axioms

MDPI

Article

Multi-Strategy-Improved Growth Optimizer and Its Applications

Rongxiang Xie [1,2], Liya Yu [3,*], Shaobo Li [2,*], Fengbin Wu [2], Tao Zhang [3] and Panliang Yuan [2]

[1] College of Computer Science and Technology, Guizhou University, Guiyang 550025, China; gs.rxxie22@gzu.edu.cn
[2] State Key Laboratory of Public Big Data, Guizhou University, Guiyang 550025, China; gs.fbwu23@gzu.edu.cn (F.W.); gs.plyuan23@gzu.edu.cn (P.Y.)
[3] College of Mechanical Engineering, Guizhou University, Guiyang 550025, China; gs.tz21@gzu.edu.cn
* Correspondence: lyyu@gzu.edu.cn (L.Y.); lishaobo@gzu.edu.cn (S.L.)

Abstract: The growth optimizer (GO) is a novel metaheuristic algorithm designed to tackle complex optimization problems. Despite its advantages of simplicity and high efficiency, GO often encounters localized stagnation when dealing with discretized, high-dimensional, and multi-constraint problems. To address these issues, this paper proposes an enhanced version of GO called CODGBGO. This algorithm incorporates three strategies to enhance its performance. Firstly, the Circle-OBL initialization strategy is employed to enhance the quality of the initial population. Secondly, an exploration strategy is implemented to improve population diversity and the algorithm's ability to escape local optimum traps. Finally, the exploitation strategy is utilized to enhance the convergence speed and accuracy of the algorithm. To validate the performance of CODGBGO, it is applied to solve the CEC2017, CEC2020, 18 feature selection problems, and 4 real engineering optimization problems. The experiments demonstrate that the novel CODGBGO algorithm effectively addresses the challenges posed by complex optimization problems, offering a promising approach.

Keywords: exploration strategy; exploitation strategy; optimization algorithms; feature selection; realistic engineering optimization

MSC: 65K05

Citation: Xie, R.; Yu, L.; Li, S.; Wu, F.; Zhang, T.; Yuan, P. Multi-Strategy-Improved Growth Optimizer and Its Applications. *Axioms* **2024**, *13*, 361. https://doi.org/10.3390/axioms13060361

Academic Editors: Fevrier Valdez and Giovanni Nastasi

Received: 5 March 2024
Revised: 6 May 2024
Accepted: 23 May 2024
Published: 28 May 2024

1. Introduction

There are many realistic optimization problems in the field of scientific research and engineering applications, and they can almost always be converted into optimization problems for solutions [1,2]. There are many challenges and difficulties in solving these problems, such as nonlinearity, discretization, and high complexity [3,4]. Meta-heuristic algorithms are an effective method for solving this type of problem, which is widely used in real-world optimization problems due to their simplicity and ease of implementation [5,6].

In recent years, researchers have proposed a large number of optimization algorithms to solve realistic optimization problems. In this paper, these algorithms are categorized into four groups: evolution-based algorithms, population-based algorithms, chemistry and physics-based algorithms, and human-based algorithms. Among them, evolution-based algorithms are typically represented by Evolutionary Strategies (ES) [7], Biogeography-Based Optimization (BBO) [8], and Genetic Algorithm (GA) [9]. The algorithms based on population intelligence are Fire Hawk Optimizer (FHO) [10], Bat Algorithm (BA) [11], fox optimizer [12], Golden Jackal Optimization (GJO) [13]. Algorithms based on chemistry and physics have Gravitational Search Algorithm (GSA) [14], Big Bang-Big Crunch algorithm (BB-BC) [15], Simulated Annealing (SA) [16], Magnetic Optimization Algorithms (MOA) [17], Water Evaporation Optimization (WEO) [18], Atom Search Optimization (ASO) [19]. Human-based algorithms have search and rescue optimization [20], Human Mental Search (HMS) [21], arithmetic optimization algorithm (AOA) [22]. However, as

optimization problems become highly complex, these aforementioned algorithms have some drawbacks in solving real-world optimization problems. For instance, there are problems such as getting stuck in localized stagnation and poor convergence performance. Therefore, optimization algorithms with improved strategies are widely used to solve complex real-world optimization problems.

Several researchers have proposed enhanced optimization algorithms to tackle specific engineering design challenges. A.M. Shaheen et al. introduced the enhanced equilibrium optimization algorithm (IEOA) for power distribution network configuration, achieving optimal allocation of distributed generators [23]. Bahaeddin Turkoglu et al. introduced a binary artificial algae algorithm to enhance feature selection and classification accuracy [24]. Gang Hu developed an enhanced variant of the black widow optimization algorithm for feature selection, yielding notable performance outcomes [25]. Min Xu proposed a novel binary arithmetic optimization algorithm (BAOA) that outperformed competitors in benchmark datasets for feature selection [26]. Gang Hu et al. proposed an enhanced hybrid arithmetic optimization algorithm named CSOAOA for solving engineering problems and demonstrated its utility in solving real optimization problems [27].

Although existing heuristic algorithms have made good progress in solving optimization problems, the number of combinations that algorithms need to search increases exponentially as the dimensionality of the optimization problem expands and the complexity of constraints increases. Existing algorithms often face the problem of premature convergence, meaning that they may stop at suboptimal solution sets before reaching the global optimum. The fundamental reason for this is that the exploration and exploitation performance of the algorithms is weak, which limits their effectiveness in capturing the intrinsic patterns and features of the actual optimization problem data. Therefore, it is necessary to explore a new, suitable, and efficient metaheuristic algorithm that can fully explore the search space during the optimization process in order to better mitigate the challenges brought by high-dimensional problems. Fortunately, the inspiration for the growth optimizer (GO) comes from the learning process of individuals in social growth, and it has been proven to be a robust tool with high exploration capability [28]. Relevant studies have shown that GO has strong exploration capability and application scalability. For example, GO has been successfully applied to the parameter identification of solar photovoltaic cells [29,30], multi-level threshold image segmentation and wireless sensor network node deployment [31], and enhancing intrusion detection systems in the internet of things and cloud environments [32].

To the best of our knowledge, existing papers have not attempted to propose an improved GO applicable to both continuous and discrete optimization problems. Considering that GO may fall into local optima when solving high-dimensional complex optimization problems, this paper proposes an enhanced growth optimizer (CODGBGO) by combining three strategies. Among them, the Circle-OBL initialization strategy is used to generate initial solutions with good distributions. Secondly, the exploration strategy expands the search space by self-learning in a region of radius R and learning the differences among individuals, which in turn enhances the exploration ability of the algorithm and its ability to jump out of the local optimum trap. Finally, the exploitation strategy greatly improves the convergence speed and convergence accuracy of the algorithm through the guidance of optimal individuals. The main contributions of this paper are summarized as follows:

- Using the Circle-OBL initialization strategy for initializing better-distributed populations makes the performance of the algorithm improve.
- The exploration strategy improves the global exploration performance of the algorithm by learning self-knowledge over a radius R and improving population diversity by learning the differences between individuals.
- The exploitation strategy leads to an improvement in the convergence speed and convergence accuracy of the algorithm through the bootstrapping of the optimal individuals.

- CODGBGO is proposed by combining the above strategies, and it is confirmed to be a promising optimization method by numerical optimization, feature selection, and engineering optimization.

The subsequent work is structured as follows: Section 2 presents the fundamental theory of primitive GO. In Section 3, CODGBGO is introduced as a novel approach. Subsequently, in Section 4, we apply the proposed CODGBGO to solve various problem sets, including the CEC2017 [33], and IEEE CEC2020 [34], as well as addressing 18 feature selection problems and 4 mechanical design problems. Section 5 provides the conclusions and future directions.

2. The Theory of Growth Optimizer

Inspired by the process of personal growth, GO was introduced in 2022 as a novel optimizer [28]. Learning and reflection are two key stages of GO; they complement each other and foster individual growth. Learning is to draw knowledge from the gaps between different individuals, and reflecting is to improve one's knowledge by summarizing one's strengths and weaknesses.

