.2948s	62.803s
f_{22}	3.1939s	1.8792	3.4971	0.2735	0.9010	0.3507	0.9208	51.538s
f_{23}	0.9269s	0.6128	0.9996	0.1083	0.3092s	0.1103s	0.3138s	4.9359s
f_{24}	1.4982s	1.9853	1.4977	0.3796	0.8699s	0.3763s	0.8538s	17.9298s

found by DEPSO and JADE, respectively, but for f_{18} , CWDEPSO performs best in the best and successfully achieves the theoretical optimal solution of the function. For the fixed-dimensional multimodal benchmark functions f_{22} , f_{23} , and f_{24} , CWDEPSO still has a good performance. It performs slightly worse than JADE on function f_{23} and achieves the best performance in f_{24} .

In accordance with [70], to make the comparison results statistically significant and reliable, Wilcoxon's rank-sum test was used to test the results of different algorithms for significant differences at 5% significance level. If there is a significant difference, then it is 1; otherwise, it is 0. Table 5 shows the results of the Wilcoxon's rank-sum test for each algorithm on each function. We can find that, in $f_1 \sim f_{11}$ and $f_{14} \sim f_{22}$, CWDEPSO is significantly different from the other algorithms on most of the tested functions; however, there is no significant difference in some of the functions: DE(f_9), PSO(f_9, f_{16}), JADE(f_6, f_9, f_{16}), SinDE(f_9, f_{17}), AIWCPSO(f_{16}, f_{20}), MPSPSO-ST(f_{16}, f_{19}, f_{22}), and DEPSO($f_{15} \sim f_{22}$). In f_{12} , there is a significant difference only with SinDE and DEPSO, and in f_{13} , there is a significant difference only with PSO. In f_{23} and f_{24} , there are less significant differences since all the algorithms converge around the theoretical optimum. But there are still significant differences in some functions: DE(f_{23}, f_{24}), DEPSO(f_{23}, f_{24}), JADE(f_{23}), and SinDE(f_{24}).

Figure 9 shows the convergence images of the eight algorithms in the 10 tested functions ($f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{13}, f_{17}, f_{22}$). From Figure 9(a), it can be seen that CWDEPSO has the best performance in this function. In Figure 9(b), we can find that CWDEPSO has the largest convergence accuracy and convergence speed. As shown in Figure 9(c), CWDEPSO still has the best

performance in terms of convergence accuracy and speed. In Figure 9(d), CWDEPSO can effectively overcome the noise interference of this function, and the convergence accuracy and speed are optimal. In terms of multimodal functions, DEPSO shows the best performance in Figures 9(e) and 9(h). In Figure 9(f), CWDEPSO has a higher convergence accuracy than the remaining seven algorithms. In Figure 9(g), CWDEPSO has excellent global and local search capability and can escape local extremes effectively. In Figure 9(i), CWDEPSO is clearly superior to the remaining seven algorithms. In Figure 9(j), JADE achieves the highest convergence accuracy, but CWDEPSO converges very close to it and converges faster. In Figures 9(k) and 9(l), CWDEPSO outperforms all other algorithms with the fastest convergence speed and the best search accuracy.

In this paper, comparing algorithm CWDEPSO to PSO, DE adds some parameter operations. Among them, the scaling factor F and crossover probability CR in the DE algorithm account for most of them, which increases the running time a lot. It is known that the time complexity of both PSO and DE is $O(G \max \cdot Np \cdot D)$ [41]. Compared with PSO, CWDEPSO adds the introduction of the DE algorithm, which has a time complexity of $O(G \max \cdot Np \cdot D)$, and also adds the parameter update process, which has a complexity of $O(G \max \cdot Np)$, so the complexity of CWDEPSO is $O(G \max \cdot Np \cdot D)$. Table 6 shows the running time required for each algorithm until all iterations are completed. We can find that PSO requires the least amount of computing time, followed by AIWCPSO, but DEPSO takes a very long time. Also, the algorithms proposed in this paper, CWDEPSO and JADE, run in approximately the same time. It is worth noting that CWDEPSO takes less time than MPSPSO-ST in

TABLE 7: Experiments were carried out on 24 classical test functions with CWDEPSO, GSA, ABC, MFO, SCA, and BBO.