2.1. Learning Stage

In the learning stage, growth resistance (GR) is used to indicate the amount of knowledge that an individual has learned. Assuming that the individual is represented as $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$, where $x_{i,D}$ represents the Dth dimension knowledge of the ith individual. The value of GR_i is taken as the value of the objective function of the individual x_i, where a larger GR_i indicates that the individual has learned a lesser amount of knowledge and vice versa. The individual learns by examining the gaps between other individuals in the population and growing in the process. The gaps between the leader and the elite (Gap_1), the leader and the bottom (Gap_2), the elite and the bottom (Gap_3), and the gap between two random individuals (Gap_4) are mainly modeled in the learning stage. Each gap is described by the following equation:

$$\begin{cases} Gap_1 = x_{best} - x_{better} \\ Gap_2 = x_{best} - x_{worse} \\ Gap_3 = x_{better} - x_{worse} \\ Gap_4 = x_{L1} - x_{L2} \end{cases} \tag{1}$$

where x_{best} represents the leader of the society, x_{better} represents one of the $P_1 - 1$ optimal individuals, referred to as the elite, x_{worse} represents one of the bottom P_1 individuals, where P_1 takes the value 5, x_{L1} and x_{L2} denote two different random individuals. $Gap_k (k = 1, 2, 3, 4)$ represents the gap between two individuals of different types; learners learn from and benefit from the gap Gap_k.

Generally, individuals will tend to learn from information that has a large knowledge gap. The learning factor (LF) is introduced for each group of knowledge gaps, and the learning factor LF_k will affect an individual's learning efficiency for the kth group of knowledge gaps, expressed as the following equation:

$$LF_k = \frac{\|Gap_k\|}{\sum_{k=1}^{4} \|Gap_k\|}, (k = 1, 2, 3, 4) \tag{2}$$

where LF_k represents the kth group knowledge gap normalization and its value belongs to the range [0, 1]. The larger LF_k indicates the more efficiently the individual learns from the kth group knowledge gap, and vice versa.

The extent of knowledge learning varies across individuals, where SF_i is used to denote the extent of learning of the *ith* individual, expressed as the following equation.

$$SF_i = \frac{GR_i}{GR_{max}} \tag{3}$$

where GR_i represents the resistance to growth of the *ith* individual and GR_{max} represents the maximum resistance to growth. A larger SF_i indicates that the individual has a greater range of knowledge to learn and tends to perform the exploration phase and, conversely, tends to perform the exploitation phase.

Individuals learn by learning from individual knowledge gaps to promote their growth. For example, the amount of knowledge acquired by the *ith* individual through learning from the *kth* group of knowledge gaps is denoted as KA_k using the following equation:

$$KA_k = SF_i \cdot LF_k \cdot Gap_k, (k = 1, 2, 3, 4) \tag{4}$$

The specific learning process of the *ith* individual is expressed by the following equation:

$$x_i^{new} = x_i^{It} + KA_1 + KA_2 + KA_3 + KA_4 \tag{5}$$

where It represents the current number of iterations and x_i^{new} represents the new state of the *ith* individual after learning.

By adjusting the individuals during the learning stage, the quality of the individuals is likely to improve, but it is also likely to regress; therefore, for the individuals whose quality has improved, the retention operation is performed. For individuals whose quality regresses, they are retained with probability P_2, and those that exceed the probability P_2 are discarded. This process is described using the following equation:

$$x_i^{It+1} = \begin{cases} x_i^{new} & if\ f\left(x_i^{new}\right) < f\left(x_i^{It}\right) \\ \begin{cases} x_i^{new} & if\ r_1 < p_2 \\ x_i^{It} & else \end{cases} & else \end{cases} \tag{6}$$

where r_1 denotes a random number in the range [0, 1] and P_2 takes the value 0.001. $f\left(x_i^{new}\right)$ denotes the objective function value after learning for the *ith* individual.

2.2. Reflection Stage

The reflection stage is mainly a stage in which the individual retains his good aspects by reflecting on his deficiencies and, for the bad aspects, learns from the good individuals. For the aspects that cannot be remedied, it is necessary to abandon the previous knowledge and learn systematically again. The reflexive stage of GO is expressed using the following formula:

$$x_{i,j}^{It+1} = \begin{cases} \begin{cases} lb + r_4 \times (ub - lb) & if r_3 < AF \\ x_{i,j}^{It} + r_5 \times (R_j - x_{i,j}^{It}) & else \end{cases} & if\ r_2 < P_3 \\ x_{i,j}^{It} & else \end{cases} \tag{7}$$

$$AF = 0.01 + 0.99 \times \left(1 - \frac{FEs}{MaxFEs}\right) \tag{8}$$

where lb and ub are the lower and upper bounds of the problem, respectively, r_2, r_3, r_4, r_5 represents pseudorandom numbers in the range [0, 1], P_3 represents the probability of reflection, which takes the value of 0.3 in this paper, AF represents the attenuation factor, FEs represents the current number of function evaluations, and $MaxFEs$ represents the maximal number of function evaluations, and AF decreases linearly from 1 to 0.01 as FEs increases. R_j denotes the *jth* dimension information of the leader individual and the elite individual, which means that in the reflection stage, if *ith* individual needs to learn from

the *jth* dimension of the other excellent individuals, there will be an upper-level individual to guide it.

2.3. Implementation of GO for Optimization

In this subsection, a detailed implementation of GO is given. Algorithm 1 gives the pseudo-code of GO. The implementation steps of the GO algorithm are given below:

Step 1: Initialize the run parameters containing the population size (N), the dimension of the problem to be solved (D), ub and lb of the problem to be solved, and the parameters P_1, P_2, P_3.

Step 2: Initialize the population and calculate the fitness function value based on the initialization parameters. Each individual is a $1 \times D$ vector. The whole population is an $N \times D$ matrix.

Step 3: Enter the main loop. Sort the individuals in the population by GR to obtain the optimal solution x_{best}, x_{best} will be updated in each iteration.

Step 4: Execute the learning stage by first selecting the elite individual x_{better}, the bottom individual x_{worse}, and the random individuals x_{L1} and x_{L2}, and sequentially using Equations (1)–(6) to update the position of *ith* individual and update the globally optimal $gbestx$ in a timely manner. The number of function evaluations FEs is added to 1.

Step 5: Execute the reflection phase by updating the *jth* dimension of the *ith* individual according to Equations (7) and (8), and then use Equation (6) to update the *ith* individual position and update the global optimal solution $gbestx$ in a timely manner. The number of function evaluations FEs is added to 1.

Step 6: If the loop terminates, return the globally optimal solution $gbestx$, otherwise go to step 3 to continue execution.

Algorithm 1: The pseudocode of the GO algorithm

Input: N, D, ub, lb, $P_1 = 5$, $P_2 = 0.001$, $P_3 = 0.3$, $FEs = 0$
Output: global optimum solution: $gbestx$
Initialize the population foundation $x_i = lb + (ub - lb) \cdot rand(N \times D)$ and calculate the value of the objective function
while $FEs <= MaxFEs$ *do*
    $[\sim, ind] = sort(GR)$
    $x_{best} = x(ind(1),:)$
    %Learning stage:
    for $i = 1 : N$ *do*
        $x_{better} = x(ind(randi([2, 1 + P_1], 1)),:)$
        $x_{worse} = x(ind(randi([N - P_1, N], 1)),:)$
        Find two random individuals that are different from x_i, x_{L1} and x_{L2}
        Calculate Gap_k, k = 1, 2, 3, 4 by Equation (1)
        Calculate LF_k, k = 1, 2, 3, 4 foundation Equation (2)
        Calculate SF_i foundation Equation (3)
        Calculate KA_k, k = 1, 2, 3, 4 foundation Equation (4)
        Calculate the learning process for the ith individual once foundation Equation (5)
        Calculate the update of the ith individual foundation Equation (6)
        Timely updates $gbestx$
        $FEs = FEs + 1$
    end
    %Reflection stage:
    for $i = 1 : N$ *do*
        Calculate the reflection process for the ith individual once foundation Equation (7) and Equation (8)
        Calculate the update of the ith individual foundation Equation (6)
        Timely updates $gbestx$
        $FEs = FEs + 1$
    end
end
Output $gbestx$

3. A Multi-Strategy Enhanced Growth Optimizer

The original GO has the advantages of faster convergence and simpler structure. However, when dealing with complex high-dimensional optimization problems, there is a lack of population diversity in the iterative process of the algorithm, which leads to the problem that the algorithm tends to be easy to trap in the local optimum. At the same time, there are deficiencies in the exploitation capability, resulting in the loss of convergence accuracy and convergence speed. In this work, the Circle-OBL initialization strategy, exploration strategy, and exploitation strategy are introduced into the original GO to form CODGBGO. In CODGBGO, firstly, the Circle-OBL initialization strategy is used to generate a well-initialized population to improve the overall quality of the population. Secondly, the exploration strategy is suggested in the learning stage to improve the diversity of the population; this improves the ability of the algorithm to escape from the local optimization

trap. Finally, the exploitation strategy is employed in the reflection stage to improve the performance of the algorithm to exploitation, speed up the convergence speed of the algorithm, and improve the convergence accuracy.