f	Criteria	CWDEPSO	GSA	ABC	MFO	SCA	BBO
f_1	Best	2.01E-28	3.13E-17	5.72E-06	5.02E-01	1.73E-10	9.05E+01
	Worst	4.89E-24	9.32E-17	1.03E-04	1.17E+01	4.30E+02	2.21E+02
	Mean	6.29E-25	6.19E-17	3.55E-05	3.06E+00	4.68E+01	1.52E+02
	S.D.	4.59E-24	5.49E-17	9.97E-05	9.75E+00	4.05E+02	1.38E+02
f_2	Best	4.83E-25	3.24E-01	5.92E-03	4.49E+03	2.19E-08	1.05E+06
	Worst	2.90E-21	9.06E+00	5.92E-03	1.00E+08	8.60E+06	2.81E+06
	Mean	5.65E-22	3.94E+00	2.94E-02	1.00E+07	8.68E+05	1.62E+06
	S.D.	3.02E-21	9.70E+00	4.89E-02	9.49E+07	8.15E+06	1.77E+06
f_3	Best	1.93E-76	1.28E+13	1.30E-03	1.01E+28	2.01E-13	2.55E+07
	Worst	1.82E-68	2.29E+19	1.93E-01	1.00E+44	1.62E+17	9.94E+16
	Mean	2.00E-69	2.46E+18	4.05E-02	1.03E+43	1.62E+16	9.99E+15
	S.D.	1.71E-68	2.16E+19	1.75E-01	9.46E+43	1.54E+17	9.43E+16
f_4	Best	5.37E-19	5.41E+01	2.58E+02	9.18E+01	3.89E+00	5.30E+01
	Worst	1.83E-16	9.48E+01	3.44E+02	4.36E+02	6.75E+01	1.15E+02
	Mean	4.28E-17	7.34E+01	2.94E+02	2.65E+02	2.01E+01	8.54E+01
	S.D.	2.00E-16	3.94E+01	8.12E+01	3.02E+02	6.06E+01	6.00E+01
f_5	Best	7.89E-04	2.80E-02	1.05E-01	1.13E-01	8.50E-03	8.18E-02
	Worst	2.14E-03	7.81E-02	3.10E-01	8.18E+00	4.64E-01	4.04E-01
	Mean	1.44E-03	4.59E-02	2.22E-01	2.84E+00	9.12E-02	2.77E-01
	S.D.	1.31E-03	4.95E-02 2	05E-01	1.07E+01	4.18E-01	2.83E-01
f_6	Best	9.37E-02	2.63E+01	5.12E+00	9.30E+01	2.78E+01	4.01E+02
	Worst	2.35E+01	9.44E+01	8.90E+01	1.11E+03	8.46E+04	7.93E+02
	Mean	1.84E+01	3.43E+01	2.41E+01	2.56E+02	9.06E+03	5.88E+02
	S.D.	2.49E+01	6.34E+01	8.00E+01	9.29E+02	7.97E+04	4.19E+02
f_7	Best	0.00E+00	1.09E+01	5.04E+00	1.27E+02	7.47E-11	1.63E+01
	Worst	2.27E+01	3.28E+01	1.09E+01	2.32E+02	1.31E+02	2.88E+01
	Mean	2.46E+00	1.92E+01	6.87E+00	1.80E+02	1.50E+01	2.26E+01
	S.D.	2.13E+01	1.74E+01	6.17E+00	8.05E+01	1.23E+02	1.30E+01
f_8	Best	5.06E-14	5.78E-09	2.74E-02	1.46E+00	2.00E+01	3.75E+00
	Worst	5.91E-13	8.56E-09	1.93E-01	2.00E+01	2.00E+01	4.77E+00
	Mean	1.32E-13	6.73E-09	7.80E-02	1.37E+01	2.00E+01	4.34E+00
	S.D.	4.88E-13	2.68E-09	1.65E-01	2.52E+01	4.96E-02	1.04E+00
f_9	Best	1.87E-07	1.46E-17	2.10E-07	1.91E+01	2.06E+00	4.88E-01
	Worst	3.58E-01	1.79E-01	1.45E-05	4.23E+01	5.27E+00	8.73E-01
	Mean	4.48E-02	4.48E-02	3.72E-06	2.84E+01	2.58E+00	6.33E-01
	S.D.	3.41E-01	2.28E-01	1.52E-05	2.47E+01	2.91E+00	4.30E-01
f_{10}	Best	4.53E-28	7.43E+02	2.29E-04	8.49E+03	1.93E-10	5.76E+03
	Worst	2.61E-24	1.39E+04	2.56E-02	9.33E+05	1.60E+02	3.29E+04
	Mean	4.72E-25	4.51E+03	4.13E-03	3.50E+05	1.65E+01	1.66E+04
	S.D.	2.64E-24	1.26E+04	2.34E-02	9.61E+05	1.51E+02	2.32E+04
f_{11}	Best	1.62E-29	2.09E+01	1.53E-06	2.11E-02	8.18E-14	2.03E+00
	Worst	2.21E-26	7.85E+01	2.35E-03	4.00E+02	1.16E+00	4.48E+03
	Mean	5.35E-27	4.15E+01	3.69E-04	1.00E+02	1.21E-01	1.03E+03
	S.D.	2.15E-26	6.37E+01	2.27E-03	3.74E+02	1.10E+00	4.46E+03
f_{12}	Best	3.31E-01	4.32E-01	3.09E-01	3.18E-01	4.85E-01	4.61E-01
	Worst	6.02E-01	5.30E-01	4.33E-01	2.04E+02	8.40E-01	1.49E+00
	Mean	4.48E-01	4.76E-01	3.75E-01	6.14E+01	6.57E-01	1.01E+00
	S.D.	2.78E-01	8.61E-02	1.03E-01	2.13E+02	4.74E-01	1.19E+00
f_{13}	Best	0.00E+00	1.42E+01	1.85E-03	6.77E-01	3.39E-11	1.98E+00
	Worst	3.43E-02	2.79E+01	9.47E-02	1.80E+02	1.54E+00	4.60E+00
	Mean	9.84E-03	2.09E+01	3.43E-02	1.89E+01	2.83E-01	2.91E+00
	S.D.	3.29E-02	1.26E+01	8.84E-02	1.70E+02	1.58E+00	2.20E+00
f_{14}	Best	-8.70E+02	-8.58E+02	-8.70E+02	-8.70E+02	-8.70E+02	-7.90E+02
	Worst	-7.73E+02	-7.76E+02	-8.70E+02	-8.70E+02	-8.48E+02	-7.54E+02
	Mean	-8.33E+02	-8.27E+02	-8.70E+02	-8.70E+02	-8.57E+02	-7.74E+02
	S.D.	9.73E+01	8.43E+01	0.00E+00	0.00E+00	2.51E+01	4.13E+01

TABLE 7: Continued.