3.1. Circle-OBL Initialization Strategy

Literature [35] indicates that the improvement of the initialization scheme using chaotic mapping and OBL strategies can generate initial populations with better solution quality, leading to better optimization results. Among them, chaotic mapping is used to address the problem of premature convergence by generating initial solutions with higher diversity levels.The OBL strategy aims to accelerate the convergence of the algorithm by exploring a wider region of the solution space during the initialization process. Therefore, in this subsection, the population will be initialized using the Circle-OBL initialization strategy, where the mathematical expression for the chaotic circle mapping is given by Equation (9) [36].

$$z_{k+1} = z_k + a - mod(\frac{b}{2\pi} sin(2\pi z_k), 1) \tag{9}$$

According to the literature [36], where $a = 0.5$, $b = 0.2$, $z_1 = rand$, mod represents the modulo operation, sequences are more diverse after circle chaotic mapping. Figure 1a,b show the sequence distribution after 500 iterations for pseudo-random mapping and circle chaotic mapping, respectively, and it can be seen that circle chaotic mapping sequences have better traversal. Subsequently, the individuals are initialized using the generated circle chaotic mapping sequences, and at the same time, they are subjected to opposing learning, which is represented by the following equation:

$$\begin{cases} x_i = z \odot (ub - lb) + lb \\ x_i^* = rand(1, D) \odot (ub + lb) - x_i \end{cases} \quad (i = 1, 2, \ldots, N) \tag{10}$$

where x_i^* represents the opposing individual of ith individual. Compose N chaotic individuals into a chaotic population P_4 and N opposing individuals into an opposing population P_4^*, expressed as Equation (11).

$$\begin{cases} P_4 = (x_1;\ x_2; \ldots; x_N) \\ P_4^* = (x_1^*;\ x_2^*; \ldots; x_N^*) \end{cases} \tag{11}$$

Combining the two populations to form $P = \{P_4 \cup P_4^*\}$, and taking the top N individuals with the best objective function values to form the final initialized population.

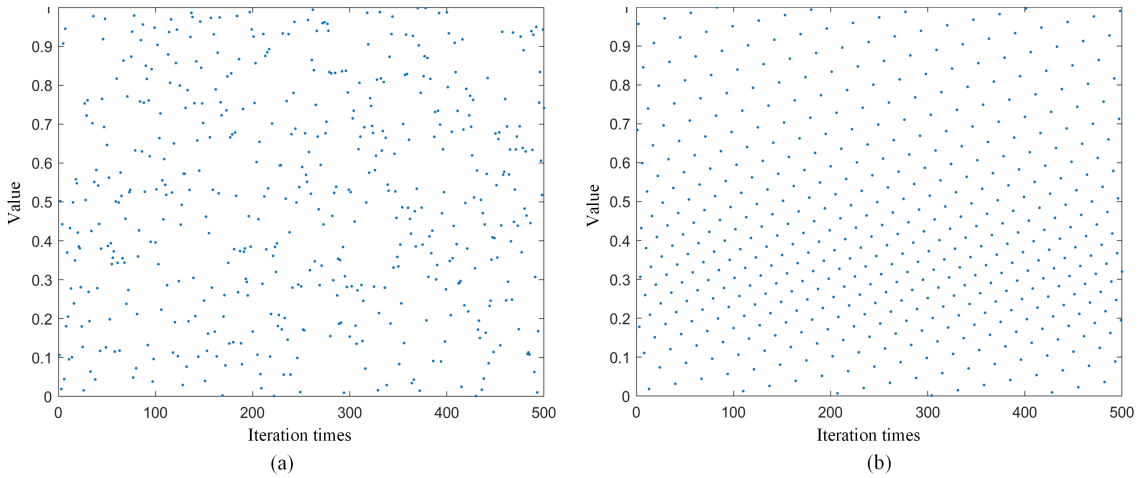

Figure 1. Sequence distribution map. (**a**) Random mapping (**b**) Circle chaotic mapping.

3.2. Exploration Strategy

To address the problem that the original GO is prone to falling into the local optimum trap due to the lack of population diversity at the late stage of the algorithm iteration, the algorithm is easy to fall into the local optimum trap. In this section, the exploration strategy is proposed to enhance population diversity and then improve the ability of the algorithm to jump out of the local optimal trap. The strategy mainly contains the differential strategy part and the self-search strategy part.

Literature [37] indicates that the differential strategy can effectively enhance the algorithm's global search capability as well as its ability to jump out of the local optimum trap. Inspired by this, the difference strategy is introduced into the algorithm in this section to enhance its performance. Here, our main consideration is to learn from the knowledge gap between the current individual and a random individual and the knowledge gap between two random individuals, and the two sets of knowledge gaps are expressed using the following equation:

$$\begin{cases} DE_{k1/i} = x_{k1} - x_i^{It} \\ DE_{k2/k3} = x_{k2} - x_{k3} \end{cases} \tag{12}$$

where x_{k1}, x_{k2}, x_{k3} denote three mutually exclusive individuals in the population, x_i^{It} denotes the *ith* individual, $DE_{k1/i}$ denotes the knowledge gap between the current individual and a random individual, and $DE_{k2/k3}$ denotes the knowledge gap between two random individuals. Subsequently, the *ith* individual learns from the two sets of gaps; here, the *ith* individual learns knowledge equally from both sets of knowledge gaps. This process is represented using the following equation:

$$x_i^{new} = x_i^{It} + 0.5 \cdot DE_{k1/i} + 0.5 \cdot DE_{k2/k3} \tag{13}$$

Literature [38] indicates that individuals can effectively enhance the global search performance of an algorithm by learning within a certain range. Inspired by this, a self-learning strategy is proposed in this subsection to enhance the algorithm's performance. A self-search strategy is the process by which an individual improves himself by learning knowledge within a radius of R. In the selection of R, the wider distribution of normally distributed random numbers is considered suitable to be used as a learning radius, allowing the global search ability to be strengthened. At the same time, considering that in the iterative process, a larger search radius will also cause the loss of convergence speed, multiplying a tail term on the basis of a normally distributed random number makes a balance in each function. Hence, R is expressed using the following equation:

$$R = N(0,1) \cdot \pi/8 \tag{14}$$

where $N(0,1)$ denotes a random number obeying a normal distribution, $\pi/8$ as a tail term to balance global search with local search. The self-search process for the *ith* individual is expressed using the following equation:

$$x_i^{new} = x_i^{It} \cdot (1 + R) \tag{15}$$

The equal combination of the differential strategy and the self-search strategy forms the exploration strategy. The process is shown in Figure 2, the green line represents the simulation process of the exploration strategy. From the figure, it can be seen that through the self-learning strategy and the differential strategy, the individual makes it jump out of the local optimal region by learning in the region with radius R. The learning of the local optimal region can be seen by the self-learning strategy and the difference strategy. At the same time, the individual learns from the differences among other individuals, which improves their global search ability. Meanwhile, it can be seen that the population exploration space

has improved. Through the above analysis, it is confirmed that the exploration strategy enhances the algorithm's global search ability. expressed as the following equation:

$$x_i^{new} = \begin{cases} x_i^{It} + 0.5 \cdot DE_{k1/i} + 0.5 \cdot DE_{k2/k3} & if\ rand < 0.5 \\ x_i^{It} \cdot (1 + R) & else \end{cases} \tag{16}$$

The new state of *ith* individual is subsequently retained using elite retention, expressed as the following equation:

$$x_i^{It+1} = \begin{cases} x_i^{new} & if\ f(x_i^{new}) < f(x_i^{It}) \\ x_i^{It} & else \end{cases} \tag{17}$$

where $f(x_i^{new})$ denotes the objective function value after exploration strategy for the *ith* individual.

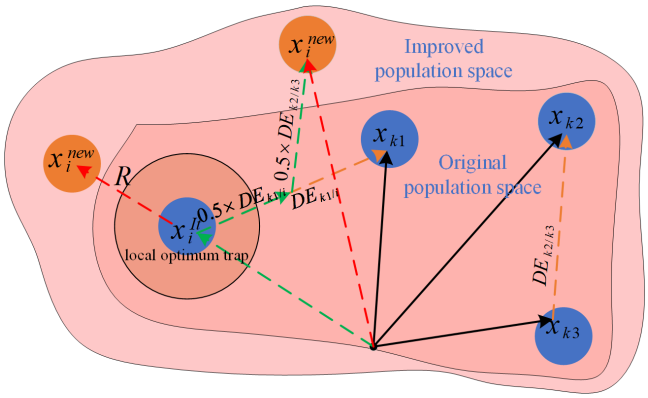

Figure 2. The process of exploration strategy.