f	Criteria	CWDEPSO	GSA	ABC	MFO	SCA	BBO
f_{15}	Best	0.00E+00	4.15E-11	0.00E+00	0.00E+00	2.47E-13	2.13E-12
	Worst	1.86E-12	5.78E-11	4.12E-14	0.00E+00	2.96E-12	3.31E-12
	Mean	5.09E-13	5.37E-11	4.12E-15	0.00E+00	1.26E-12	2.58E-12
	S.D.	1.98E-12	1.37E-11	3.91E-14	0.00E+00	2.36E-12	1.18E-12
f_{16}	Best	3.04E-02	1.39E-01	2.53E-01	5.00E-01	8.52E-01	6.06E-01
	Worst	6.29E-01	2.52E-01	4.00E-01	4.42E+00	1.26E+00	1.02E+00
	Mean	4.02E-01	1.96E-01	3.32E-01	1.03E+00	1.02E+00	8.33E-01
	S.D.	5.12E-01	1.13E-01	1.57E-01	3.59E+00	3.78E-01	3.51E-01
f_{17}	Best	0.00E+00	6.04E-01	0.00E+00	0.00E+00	0.00E+00	1.90E-01
	Worst	0.00E+00	7.46E-01	4.06E-02	3.69E-02	0.00E+00	2.93E-01
	Mean	0.00E+00	6.70E-01	1.21E-02	3.69E-03	0.00E+00	2.39E-01
	S.D.	0.00E+00	1.10E-01	4.62E-02	3.50E-02	0.00E+00	9.08E-02
f_{18}	Best	1.78E-15	1.50E+01	6.04E+00	7.90E+01	8.17E+00	2.02E+01
	Worst	2.80E+01	3.70E+01	1.21E+01	2.63E+02	1.30E+02	3.08E+01
	Mean	5.71E+00	2.56E+01	8.84E+00	1.58E+02	5.18E+01	2.47E+01
	S.D.	3.53E+01	1.95E+01	5.19E+00	1.97E+02	1.35E+02	8.88E+00
f_{19}	Best	4.67E-11	1.05E-01	1.26E-06	1.91E+00	5.20E-01	9.28E-01
	Worst	2.55E-07	2.56E+00	2.73E-04	8.47E+00	1.31E+04	2.76E+00
	Mean	6.55E-08	1.12E+00	3.23E-05	4.88E+00	1.31E+03	1.43E+00
	S.D.	2.92E-07	2.24E+00	2.54E-04	5.52E+00	1.24E+04	1.69E+00
f_{20}	Best	6.14E-09	1.06E-17	3.31E-06	3.91E+00	2.20E+00	2.16E+00
	Worst	1.10E-02	1.55E+01	1.43E-03	3.32E+01	2.06E+05	1.47E+01
	Mean	1.10E-03	6.58E+00	2.15E-04	1.26E+01	2.06E+04	7.32E+00
	S.D.	1.04E-02	1.58E+01	1.32E-03	2.95E+01	1.95E+05	9.81E+00
f_{21}	Best	6.02E-09	3.36E-09	5.81E-03	1.53E-02	9.98E-07	4.11E-01
	Worst	1.85E-03	7.46E-03	1.07E-01	1.09E+01	5.55E+00	1.13E+00
	Mean	2.04E-04	2.05E-03	5.51E-02	4.66E+00	8.07E-01	7.24E-01
	S.D.	1.74E-03	7.29E-03	1.18E-01	1.34E+01	5.22E+00	6.00E-01
f_{22}	Best	3.07E-04	9.95E-04	5.25E-04	5.35E-04	6.27E-04	1.33E-03
	Worst	4.24E-04	6.45E-03	1.85E-03	8.33E-03	1.77E-03	2.09E-02
	Mean	3.19E-04	3.29E-03	1.08E-03	1.75E-03	1.38E-03	5.86E-03
	S.D.	1.11E-04	4.65E-03	1.31E-03	7.02E-03	9.24E-04	2.38E-02
f_{23}	Best	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.01E+00	-3.32E+00
	Worst	-3.20E+00	-3.32E+00	-3.32E+00	-3.14E+00	-2.98E+00	-3.20E+00
	Mean	-3.30E+00	-3.32E+00	-3.32E+00	-3.24E+00	-2.99E+00	-3.30E+00
	S.D.	1.50E-01	1.33E-15	1.83E-15	2.31E-01	3.38E-02	1.49E-01
f_{24}	Best	-1.05E+01	-1.05E+01	-1.05E+01	-1.05E+01	-5.08E+00	-1.02E+01
	Worst	-1.05E+01	-2.43E+00	-5.18E+00	-2.43E+00	-2.49E+00	-2.43E+00
	Mean	-1.05E+01	-9.73E+00	-1.00E+01	-8.24E+00	-4.36E+00	-5.48E+00
	S.D.	8.14E-15	7.69E+00	5.08E+00	1.11E+01	2.61E+00	9.82E+00

some complex test functions f_{10} , f_{14} , f_{15} , etc., and the time is roughly similar to DE. The extra time is mainly spent on parameter update operations, similar to JADE. Therefore, as the difficulty in the test function increases, the running time does not vary as much as that of DEPSO and MPSPSO-ST.

In general, as shown in Tables 4 and 5 and Figure 9, CWDEPSO has better convergence speed and search accuracy than PSO, DE, AIWCPSO, MPSPSO-ST, JADE, SinDE, and DEPSO, so combining DE and PSO in this way can effectively improve the algorithm. The performance proves that CWDEPSO is a more efficient algorithm for numerical optimization.

5.3. Comparison of CWDEPSO with Other Algorithms. In this part, we compared CWDEPSO algorithm with the optimization algorithms put forward recently, including the BBO [5], SCA

[6], MFO [7], ABC [11], and GSA [12]. To ensure a fair comparison, we set the parameters in both sets of experiments all according to the original paper, where the number of generations is set to 500 and the size of the population is set to 40. Each is conducted independently 20 times, and the details are listed as follows:

GSA setting: $G_0 = 100, \alpha = 20$.

ABC setting: Limit = 200.

MFO setting: t is a random number in the range $[-2, 1]$.

SCA setting: r_1, r_2, r_3 , and r_4 are random number.

BBO setting: $\mu = 0.005, \mu = 0.8$.

CWDEPSO setting: $c_1: 2.5 \sim 0.5, c_2: 0.5 \sim 2.5, \omega = 0.924 \sim 0.375, c = 0.1, V_{\max} = 0.2 \times \text{Bound}$.

(a)

(b)

(c)

(d)

FIGURE 10: Continued.

(e)

(f)

(g)

(h)

FIGURE 10: Continued.

FIGURE 10: Compare six algorithms' convergence curves for different benchmark functions: (a) f_2 ; (b) f_3 ; (c) f_5 ; (d) f_6 ; (e) f_7 ; (f) f_9 ; (g) f_{11} ; (h) f_{12} ; (i) f_{13} ; (j) f_{17} ; (k) f_{22} ; (l) f_{24} .

The test functions used in this section are shown in Tables 1–3. The outcomes gained are presented in Table 7 and Figure 10. Table 7 contains the best results, mean, and S.D. The greatest values are presented in bold. Figure 11 shows the S.D. and the mean radar charts.

According to Table 7, for the unimodal functions $f_1 \sim f_5$, the CWDEPSO algorithm performs well in terms of search stability and accuracy, and the convergence accuracy of CWDEPSO algorithm is higher compared with the other five optimization algorithms. For the multimodal functions $f_6, f_7, f_8, f_{10}, f_{11}, f_{13}$, and f_{24} , CWDEPSO still

has excellent search accuracy, search speed, and stability. For multimodal functions f_9 and f_{12} , ABC shows better performance. In the search for f_{23} , GSA and ABC have the best performance, but CWDEPSO's convergence accuracy is very close to it. For $f_{15}, f_{17}, f_{18}, f_{19}, f_{21}$, and f_{22} , CWDEPSO shows outstanding accuracy and stability of search, and its convergence accuracy is higher than the other five comparing algorithms. For f_{14}, f_{16} , and f_{20} , MFO, GSA, and ABC have the best convergence accuracy and stability, respectively. For f_{14} , CWDEPSO also finds the same optimal solution as MFO but does not perform as stable as MFO.

FIGURE 11: Continued.

FIGURE 11: Radar chart of six algorithms, mean and S.D.

TABLE 8: The results of Wilcoxon's rank-sum test for the five algorithms on different test functions.

	GSA	ABC	MFO	SCA	BBO
f_1	1	1	1	1	1
f_2	1	1	1	1	1
f_3	1	1	1	1	1
f_4	1	1	1	1	1
f_5	1	1	1	1	1
f_6	1	0	1	1	1
f_7	1	1	1	0	1
f_8	1	1	1	1	1
f_9	0	0	1	1	1
f_{10}	1	1	1	1	1
f_{11}	1	1	1	1	1
f_{12}	0	0	0	1	1
f_{13}	1	1	1	0	1
f_{14}	0	1	1	0	1
f_{15}	1	1	1	1	1
f_{16}	1	0	1	1	1
f_{17}	1	1	0	0	1
f_{18}	1	1	1	1	1
f_{19}	1	1	1	1	1
f_{20}	1	1	1	1	1
f_{21}	0	1	1	1	1
f_{22}	1	1	1	1	1
f_{23}	0	0	1	0	1
f_{24}	1	1	0	1	1

Table 8 shows the results of Wilcoxon's rank-sum test for each algorithm on each function. CWDEPSO differs significantly from the other algorithms on most of the tested functions and not significantly on a few functions: GSA ($f_9, f_{12}, f_{14}, f_{21}, f_{23}$), ABC ($f_6, f_9, f_{12}, f_{16}, f_{23}$), MFO (f_{12}, f_{17}, f_{24}), and SCA ($f_7, f_{13}, f_{17}, f_{23}$).

Figure 10 gives the convergence images for each algorithm on twelve test functions ($f_2, f_3, f_5, f_6, f_7, f_9, f_{11}, f_{12}, f_{13}, f_{17}, f_{22}, f_{24}$). These data are averaged over 20 independent experiments. For unimodal functions, in Figures 10(a)–10(c), CWDEPSO outperforms the other compared algorithms by far in terms of both convergence speed and convergence accuracy, showing the optimal search performance. For the multimode function, in Figures 10(d) and 10(e), CWDEPSO demonstrates excellent convergence speed and the ability to avoid local convergence. In Figures 10(f) and 10(h), ABC shows better performance. In Figures 10(g), 10(i), 10(k), and 10(l), CWDEPSO outperforms all other algorithms with the fastest convergence speed and the best search accuracy. In Figure 10(j), CWDEPSO is able to search for the optimal value and provides the highest convergence accuracy.

In conclusion, the work done in this paper is beneficial, which can improve the performance of the algorithms significantly. As an innovation in this paper, we introduce DE algorithm in the late stage of PSO, which improves the capability of algorithm development and exploration, enhances the population diversity, and effectively avoids premature convergence. Although the introduction of some parameters and other operations increase the runtime of the algorithm, it significantly improves the performance of the algorithm. In the improvement of inertia coefficients ω , through the introduction of nonlinear and chaotic mechanisms, the global search ability in the early stage is

strengthened along with the local search ability in the later stage, and the inclusion of chaos theory can make the algorithm more diverse and avoid falling into local extremes. The nonlinear optimization for the learning factors c_1 and c_2 enhances the late local exploitation ability and makes the algorithm less likely to converge prematurely. The two parameters, DE deflation factor F and crossover probability CR , are adaptively processed through the Cauchy and Gaussian distributions, which can be automatically updated to a suitable value by the performance of previous generations, thus improving the robustness and accuracy of the algorithm. Finally, the value of the particles is more fully utilized through an improved selection strategy, which helps to improve the algorithm performance. In summary, CWDEPSO can deal with numerical optimization problems effectively.

6. Engineering Application and Analysis

In this section, we use the CWDEPSO algorithm to optimize the two sets of engineering structures for spring tension design and pressure vessel structure design.