3.3. Exploitation Strategy

In the literature [39], it is stated that using optimal individuals for bootstrapping can effectively enhance algorithm exploitation and improve the convergence speed. Inspired by this, an exploitation strategy based on optimal individual bootstrapping is proposed in this section, and at the same time, taking into account the situation of falling into a local optimum, the differential strategy is also taken into account in order to balance the exploitation phase and the exploration phase, aiming to enhance the algorithm development capability as much as possible, where the gap is represented by the following equation:

$$Gap_{k4/i} = x_{k4} - x_i^{It} \tag{18}$$

where x_{k4} denotes a random individual, $Gap_{k4/i}$ denotes the gap between the current individual and the randomized individual. Also, in order to enhance the algorithm exploitation, the globally optimal individual is used to guide the position update of the current individual; this process is shown in Figure 3. From the figure, the individual quickly approaches the globally optimal individual, and at the same time, due to the consideration of the differential idea, the individual avoids falling into the local optimal region, which improves the convergence speed of the algorithm as well as the practical exploitation ability. The process is expressed using the following equation:

$$x_i^{new} = x_{best} + rand \cdot Gap_{k4/i} \tag{19}$$

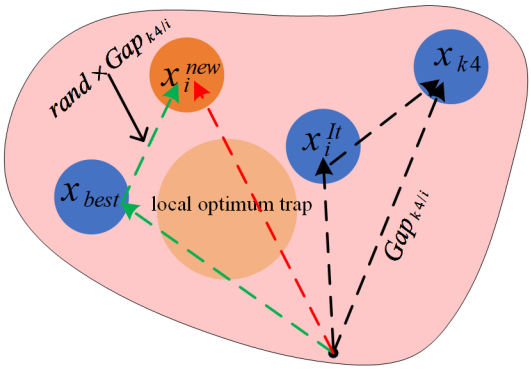

Figure 3. The process of exploitation strategy.

3.4. Implementation of CODGBGO for Optimization

In this subsection, a detailed implementation of CODGBGO is given. Algorithm 2 gives the pseudo-code of CODGBGO. The implementation steps of the CODGBGO algorithm are given below:

Step 1: Initialize the run parameters containing N, D, ub and lb, and the parameters P_1, P_2, P_3, α, β.

Step 2: Use the Circle-OBL initialization strategy. Initialize the population according to Equations (10) and (11), and calculate the objective function value based on the initialization parameters. Each individual is a $1 \times D$ vector. The whole population is an $N \times D$ matrix.

Step 3: Enter the main loop. Sort the individuals in the population by GR to obtain the optimal solution x_{best}. x_{best} will be updated in each iteration.

Step 4: Execute the learning stage, if $rand < \alpha$, by first selecting the elite individual x_{better}, the bottom individual x_{worse}, and the random individuals x_{L1} and x_{L2}, then sequentially using Equations (1)–(6) to update the position of the ith individual, otherwise using Equations (12)–(17) to update the position of the ith individual, and updating the globally optimal node $gbestx$ in a timely manner. The number of function evaluations FEs is added to 1.

Step 5: Execute the reflection stage, if $rand < \beta$, by updating the jth dimension of the ith individual according to Equations (7) and (8), and then use Equation (6) to update the individual position; otherwise update the position of the ith individual according to Equations (17)–(19), and update the global optimal solution $gbestx$ in a timely manner. The number of function evaluations FEs is added to 1.

Step 6: If the loop terminates, return the globally optimal solution $gbestx$; otherwise, go to step 3 to continue execution.

3.5. Compytional Complexity

This section analyzes the computational complexity of the proposed CODGBGO algorithm. The computational complexity of the original GO initialization is $O(N)$, In each iteration, each member of the population undergoes two stages of updating and evaluation of its objective function. The computational complexity of the update process is $O(2{\cdot}T{\cdot}N{\cdot}D)$, where T is the maximum number of iterations. Therefore, the computational complexity of GO is $O(N{\cdot}(1 + 2T{\cdot}D))$. Compared to GO, CODGBGO first introduces the Circle-OBL initialization strategy in the initialization process, so the computational complexity of initialization is $O(2N)$. The introduction of exploration and exploitation strategies does not change the original GO logic, so the computational complexity of the update process remains $O(2{\cdot}T{\cdot}N{\cdot}D)$. Therefore, the computational complexity of CODGBGO is $O(2N{\cdot}(1 + T{\cdot}D))$.

Algorithm 2: The pseudocode of the CODGBGO algorithm

Input: N, D, ub, lb, $P_1 = 5$, $P_2 = 0.001$, $P_3 = 0.3$, α, β, $FEs = 0$
Output: global optimum solution: $gbestx$

Initialize the population x refer to Equations (9)–(11), and calculate the value of the objective function
while $FEs <= MaxFEs$ *do*
    $[\sim, ind] = sort(GR)$
    $x_{best} = x(ind(1),:)$
    %Learning stage:
    for $i = 1 : N$ *do*
        if $rand < \alpha$
            $x_{better} = x(ind(randi([2, 1+P_1], 1)),:)$
            $x_{worse} = x(ind(randi([N-P_1, N], 1)),:)$
            Find two random individuals that are different from x_i, x_{L1} and x_{L2}
            Calculate Gap_k, k = 1, 2, 3, 4 by Equation (1)
            Calculate LF_k, k = 1, 2, 3, 4 foundation Equation (2)
            Calculate SF_i foundation Equation (3)
            Calculate KA_k, k = 1, 2, 3, 4 foundation Equation (4)
            Calculate the learning process for the ith individual once foundation Equation (5)
            Calculate the update of the ith individual foundation Equation (6)
        else
            Calculate the learning process for the ith individual once foundation Equations (12)–(16)
            Calculate the update of the ith individual foundation Equation (17)
        end if
        Timely updates $gbestx$
        $FEs = FEs + 1$
    end
    %Reflection phase:
    for $i = 1 : N$ do
        if $rand < \beta$
            Calculate the reflection process for the ith individual once foundation Equations (7) and (8)
            Calculate the update of the ith individual foundation Equation (6)
        else
            Calculate the reflection process for the ith individual once foundation Equations (18) and (19)
            Calculate the update of the ith individual foundation Equation (17)
        end if
        Timely updates $gbestx$
        $FEs = FEs + 1$
    end
end
Output $gbestx$

4. Experimental Results

In this section, we will conduct a series of experiments to evaluate the performance of the proposed CODGBGO. First, its performance in solving numerical optimization problems is evaluated using the CEC2017 test set and the CEC2020 test set. Second, its performance in solving discretization problems is evaluated using 18 feature selection problems. Finally, four constrained engineering applications are solved using the proposed CODGBGO to evaluate its performance in solving realistic constrained optimization problems. Meanwhile, in order to comprehensively and objectively evaluate the solution performance of CODGBGO, the proposed CODGBGO is compared with numerous algorithms. The compared algorithms include classical algorithms, improved algorithms, high citation algorithms, popular algorithms, new algorithms, and superior algorithms.

To ensure the fairness of the experiments, the population size was set to 60, and the maximum number of function evaluations was set to 60,000. The test dimensions were set to 30D, 50D, and 100D for CEC2017, 10D, and 20D for CEC2020. All experiments were performed using Windows 11 as the operating system, and the code execution took place within the MATLAB R2021b environment.

4.1. Comparison Algorithm and CEC Benchmark Problems

In order to comprehensively and objectively evaluate the performance of the proposed CODGBGO, this paper compares CODGBGO with several existing optimization algorithms on several sets of test functions. Among them, the test function sets selected in this paper are IEEE CEC2017 and IEEE CEC2020, with detailed information as shown in Tables 1 and 2, involving unimodal functions, multimodal functions, hybrid functions, and composition functions. The unimodal function contains only one local optimal solution and is mainly used to verify the local search capability of CODGBGO. The complexity of multimodal, hybrid, and composition functions is higher than that of unimodal functions, and due to the inclusion of multiple local optimal solutions, they are easy to fall into the local optimal trap in the solution process. Therefore, the tests on multimodal, hybrid, and synthetic functions are mainly used to verify the performance of CODGBGO in solving complex problems, to check the ability to jump out of local optimums, and to evaluate the balance between the global search phase and the local search phase. Meanwhile, in order to comprehensively reflect the performance of CODGBGO, 21 comparison algorithms are selected for experimental comparison in this paper. These include classical algorithms, improved algorithms, high citation algorithms, popular algorithms, new algorithms, and superior algorithms. The specific parameter configurations are shown in Table 3.

Table 1. Information about the IEEE CEC2017 function test set.

Index	Types	Name	Optimum
CEC2017_F1	Unimodal	Shifted and Rotated Bent Cigar Function	100
CEC2017_F3		Shifted and Rotated Zakharov Function	300
CEC2017_F4	Multimodal	Shifted and Rotated Rosenbrock's Function	400
CEC2017_F5		Shifted and Rotated Rastrigin's Function	500
CEC2017_F6		Shifted and Rotated Expanded Scaffer's F6 Function	600
CEC2017_F7		Shifted and Rotated Lunacek Bi-Rastrigin Function	700
CEC2017_F8		Shifted and Rotated Non-Continuous Rastrigin's Function	800
CEC2017_F9		Shifted and Rotated Lévy Function	900
CEC2017_F10		Shifted and Rotated Schwefel's Function	1000

Table 1. *Cont.*

Index	Types	Name	Optimum
CEC2017_F11	Hybrid	Hybrid function 1 (N = 3)	1100
CEC2017_F12		Hybrid function 1 (N = 3)	1200
CEC2017_F13		Hybrid function 3 (N = 3)	1300
CEC2017_F14		Hybrid function 4 (N = 4)	1400
CEC2017_F15		Hybrid function 5 (N = 4)	1500
CEC2017_F16		Hybrid function 6 (N = 4)	1600
CEC2017_F17		Hybrid function 6 (N = 5)	1700
CEC2017_F18		Hybrid function 6 (N = 5)	1800
CEC2017_F19		Hybrid function 6 (N = 5)	1900
CEC2017_F20		Hybrid function 6 (N = 6)	2000
CEC2017_F21	Composition	Composition function 1 (N = 3)	2100
CEC2017_F22		Composition function 2 (N = 3)	2200
CEC2017_F23		Composition function 3 (N = 4)	2300
CEC2017_F24		Composition function 4 (N = 4)	2400
CEC2017_F25		Composition function 5 (N = 5)	2500
CEC2017_F26		Composition function 6 (N = 5)	2600
CEC2017_F27		Composition function 7 (N = 6)	2700
CEC2017_F28		Composition function 8 (N = 6)	2800
CEC2017_F29		Composition function 9 (N = 3)	2900
CEC2017_F30		Composition function 10 (N = 3)	3000
Search range: [−100,100]			

Table 2. Information about the IEEE CEC2020 function test set.

Index	Types	Name	Optimum
CEC2020_F1	Unimodal	Shifted and Rotated Bent Cigar Function	100
CEC2020_F2	Multimodal	Shifted and Rotated Schwefel's Function	1100
CEC2020_F3		Shifted and Rotated Lunacek bi-Rastrigin Function	700
CEC2020_F4		Expanded Rosenbrock's plus Griewangk's Function	1900
CEC2020_F5	Hybrid	Hybrid Function 1 (N = 3)	1700
CEC2020_F6		Hybrid Function 2 (N = 4)	1600
CEC2020_F7		Hybrid Function 3 (N = 5)	2100
CEC2020_F8	Composition	Composition Function 1 (N = 3)	2200
CEC2020_F9		Composition Function 2 (N = 4)	2400
CEC2020_F10		Composition Function 3 (N = 5)	2500
Search range: [−100, 100]			

Table 3. The comparison algorithm details and parameter settings.

Algorithms	Proposed Time	Parameters Settings	Citations	Type
Particle Swarm Optimization (PSO) [40]	1995	$w = 1$, $w_p = 0.99$, $c_1 = 1.5$, $c_2 = 2.0$	81,387	classical
Differential Evolution (DE) [41]	1997	$F = 0.5$, $CR = 0.9$	33,925	classical
Comprehensive Learning Particle Swarm Optimizer (CLPSO) [42]	2006	$w_{min} = 0.2$, $w_{max} = 0.9$, $c = 1.496$	2838	Improved
Bezier Search Differential Evolution (BESD) [43]	2021	$K = 5$	40	Improved
Bernstein-levy Differential Evolution (BDE) [44]	2023	*No parameters*	3	Improved
Fractional Order Particle Swarm Optimization (FOPSO) [45]	2022	$\lambda = 0.1$	18	Improved
Improved Chaotic Grey Wolf Optimizer (ICGWO) [46]	2023	*No parameters*	15	Improved

Table 3. *Cont.*

Algorithms	Proposed Time	Parameters Settings	Citations	Type
Teaching Learning-Based Optimization (TLBO) [47]	2011	$T_F = 1$ *or* 2	3438	high citation
Grey Wolf Optimizer (GWO) [48]	2014	$\alpha = 2 - 2{\cdot}(FEs - MaxFEs)$	11,102	high citation
Multi-Verse Optimizer (MVO) [49]	2016	$WEP_{Max} = 1,\ WEP_{Min} = 0.2$	2094	high citation
Whale Optimization Algorithm (WOA) [11]	2016	$b = 1,\ a_1 = 2 - (2{\cdot}FEs/MaxFEs), a_2 = -1 - (FEs/MaxFEs)$	7672	high citation
Salp Swarm Algorithm (SSA) [50]	2017	$c_1 = 2{\cdot}exp((4{\cdot}FEs/MaxFEs)^2)$	3284	high citation
Harris Hawks Optimization (HHO) [51]	2019	$E_0 = 2{\cdot}rand - 1,\ E_1 = 2 - 2{\cdot}(FEs/MaxFEs)$	3048	high citation
Artificial Butterfly Optimization (ABO) [52]	2017	$ratio_e = 0.2,\ step_e = 0.05$	67	popular
Butterfly Optimization Algorithm (BOA) [53]	2019	$p = 0.8,\ \alpha = 0.1, c = 0.01$	813	popular
Equilibrium Optimizer (EO) [54]	2020	$V = 1,\ a_1 = 2,\ a_2 = 1,\ GP = 0.5$	1177	popular
Dung Beetle Optimizer (DBO) [55]	2023	$k\ and\ \lambda = 0.1, b = 0.3, S = 0.5$	38	new
INFO [56]	2022	$I = randi([2,5])$	258	new
Snake Optimizer (SO) [57]	2022	$T_1 = 0.25,\ T_2 = 0.6,\ C_1 = 0.5,\ C_2 = 0.05,\ C_3 = 2$	198	new
Growth Optimizer (GO)	2023	$p_1 = 5,\ p_2 = 0.001,\ p_3 = 0.3$	11	new
LSHADE [58]	2014	$NP_{init} = 18{\cdot}D,\ NP_{min} = 4,\ \lvert A\rvert = 2.6{\cdot}NP, p = 0.11, H = 6$	671	superior
Improved Multi-Operator Differential Evolution (IMODE) [59]	2020	$NP_{init} = 6{\cdot}D^2,\ NP_{min} = 4,\ \lvert A\rvert = 2.6, H = 20{\cdot}D$	106	superior
Adaptive L-SHADE algorithm (ALSHADE) [60]	2022	$NP_{init} = 18{\cdot}D,\ NP_{min} = 4,\ \lvert A\rvert = 2.6{\cdot}NP, p = 0.11, e = 0.5$	6	superior

4.2. Effectiveness of Strategies and Parameter Settings

4.2.1. Effectiveness of Strategies

In this subsection, the effectiveness of each of the added strategies is mainly tested. Among them, CODGBGO is a new optimization algorithm formed by integrating the Circle-OBL initialization strategy, the exploration strategy, and the exploitation strategy on the basis of GO. In order to verify the effectiveness of each strategy, each strategy is individually integrated into GO to form a new combination. Among them, the Circle-OBL initialization strategy is integrated into GO to form COGO, the exploration strategy is integrated into GO to form DGGO, and the exploitation strategy is integrated into GO to form BGO. These combinations were evaluated on the CEC2017 test function set with 30D dimensions, and each experiment was run independently for 30 times. The experimental results were ranked using the Friedman mean rank test to verify the effectiveness of the strategies. The experimental results are shown in Table 4. From Table 4, it can be observed that the rankings of COGO, DGGO, and BGO are all superior to GO. This indicates that the introduction of each strategy is beneficial for enhancing the performance of GO. Moreover, combining the three strategies into GO yields a more effective enhancement of its performance.