6.1. Spring Tension Design. Spring tension design is a common engineering problem in practice, and many works [74, 75] have studied it. The problem is composed of three decision variables: coil diameter (d), average coil diameter (D), and coil winding number (P), which are represented by x_1, x_2 , and x_3 , respectively, in this paper. Figure 12 shows the spring tension structure diagram. Our goal is to make the spring as light as possible when the three decision variables meet the constraints. The constraint problem of spring tension design is shown as follows:

$$\begin{aligned} \min f(x) &= x_1^2 x_2 (2 + x_3) \\ \text{subject to } \begin{cases} g_1(x) = 1 - \frac{x_2^3 x_3}{7178 x_1^4} \leq 0, \\ g_2(x) = \frac{4x_2^2 - x_1 x_2}{12566(x_1^3 x_2 - x_1^4)} + \frac{1}{5108 x_1^2} - 1 \leq 0, \\ g_3(x) = 1 - \frac{140.45 x_1}{x_2^3 x_3} \leq 0, \\ 0.05 \leq x_1 \leq 2, 0.25 \leq x_2 \leq 1.3, 2 \leq x_3 \leq 15. \end{cases} \end{aligned} \quad (29)$$

For this engineering design, we use the metaheuristic algorithm to solve it. We choose renewing the particles whenever the constraints are violated. CWDEPSO is used to solve the spring tension optimization problem, and the results are compared with those of other metaheuristic algorithms AIWCP SO [52], ABC [11], JADE [45], CPSO [25],

and GSA [12]. The results of the comparison are shown in Table 9.

From the data in Table 9, it can be seen that CWDEPSO performs best among the six algorithms and can successfully find excellent solutions under several constraints. CPSO and JADE got the second and third best solutions, respectively;

FIGURE 12: Schematic diagram of spring tension design.

TABLE 9: The best solution of six algorithms solving the spring tension design.

Algorithm	x_1	x_2	x_3	$f(x)$
AIWCPSO	0.058751118185016	0.347113573247800	10.980555889151308	0.015552
ABC	0.0500325637475810	0.283039500783788	2.00008949648746	0.0028341
JADE	0.071951969130501	0.471228721140340	2.065939821271334	0.0099192
CPSO	0.207092843130427	1.14580154678520	7.64997103999727	0.0033512
GSA	0.0558993256271489	0.324136359769932	4.86248662696000	0.0069506
CWDEPSO	0.0500000000000009	0.282022721751902	2.00000542008347	0.0028202

FIGURE 13: Schematic diagram of pressure vessel design.

TABLE 10: The best solution of six algorithms solving the pressure vessel design.

Algorithm	x_1	x_2	x_3	x_4	$f(x)$
AIWCPSO	0.840421320019978	0.415420693937337	43.5451461150248	159.498089121814	6000.48281
SCA	0.863388502228	0.448954461704	43.656747097939	164.07320040515	6754.8157
BBO	1.570707301106	0.557690189563	57.528423836813	48.085076155174	7079.85636
CLPSO	0.922188471264	0.455838239163	47.781786075864	116.97939244862	6179.8822
MFO	1.21665366059615	0.601392804181567	63.0390338022153	19.7573957836916	7136.5180
CWDEPSO	0.814596471868821	0.402655458115425	42.2070710812561	175.295185842192	5950.5602

and the value of the obtained decision variables differs significantly from that obtained by CWDEPSO, which indicates that CWDEPSO has a strong exploration ability and can effectively find the best value. Besides, AIWCPSO and ABC performed moderately, while GSA, which did well in the previous section, performed worst. It shows that these algorithms will fall into a local extremum when dealing with complex multiconstraint problems, and it is difficult to converge to the optimal value. Compared with them,

CWDEPSO is less affected in dealing with the optimization of the spring tension structure.

6.2. Pressure Vessel Design. This part uses the experiment of pressure vessel design first proposed by Sandgren [76] to verify the performance of the CWDEPSO algorithm further. It contains four decision variables: cylindrical container thickness (T_s), hemispherical cover thickness (T_h), cylin-

dical container inner diameter (R), and cylindrical container length (L), which are represented by x_1 , x_2 , x_3 , and x_4 , respectively. The mathematical model of the experiment is shown in Figure 13.

$$\begin{aligned} \min f(x) &= 19.84x_1^2x_3 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 \\ &\quad + 0.6224x_1x_3x_4 \\ \text{subject to } &\begin{cases} g_1(x) = -x_1 + 0.0193x_3 \leq 0, \\ g_2(x) = -x_2 + 0.00954x_3 \leq 0, \\ g_3(x) = -\pi x_3^2x_4 - \frac{4\pi x_3^3}{3} + 1296000 \leq 0, \\ g_4(x) = x_4 - 240 \leq 0, \\ x_1 \geq 0, x_2 \leq 10, x_3 \geq 10, x_4 \leq 200. \end{cases} \end{aligned} \quad (30)$$

CWDEPSO and the recently proposed AIWCPSO [52], SCA [6], BBO [5], CLPSO [24], and MFO [7] algorithms are used to solve this engineering experiment. The comparison results are shown in Table 10.

We can see from Table 10 that the CWDEPSO algorithm is the most accurate than the other five algorithms in the optimization of the pressure vessel design. MFO performed the second best, but there was a big gap between variable x_4 and CWDEPSO, indicating that CWDEPSO had better exploration ability. AIWCPSO, SCA, and CLPSO have a general performance. Compared with CWDEPSO, there are some differences in the obtained decision variables and solution results, which shows that CWDEPSO has better search accuracy. Finally, the performance of BBO is unsatisfactory, and the convergence speed is slower than other algorithms.

Through the above two experiments, we can find that CWDEPSO is feasible and effective in the treatment of engineering structure optimization. Besides, it performs well in the treatment of engineering structure optimization, and its application scope is relatively broad.