Table 4. Friedman mean rank test results for different combinations of strategies.

Problems	GO	COGO	DGGO	BGO	CODGBGO
CEC2017_F1	3.53	3.43	2.23	2.93	2.87
CEC2017_F3	4.37	3.67	1.53	3.97	1.47
CEC2017_F4	3.13	2.90	3.27	3.30	2.40
CEC2017_F5	3.77	4.30	3.90	1.33	1.70
CEC2017_F6	2.23	2.40	2.80	3.83	3.73
CEC2017_F7	4.07	4.20	3.67	1.50	1.57
CEC2017_F8	3.87	3.97	4.03	1.33	1.80
CEC2017_F9	2.53	2.47	1.93	4.40	3.67
CEC2017_F10	3.70	4.17	4.13	1.53	1.47
CEC2017_F11	3.73	3.43	3.43	2.40	2.00
CEC2017_F12	4.10	2.90	2.23	2.97	2.80
CEC2017_F13	3.60	2.87	2.40	4.03	2.10
CEC2017_F14	4.03	3.60	2.50	2.97	1.90
CEC2017_F15	3.50	3.43	2.67	3.30	2.10
CEC2017_F16	3.83	2.90	3.60	2.80	1.87
CEC2017_F17	3.60	3.10	3.43	2.53	2.33
CEC2017_F18	4.23	4.00	1.47	3.57	1.73
CEC2017_F19	4.00	3.80	2.60	3.03	1.57
CEC2017_F20	3.47	2.90	3.43	2.83	2.37
CEC2017_F21	3.90	4.27	3.83	1.53	1.47
CEC2017_F22	4.00	3.13	2.53	3.87	1.47
CEC2017_F23	3.80	3.90	3.63	1.97	1.70
CEC2017_F24	4.00	4.30	3.57	1.73	1.40
CEC2017_F25	3.03	2.20	3.20	3.10	3.47
CEC2017_F26	3.63	3.43	3.13	2.57	2.23
CEC2017_F27	2.27	2.93	3.03	3.27	3.50
CEC2017_F28	2.70	2.83	2.83	3.10	3.53
CEC2017_F29	3.67	3.93	3.40	1.70	2.30
CEC2017_F30	4.40	3.50	2.00	3.33	1.77
Mean Rank	3.61	3.41	2.98	2.78	2.22
Final Rank	5	4	3	2	1

4.2.2. Parameter {α,β} Settings

In this subsection, the two control parameters of CODGBGO are studied in detail to determine the optimal parameter combination to achieve the best performance of CODGBGO. In order to avoid experimental redundancy in parameter selection, a round of testing was performed before parameter {α, β} was selected, and finally parameter a was filtered to have α significant advantage of its effect in the interval [0.7, 0.9], and parameter β was filtered to have a significant advantage in the interval [0.85, 0.95]. For further determination of the parameter values, parameter α is divided into sets {0.7, 0.8, 0.9} at intervals of 0.1 and parameter b is divided into sets {0.85, 0.9, 0.95} at intervals of 0.5, so there are 9 combinations: {0.7, 0.85}, {0.7, 0.9}, {0.7, 0.95}, {0.8, 0.85}, {0.8, 0.9}, {0.8, 0.95}, {0.9, 0.85}, {0.9, 0.9}, and {0.9, 0.95}. In order to determine the best parameter combinations, these nine combinations are tested in the CEC2017 test function set with a test dimension of 30D. Each experiment was run independently 30 times, using the Friedman mean rank test to rank the experimental results and determine the optimal parameter combination. The experimental results are shown in Table 5.

From Table 5, it can be found that when the value of β is 0.85, the sum of the final rankings of the combinations {0.7, 0.85}, {0.8, 0.85}, and {0.9, 0.85} is 24. When the value of β is 0.9, the sum of the final rankings of the combinations {0.7, 0.9}, {0.8, 0.9}, and {0.9, 0.9} is 13. When the value of β is 0.95, the sum of the final rankings of the combinations {0.7, 0.95}, {0.8, 0.95}, and {0.9, 0.95} is 8. To summarize, the algorithm achieves better results when the value of the parameter β is 0.95. In the same way, when the value of α is 0.7, the sum of the final rankings of the combinations {0.7, 0.85}, {0.7, 0.9}, and {0.7, 0.95} is 14. When the value

of α is 0.8, the sum of the final rankings of the combinations {0.8, 0.85}, {0.8, 0.9}, and {0.8, 0.95} is 11. When the value of α is 0.9, the sum of the final rankings of the combinations {0.9, 0.85} and {0.8, 0.95} is 11. 0.9, 0.85}, {0.9, 0.9}, and {0.9, 0.95} have a final rank sum of 20. In summary, the algorithm can achieve better results when the parameter α takes the value of 0.8. Therefore, the parameter {0.8, 0.95} is chosen as the best parameter combination for subsequent experiments in this paper.

Table 5. Friedman mean rank test results for different parameter combinations.

Problems	{0.7, 0.9}	{0.7, 0.95}	{0.7, 0.85}	{0.8, 0.85}	{0.8, 0.9}	{0.9, 0.85}	{0.9, 0.9}	{0.9, 0.95}	{0.8, 0.95}
CEC2017_F1	5.03	5.17	5.90	5.37	4.13	5.40	4.40	4.43	5.17
CEC2017_F3	4.40	4.60	3.87	3.20	4.83	6.80	5.73	6.17	5.40
CEC2017_F4	4.70	5.43	5.77	4.30	4.17	5.17	5.20	5.33	4.93
CEC2017_F5	4.77	4.43	6.10	5.20	5.17	5.87	4.73	4.07	4.67
CEC2017_F6	4.90	3.87	7.10	6.73	4.27	6.83	4.90	3.20	3.20
CEC2017_F7	4.40	4.67	4.83	4.93	5.43	5.90	5.47	5.20	4.17
CEC2017_F8	5.50	3.77	5.83	4.97	5.57	6.40	4.53	4.60	3.83
CEC2017_F9	5.20	3.17	5.83	6.13	5.57	6.97	4.93	3.23	3.97
CEC2017_F10	5.03	5.70	3.67	4.83	4.60	4.43	4.60	6.07	6.07
CEC2017_F11	4.37	5.93	4.77	4.83	4.10	5.73	4.70	5.07	5.50
CEC2017_F12	5.57	5.73	5.50	5.83	4.37	5.87	4.23	4.30	3.60
CEC2017_F13	4.63	5.30	5.60	4.63	4.77	5.80	5.53	4.27	4.47
CEC2017_F14	5.03	4.77	5.07	4.33	4.80	5.07	5.13	6.50	4.30
CEC2017_F15	5.30	5.43	4.20	4.17	5.73	4.73	5.50	5.57	4.37
CEC2017_F16	5.57	5.17	3.80	5.13	4.43	4.13	5.77	5.40	5.60
CEC2017_F17	4.93	5.23	4.63	5.03	5.10	4.73	4.07	5.93	5.33
CEC2017_F18	4.87	4.73	5.30	4.40	4.07	5.30	5.13	5.63	5.57
CEC2017_F19	5.60	4.80	4.90	4.93	4.93	4.83	4.30	5.50	5.20
CEC2017_F20	5.40	4.53	5.40	5.40	5.00	5.43	4.67	4.57	4.60
CEC2017_F21	3.87	4.37	5.07	6.57	5.63	5.83	5.57	4.00	4.10
CEC2017_F22	4.23	4.70	4.47	4.18	4.53	5.23	5.90	6.48	5.27
CEC2017_F23	4.90	4.00	5.77	6.13	4.60	5.50	5.97	4.20	3.93
CEC2017_F24	4.47	3.97	5.53	5.63	5.73	5.37	5.27	4.67	4.37
CEC2017_F25	4.50	5.17	5.37	5.30	5.10	5.63	5.57	3.93	4.43
CEC2017_F26	5.43	3.57	5.67	5.77	4.50	6.33	5.20	4.27	4.27
CEC2017_F27	5.13	5.37	4.73	4.93	5.37	5.40	4.30	4.97	4.80
CEC2017_F28	5.13	5.13	5.37	4.80	4.87	4.80	4.83	5.67	4.40
CEC2017_F29	4.23	4.50	5.57	5.27	4.73	5.90	4.03	5.70	5.07
CEC2017_F30	5.67	5.13	5.43	4.83	4.17	5.33	4.83	5.57	4.03
Mean Rank	4.92	4.77	5.21	5.10	4.84	5.54	5.00	4.98	4.64
Final Rank	4	2	8	7	3	9	6	5	1

4.3. Experimental Results

In order to further evaluate the performance of CODGBGO, in this subsection, the proposed CODGBGO is experimentally compared with 10 common optimization algorithms on IEEE CEC2017 test functions, where the compared algorithms involve classical, highly citation, improved, and new algorithms. The test functions involve unimodal, multimodal, hybrid, and composition functions. The performance of CODGBGO is comprehensively assessed through population diversity analysis, exploration and exploitation analysis, numerical analysis, convergence and stability analysis, non-parametric analysis, and expanded analysis.