7. Summary and Future Work

In this paper, for the purpose for overcoming the shortcomings of PSO and improving its performance of engineering structure design, the CWDEPSO algorithm is proposed. We introduce mutation operation and other methods according to the probability related to the current convergence failure time. This strengthens the communication between individuals within the population and improves the diversity of the population in the later iterations of the algorithm, effectively helping PSO get rid of premature convergence. In addition, nonlinear dynamic optimization and adaptive adjustment of PSO and DE parameters are performed, respectively, and chaotic processing is added to the inertia weight. By

optimizing the parameters, the algorithm can strengthen the global search ability in the early stage of iteration and focus on local search in the late stage of iteration. Moreover, the randomness and ergodicity brought by chaos make the algorithm more powerful in exploration. CWDEPSO can be used to solve numerical optimization and practical problems as well as swarm intelligence algorithms such as MBO, EWA, EHO, MS, SMA, and HHO. In this paper, 24 benchmark functions and two groups of engineering optimization experiments have been adopted for verifying the efficiency of the strategy put forward in the paper. In the primary stage of the tests, CWDEPSO is compared with PSO and DE and their classical variants. The tentative outcomes present that CWDEPSO has better results for almost all benchmark functions and has excellent robustness and stability. In the second set of experiments, CWDEPSO is compared with five common metaheuristic algorithms. The results suggest that CWDEPSO outperforms other algorithms in most benchmark functions. Finally, CWDEPSO and some metaheuristic algorithms are applied to solve the engineering structure optimization experiment in the last set of experiments. Compared with other algorithms, CWDEPSO obtained the best solution in both spring tension structure optimization and pressure vessel structure optimization. In addition, compared with other metaheuristic algorithms, CWDEPSO has the advantages of low time complexity and simple parameter setting, enhanced local search capability, is less likely to fall into local optima, and has faster convergence speed. To sum up, the CWDEPSO algorithm proposed in this article is successful and suitable for optimization problems in the field of numerical and engineering structure optimization.

In addition to optimizing engineering structures, it can also be widely used in many fields such as national defense, transportation, agriculture, chemical industry, materials, and communications. For example, making a phone call in life includes the process of finding the lowest cost and least congested route; in the design of mechanical structures, the cost will be minimized while meeting the stress requirements; in the field of robotics, it can be applied to path planning in the dynamic environment of mobile robots. However, for some functions, the effect of CWDEPSO is not ideal, and it is not easy to find the optimal value. Therefore, improving the stability of the CWDEPSO algorithm is still worth investigating further. In future work, we will try to learn from APSO [61] to further improve the performance of CWDEPSO algorithm, and CWDEPSO will be used to resolve more practical application problems.

Data Availability

The data used to support the findings of this study have been deposited in the GitHub repository (<https://github.com/MaZhiteng/A-Hybrid-Dynamic-Probability-Mutation-Particle-Swarm-Optimization-for-Engineering-Structure-Design.git>).

Disclosure

This research received no external funding.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

- [1] Y.-Y. Wang, W.-X. Peng, C.-H. Qiu, J. Jiang, and S.-R. Xia, "Fractional-order Darwinian PSO-based feature selection for media-adventitia border detection in intravascular ultrasound images," *Ultrasonics*, vol. 92, pp. 1–7, 2019.
- [2] G. Wang, J. Guo, Y. Chen, Y. Li, and Q. Xu, "A PSO and BFO-based learning strategy applied to faster R-CNN for object detection in autonomous driving," *IEEE Access*, vol. 7, pp. 18840–18859, 2019.
- [3] J. H. Tam, Z. C. Ong, Z. Ismail, B. C. Ang, and S. Y. Khoo, "A new hybrid GA-ACO-PSO algorithm for solving various engineering design problems," *International Journal of Computer Mathematics*, vol. 96, no. 5, pp. 883–919, 2019.
- [4] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.
- [5] H. Ma, D. Simon, P. Siarry, Z. Yang, and M. Fei, "Biogeography-based optimization: a 10-year review," *IEEE Transactions on Emerging Topics in Computational Intelligence*, vol. 1, no. 5, pp. 391–407, 2017.
- [6] A.-F. Attia, R. A. El Sehiemy, and H. M. Hasanien, "Optimal power flow solution in power systems using a novel sine-cosine algorithm," *International Journal of Electrical Power & Energy Systems*, vol. 99, pp. 331–343, 2018.
- [7] S. Khalilpourazari and S. Khalilpourazary, "An efficient hybrid algorithm based on water cycle and moth-flame optimization algorithms for solving numerical and constrained engineering optimization problems," *Soft Computing*, vol. 23, no. 5, pp. 1699–1722, 2019.
- [8] W. Deng, J. Xu, and H. Zhao, "An improved ant colony optimization algorithm based on hybrid strategies for scheduling problem," *IEEE Access*, vol. 7, pp. 20281–20292, 2019.
- [9] L. M. Abualigah, A. T. Khader, and E. S. Hanandeh, "A combination of objective functions and hybrid Krill herd algorithm for text document clustering analysis," *Engineering Applications of Artificial Intelligence*, vol. 73, pp. 111–125, 2018.
- [10] G.-G. Wang, A. H. Gandomi, A. H. Alavi, and D. Gong, "A comprehensive review of krill herd algorithm: variants, hybrids and applications," *Artificial Intelligence Review*, vol. 51, no. 1, pp. 119–148, 2019.
- [11] C. Ozturk, E. Hancer, and D. Karaboga, "A novel binary artificial bee colony algorithm based on genetic operators," *Information Sciences*, vol. 297, pp. 154–170, 2015.
- [12] S. Mirjalili and A. H. Gandomi, "Chaotic gravitational constants for the gravitational search algorithm," *Applied Soft Computing*, vol. 53, pp. 407–419, 2017.
- [13] Y. Feng, S. Deb, G. G. Wang, and A. H. Alavi, "Monarch butterfly optimization: a comprehensive review," *Expert Systems with Applications*, vol. 168, p. 114418, 2021.
- [14] G.-G. Wang, S. Deb, and Z. Cui, "Monarch butterfly optimization," *Neural Computing and Applications*, vol. 31, no. 7, pp. 1995–2014, 2019.
- [15] G. G. Wang, S. Deb, and L. D. S. Coelho, "Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems," *International Journal of Bio-Inspired Computation*, vol. 12, no. 1, pp. 1–22, 2018.
- [16] G. G. Wang, S. Deb, and L. D. S. Coelho, "Elephant herding optimization," in *Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI)*, pp. 1–5, IEEE, Bali, Indonesia, December 2015.
- [17] J. Li, H. Lei, A. H. Alavi, and G.-G. Wang, "Elephant herding optimization: variants, hybrids, and applications," *Mathematics*, vol. 8, no. 9, p. 1415, 2020.
- [18] G.-G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, no. 2, pp. 151–164, 2018.
- [19] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: a new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.
- [20] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.
- [21] P. Rakshit, "Improved differential evolution for noisy optimization," *Swarm and Evolutionary Computation*, vol. 52, Article ID 100628, 2020.
- [22] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 International Conference on Neural Networks*, vol. 4, pp. 1942–1948, Perth, WA, Australia, November 1995.
- [23] A. Goudarzi, Y. Li, and J. Xiang, "A hybrid non-linear time-varying double-weighted particle swarm optimization for solving non-convex combined environmental economic dispatch problem," *Applied Soft Computing*, vol. 86, Article ID 105894, 2020.
- [24] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.
- [25] J. Liu and Y. Gao, "Chaos particle swarm optimization algorithm," *Computer Science*, vol. 31, pp. 13–15, 2004.
- [26] Z. J. Wang, Z. H. Zhan, S. Kwong, H. Jin, and J. Zhang, "Adaptive granularity learning distributed particle swarm optimization for large-scale optimization," *IEEE Transactions on Cybernetics*, vol. 51, no. 3, 2020.
- [27] A. R. Jordehi, "Time varying acceleration coefficients particle swarm optimisation (TVACPSO): a new optimisation algorithm for estimating parameters of PV cells and modules," *Energy Conversion and Management*, vol. 129, pp. 262–274, 2016.
- [28] J. Y. Li, Z. H. Zhan, R. D. Liu, C. Wang, S. Kwong, and J. Zhang, "Generation-level parallelism for evolutionary computation: a pipeline-based parallel particle swarm optimization," *IEEE Transactions on Cybernetics*, 2020.
- [29] S. K. S. Fan and C. H. Jen, "An enhanced partial search to particle swarm optimization for unconstrained optimization," *Mathematics*, vol. 7, no. 4, p. 357, 2019.
- [30] Z. Ma, X. Yuan, S. Han, D. Sun, and Y. Ma, "Improved chaotic particle swarm optimization algorithm with more symmetric distribution for numerical function optimization," *Symmetry*, vol. 11, no. 7, p. 876, 2019.
- [31] X. Zhang, K. J. Du, Z. H. Zhan, S. Kwong, T. L. Gu, and J. Zhang, "Cooperative coevolutionary bare-bones particle swarm optimization with function independent decomposition for large-scale supply chain network design with