4.3.1. Population Diversity Analysis

Having a good population diversity contributes to the algorithm's ability to quickly converge to the global optimal solution and avoid getting trapped in local optima. In this section, we mainly analyze the differences in population diversity between GO and

CODGBGO. The IEEE CEC2017 test function set is used for diversity experiments, with a test dimension of 30D. The formula for calculating population diversity is shown below:

$$Ic(t) = \sqrt{\sum_{i=1}^{N}\sum_{j=1}^{D}\left(x_{i,j}(t) - c_j(t)\right)^2} \tag{20}$$

where $Ic(t)$ denotes the population diversity in generation t and $c_j(t)$ denotes the centrifugal degree of the jth dimension in generation t, using the following equation:

$$c_j(t) = \frac{1}{D}\sum_{i=1}^{N} x_{i,j}(t) \tag{21}$$

The experimental results are shown in Figure 4. As can be seen from Figure 4, the introduction of the Circle-OBL initialization strategy in CODGBGO results in a higher initial population diversity compared to GO. Additionally, the introduction of the exploration strategy ensures that CODGBGO consistently maintains higher population diversity throughout the iteration process compared to GO. In conclusion, CODGBGO is able to more effectively avoid getting trapped in local optima, allowing it to converge to the global optimal solution faster.

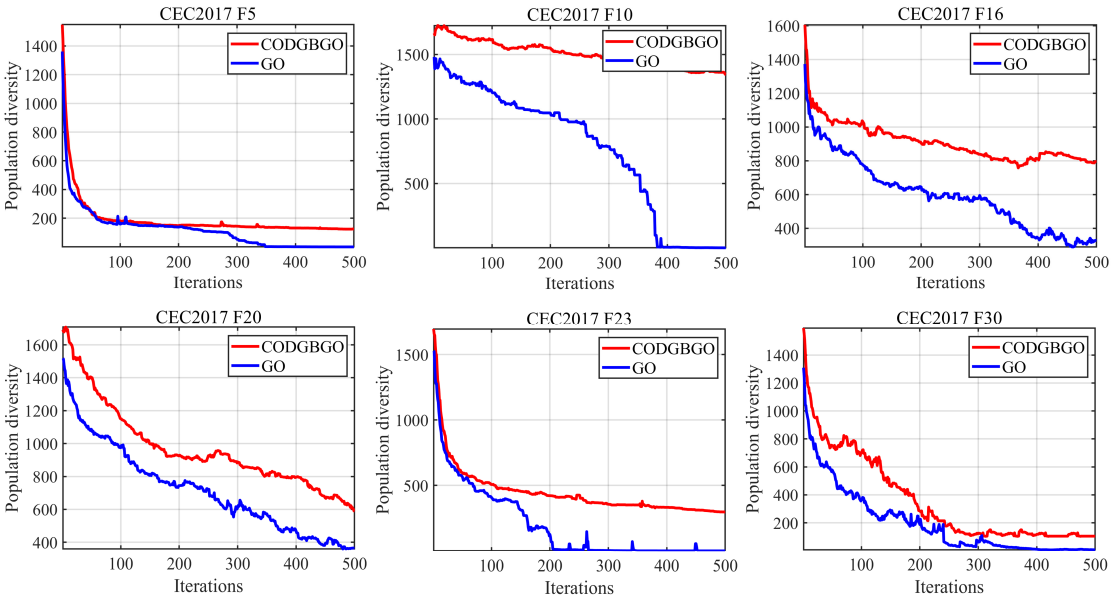

Figure 4. Comparison of population diversity between GO and CODGBGO.

4.3.2. Exploration and Exploitation Analysis

Exploration and exploitation are the two most important stages in metaheuristic algorithms, and effectively controlling these stages helps enhance algorithm performance. In this section, an analysis is conducted on the exploration and exploitation stages of CODGBGO using the IEEE CEC2017 test function set with a test dimension of 30D. The exploration ratio is calculated using Equation (22) and the exploitation ratio using Equation (23).

$$Exploration(\%) = \frac{Div(t)}{Div_{max}} \cdot 100 \tag{22}$$

$$Exploitation(\%) = \frac{|Div(t) - Div_{max}|}{Div_{max}} \cdot 100 \tag{23}$$

where $Div(t)$ denotes the dimension diversity measurement denoted as Equation (24), Div_{max} represents the maximum diversity achieved.

$$Div(t) = \frac{1}{D}\sum_{j=1}^{D}\frac{1}{N}\sum_{i=1}^{N}\left|median(x_j(t)) - x_{i,j}(t)\right| \tag{24}$$

The experimental results are shown in Figure 5. From Figure 5, it can be observed that in the early iterations, CODGBGO explores the search space with a high percentage, indicating its ability to effectively explore the search space. Subsequently, due to the introduction of the exploitation strategy into GO, the exploitation ratio gradually increases, enhancing the algorithm's convergence speed and precision. Throughout the iteration process, CODGBGO achieves a good balance between exploration and exploitation stages, effectively avoiding the problem of algorithm stagnation in local optima and improving the algorithm's solving performance.

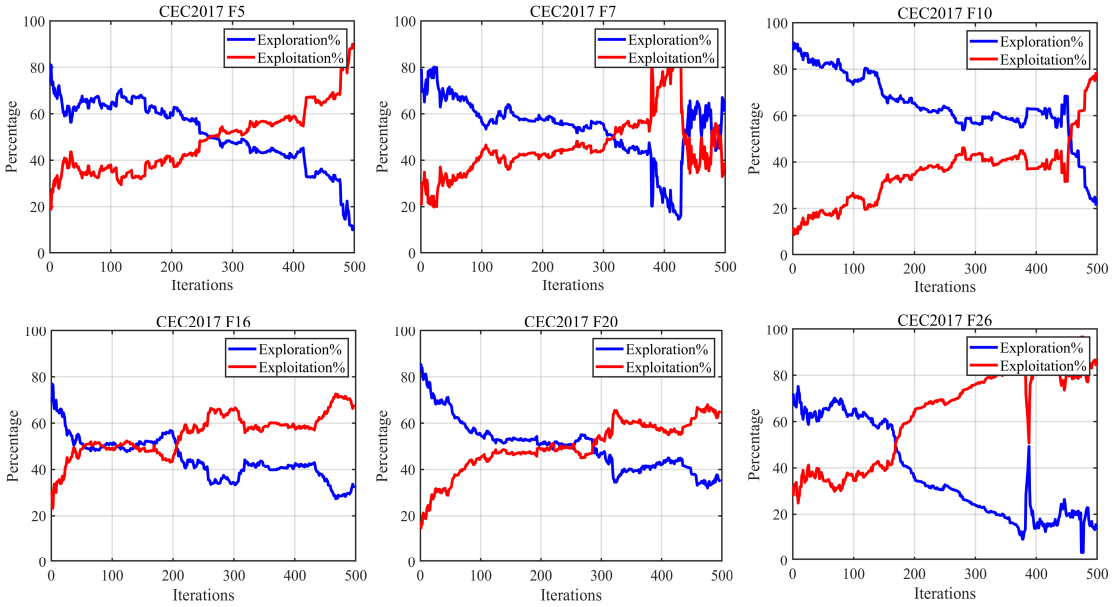

Figure 5. The exploration and exploitation ratio of CODGBGO.

4.3.3. Numerical Analysis

In this section, the performance of CODGBGO in solving numerical optimization problems is analyzed. The IEEE CEC2017 test functions are used for the experiments, and comparisons are made with 10 other state-of-the-art optimization algorithms. The test dimensions are 30D, 50D, and 100D, and the experimental results are presented in Table 6, Table 7, and Table 8, respectively, where the bold numbers in the table indicate the top rankings.