- uncertainties,” *IEEE Transactions on Cybernetics*, vol. 50, no. 10, pp. 4454–4468, 2019.
- [32] X. F. Liu, Z. H. Zhan, Y. Gao, J. Zhang, S. Kwong, and J. Zhang, “Coevolutionary particle swarm optimization with bottleneck objective learning strategy for many-objective optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 4, pp. 587–602, 2018.
- [33] X. Xia, L. Gui, F. Yu et al., “Triple archives particle swarm optimization,” *IEEE Transactions on Cybernetics*, vol. 50, no. 12, pp. 4862–4875, 2019.
- [34] S. Wang, Y. Li, H. Yang, and H. Liu, “Self-adaptive differential evolution algorithm with improved mutation strategy,” *Soft Computing*, vol. 22, no. 10, pp. 3433–3447, 2018.
- [35] G. Wu, R. Mallipeddi, P. N. Suganthan, R. Wang, and H. Chen, “Differential evolution with multi-population based ensemble of mutation strategies,” *Information Sciences*, vol. 329, pp. 329–345, 2016.
- [36] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, “Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems,” *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.
- [37] A. K. Qin and P. N. Suganthan, “Self-adaptive differential evolution algorithm for numerical optimization,” in *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, vol. 2, pp. 1785–1791, Edinburgh, Scotland, September 2005.
- [38] H. Zhao, Z.-H. Zhan, Y. Lin et al., “Local binary pattern-based adaptive differential evolution for multimodal optimization problems,” *IEEE Transactions on Cybernetics*, vol. 50, no. 7, pp. 3343–3357, 2020.
- [39] Z. J. Wang, Z. H. Zhan, Y. Lin et al., “Automatic niching differential evolution with contour prediction approach for multimodal optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 1, pp. 114–128, 2019.
- [40] Z.-H. Zhan, X.-F. Liu, H. Zhang et al., “Cloudde: a heterogeneous differential evolution algorithm and its distributed cloud version,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 3, pp. 704–716, 2016.
- [41] Z. J. Wang, Z. H. Zhan, W. J. Yu et al., “Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling,” *IEEE Transactions on Cybernetics*, vol. 50, no. 6, pp. 2715–2729, 2019.
- [42] X.-F. Liu, Z. H. Zhan, Y. Lin et al., “Historical and heuristic-based adaptive differential evolution,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 49, no. 12, pp. 2623–2635, 2018.
- [43] Z. H. Zhan, Z. J. Wang, H. Jin, and J. Zhang, “Adaptive distributed differential evolution,” *IEEE Transactions on Cybernetics*, vol. 50, no. 11, pp. 4633–4647, 2019.
- [44] Z. G. Chen, Z. H. Zhan, H. Wang, and J. Zhang, “Distributed individuals for multiple peaks: a novel differential evolution for multimodal optimization problems,” *IEEE Transactions on Evolutionary Computation*, vol. 24, no. 4, pp. 708–719, 2019.
- [45] J. Zhang and A. C. Sanderson, “JADE: adaptive differential evolution with optional external archive,” *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.
- [46] S. Wang, Y. Li, and H. Yang, “Self-adaptive mutation differential evolution algorithm based on particle swarm optimization,” *Applied Soft Computing*, vol. 81, Article ID 105496, 2019.
- [47] H. Wang, L. L. Zuo, J. Liu, W. J. Yi, and B. Niu, “Ensemble particle swarm optimization and differential evolution with alternative mutation method,” *Natural Computing*, vol. 19, no. 4, pp. 1–14, 2018.
- [48] S. Sayah and A. Hamouda, “A hybrid differential evolution algorithm based on particle swarm optimization for non-convex economic dispatch problems,” *Applied Soft Computing*, vol. 13, no. 4, pp. 1608–1619, 2013.
- [49] H. Liu, Z. Cai, and Y. Wang, “Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization,” *Applied Soft Computing*, vol. 10, no. 2, pp. 629–640, 2010.
- [50] J. A. Joines and C. R. Houck, “On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA’s,” in *Proceedings of the First IEEE Conference on Evolutionary Computation*, pp. 579–584, Orlando, FL, USA, June 1994.
- [51] D. Zhou and J. Sun, “Influences of genetic operators on population diversity in genetic programming,” *Computer Engineering and Applications*, vol. 52, pp. 39–45, 2016.
- [52] Y. Feng, G. F. Teng, A. X. Wang, and Y. M. Yao, “Chaotic inertia weight in particle swarm optimization,” in *Proceedings of the 2007 Second International Conference on Innovative Computing, Information and Control*, pp. 1899–1902, Kumamoto, Japan, September 2007.
- [53] Y. Du and F. Xu, “A hybrid multi-step probability selection particle swarm optimization with dynamic chaotic inertial weight and acceleration coefficients for numerical function optimization,” *Symmetry*, vol. 12, no. 6, p. 922, 2020.
- [54] A. Draa, S. Bouzoubia, and I. Boukhalifa, “A sinusoidal differential evolution algorithm for numerical optimisation,” *Applied Soft Computing*, vol. 27, pp. 99–126, 2015.
- [55] M. J. Lin, F. Luo, C. H. Su et al., “An improved hybrid intelligent extreme learning machine,” *Control and Decision*, vol. 30, no. 6, pp. 1078–1084, 2015.
- [56] Y. Shi and R. Eberhart, “Modified particle swarm optimizer,” in *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp. 69–73, Anchorage, AK, USA, May 1999.
- [57] A. H. Gandomi and X.-S. Yang, “Chaotic bat algorithm,” *Journal of Computational Science*, vol. 5, no. 2, pp. 224–232, 2014.
- [58] K. Chen, F. Zhou, and A. Liu, “Chaotic dynamic weight particle swarm optimization for numerical function optimization,” *Knowledge-Based Systems*, vol. 139, pp. 23–40, 2018.
- [59] K. Chen, F. Zhou, Y. Wang, and L. Yin, “An ameliorated particle swarm optimizer for solving numerical optimization problems,” *Applied Soft Computing*, vol. 73, pp. 482–496, 2018.
- [60] Y. Peng, X. Y. Peng, and Z. Q. Liu, “Statistic analysis on parameter efficiency of particle swarm optimization,” *Acta Electronica Sinica*, vol. 32, pp. 209–213, 2004.
- [61] Z. H. Zhan, J. Zhang, Y. Li, and H. S. H. Chung, “Adaptive particle swarm optimization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 39, pp. 1362–1381, 2009.
- [62] S. M. Islam, S. Das, S. Ghosh, S. Roy, and P. N. Suganthan, “An adaptive differential evolution algorithm with novel mutation and crossover strategies for global numerical optimization,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 42, no. 2, pp. 482–500, 2012.
- [63] L. Cui, G. Li, Q. Lin, J. Chen, and N. Lu, “Adaptive differential evolution algorithm with novel mutation strategies in

- multiple sub-populations,” *Computers & Operations Research*, vol. 67, pp. 155–173, 2016.
- [64] D. Datta and J. R. Figueira, “A real-integer-discrete-coded particle swarm optimization for design problems,” *Applied Soft Computing*, vol. 11, no. 4, pp. 3625–3633, 2011.
 - [65] A. Khatami, S. Mirghasemi, A. Khosravi, C. P. Lim, and S. Nahavandi, “A new PSO-based approach to fire flame detection using K-Medoids clustering,” *Expert Systems with Applications*, vol. 68, pp. 69–80, 2017.
 - [66] N. Chouikhi, B. Ammar, N. Rokbani, and A. M. Alimi, “PSO-based analysis of Echo State Network parameters for time series forecasting,” *Applied Soft Computing*, vol. 55, pp. 211–225, 2017.
 - [67] Y. Zhou, N. Wang, and W. Xiang, “Clustering hierarchy protocol in wireless sensor networks using an improved PSO algorithm,” *IEEE Access*, vol. 5, pp. 2241–2253, 2016.
 - [68] L. Luan, L. Tan, and B. Niu, “A novel hybrid global optimization algorithm based on particle swarm optimization and differential evolution,” *Information and Control-Shenyang*, vol. 36, no. 6, p. 708, 2007.
 - [69] M. Lin, Z. Wang, and F. Wang, “Hybrid differential evolution and particle swarm optimization algorithm based on random inertia weight,” in *Proceedings of the 2019 34th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, IEEE, Jinzhou, China, June 2019.
 - [70] M. A. Elaziz and S. Mirjalili, “A hyper-heuristic for improving the initial population of whale optimization algorithm,” *Knowledge-Based Systems*, vol. 172, pp. 42–63, 2019.
 - [71] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.
 - [72] N. H. Awad, M. Z. Ali, J. J. Liang et al., “CEC 2017 special session on single objective numerical optimization single bound constrained real-parameter numerical optimization,” 2017.
 - [73] G. Wu, R. Mallipeddi, and P. N. Suganthan, “Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization,” Technical Report, National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, 2017.
 - [74] G. J. Yue, C. J. Bao, H. T. Xu, and L. Gu, “Optimum design of cylindrical helical compression springs,” *Coal Mine Machinery*, vol. 3, 2012.
 - [75] H. H. Xiao, C. X. Wan, Y. M. Duan, and Q. L. Tan, “Flower pollination algorithm based on gravity search mechanism,” *Acta Automatica Sinica*, vol. 43, pp. 576–594, 2017.
 - [76] E. Sandgren, “Nonlinear integer and discrete programming in mechanical design optimization,” *Journal of Mechanical Design*, vol. 112, no. 2, pp. 223–229, 1990.