As can be seen from Table 6, when the test dimension is 30D, CODGBGO ranks first on the 24 test functions, demonstrating a very excellent solving performance. On the unimodal function F1, the solution accuracy is weaker than that of INFO, which is mainly due to the fact that INFO adopts a more advanced exploitation technique, which guarantees the convergence accuracy, while the exploitation strategy introduced in this paper is only second to INFO. However, on the simple multimodal functions F4 to F10, CODGBGO occupies a great advantage and ranks first in 85.7% of cases, which indicates that due to the introduction of the exploration strategy, the algorithm's ability to jump out of the local optimum trap has been strengthened. Meanwhile, on the complex multimodal functions consisting of hybrid and composition functions, CODGBGO obtains a winning percentage of 90%, which indicates that the exploitation strategy and exploration strategy proposed in this paper make the two phases of the algorithm well-balanced, which enhances the

algorithm's performance of the globally optimal search and can effectively cope with the challenges posed by the complex multimodal problems. Meanwhile, from a comprehensive point of view, CODGBGO outperforms other algorithms in terms of both the average ranking and the percentage of the first ranking, which indirectly confirms the role of the strategy proposed in this paper in boosting the performance of the algorithm.

Table 6. Comparison of numerical results for the IEEE CEC2017 test function set for 30D.

Problems	Metric	TLBO	SSA	INFO	DE	DBO	SO	CLPSO	BDE	BESD	GO	CODGBGO
CEC2017_F1	Mean	4.53×10^{3}	3.29×10^{3}	$\mathbf{1.05 \times 10^{2}}$	1.26×10^{5}	7.13×10^{6}	1.63×10^{4}	1.85×10^{9}	4.95×10^{5}	3.07×10^{8}	1.62×10^{2}	1.54×10^{2}
	Std	4.89×10^{3}	3.95×10^{3}	$\mathbf{7.16 \times 10^{0}}$	5.43×10^{4}	1.13×10^{7}	1.40×10^{4}	3.86×10^{8}	2.67×10^{5}	9.21×10^{7}	2.01×10^{2}	1.46×10^{2}
CEC2017_F3	Mean	4.03×10^{4}	1.14×10^{4}	6.84×10^{2}	4.00×10^{4}	7.03×10^{4}	5.50×10^{4}	9.29×10^{4}	2.70×10^{4}	2.13×10^{4}	7.70×10^{2}	$\mathbf{3.09 \times 10^{2}}$
	Std	8.00×10^{3}	4.68×10^{3}	1.53×10^{3}	7.37×10^{3}	1.64×10^{4}	1.07×10^{4}	1.56×10^{4}	6.04×10^{3}	3.76×10^{3}	5.10×10^{2}	$\mathbf{8.78 \times 10^{0}}$
CEC2017_F4	Mean	4.97×10^{2}	5.03×10^{2}	4.79×10^{2}	4.91×10^{2}	5.47×10^{2}	5.19×10^{2}	7.74×10^{2}	5.07×10^{2}	5.94×10^{2}	4.41×10^{2}	$\mathbf{4.28 \times 10^{2}}$
	Std	3.64×10^{1}	1.82×10^{1}	2.62×10^{1}	$\mathbf{8.80 \times 10^{0}}$	4.33×10^{1}	4.05×10^{1}	4.30×10^{1}	2.19×10^{1}	2.54×10^{1}	3.26×10^{1}	3.10×10^{1}
CEC2017_F5	Mean	5.90×10^{2}	6.43×10^{2}	6.36×10^{2}	7.21×10^{2}	7.10×10^{2}	5.71×10^{2}	7.28×10^{2}	6.23×10^{2}	6.81×10^{2}	6.06×10^{2}	$\mathbf{5.48 \times 10^{2}}$
	Std	1.73×10^{1}	4.24×10^{1}	3.01×10^{1}	1.28×10^{1}	4.79×10^{1}	2.70×10^{1}	1.77×10^{1}	$\mathbf{8.45 \times 10^{0}}$	1.15×10^{1}	1.84×10^{1}	1.66×10^{1}
CEC2017_F6	Mean	6.07×10^{2}	6.47×10^{2}	6.18×10^{2}	6.09×10^{2}	6.30×10^{2}	6.01×10^{2}	6.23×10^{2}	6.01×10^{2}	6.21×10^{2}	$\mathbf{6.00 \times 10^{2}}$	$\mathbf{6.00 \times 10^{2}}$
	Std	4.34×10^{0}	1.17×10^{1}	8.90×10^{0}	2.69×10^{0}	8.18×10^{0}	6.00×10^{-1}	2.68×10^{0}	2.06×10^{-1}	2.81×10^{0}	$\mathbf{7.81 \times 10^{-3}}$	2.92×10^{-2}
CEC2017_F7	Mean	8.61×10^{2}	8.98×10^{2}	9.49×10^{2}	9.48×10^{2}	9.13×10^{2}	8.14×10^{2}	1.07×10^{3}	8.77×10^{2}	9.35×10^{2}	8.58×10^{2}	$\mathbf{7.91 \times 10^{2}}$
	Std	4.16×10^{1}	5.80×10^{1}	5.35×10^{1}	1.35×10^{1}	7.17×10^{1}	3.18×10^{1}	2.59×10^{1}	1.24×10^{1}	1.42×10^{1}	$\mathbf{1.14 \times 10^{1}}$	1.90×10^{1}
CEC2017_F8	Mean	8.68×10^{2}	9.49×10^{2}	9.20×10^{2}	1.01×10^{3}	1.00×10^{3}	8.60×10^{2}	1.03×10^{3}	9.21×10^{2}	9.64×10^{2}	9.03×10^{2}	$\mathbf{8.52 \times 10^{2}}$
	Std	1.83×10^{1}	4.41×10^{1}	2.69×10^{1}	1.33×10^{1}	6.23×10^{1}	1.81×10^{1}	1.64×10^{1}	1.21×10^{1}	$\mathbf{9.94 \times 10^{0}}$	2.24×10^{1}	1.27×10^{1}
CEC2017_F9	Mean	1.20×10^{3}	4.39×10^{3}	2.80×10^{3}	1.13×10^{3}	5.57×10^{3}	1.06×10^{3}	4.49×10^{3}	9.63×10^{2}	1.92×10^{3}	$\mathbf{9.01 \times 10^{2}}$	9.03×10^{2}
	Std	2.76×10^{2}	1.49×10^{3}	6.58×10^{2}	2.97×10^{2}	2.30×10^{3}	1.07×10^{2}	6.86×10^{2}	5.94×10^{1}	3.24×10^{2}	$\mathbf{1.19 \times 10^{0}}$	2.95×10^{0}
CEC2017_F10	Mean	8.12×10^{3}	5.01×10^{3}	5.10×10^{3}	8.57×10^{3}	5.48×10^{3}	5.13×10^{3}	7.51×10^{3}	5.26×10^{3}	6.67×10^{3}	7.04×10^{3}	$\mathbf{4.54 \times 10^{3}}$
	Std	4.02×10^{2}	7.43×10^{2}	6.12×10^{2}	2.98×10^{2}	7.92×10^{2}	2.04×10^{3}	2.62×10^{2}	$\mathbf{1.76 \times 10^{2}}$	2.46×10^{2}	4.36×10^{2}	6.17×10^{2}
CEC2017_F11	Mean	1.25×10^{3}	1.30×10^{3}	1.26×10^{3}	1.23×10^{3}	1.54×10^{3}	1.22×10^{3}	1.80×10^{3}	1.20×10^{3}	1.32×10^{3}	1.17×10^{3}	$\mathbf{1.14 \times 10^{3}}$
	Std	4.93×10^{1}	5.10×10^{1}	4.65×10^{1}	2.63×10^{1}	1.28×10^{2}	4.63×10^{1}	1.86×10^{2}	2.13×10^{1}	$\mathbf{2.10 \times 10^{1}}$	2.66×10^{1}	2.80×10^{1}
CEC2017_F12	Mean	2.82×10^{5}	1.08×10^{7}	5.73×10^{4}	3.61×10^{5}	4.14×10^{7}	7.01×10^{6}	1.05×10^{8}	1.04×10^{6}	9.41×10^{6}	3.43×10^{4}	$\mathbf{1.69 \times 10^{4}}$
	Std	3.43×10^{5}	9.93×10^{6}	3.70×10^{4}	1.30×10^{5}	6.31×10^{7}	1.07×10^{7}	2.78×10^{7}	6.60×10^{5}	3.50×10^{6}	2.79×10^{4}	$\mathbf{8.16 \times 10^{3}}$
CEC2017_F13	Mean	1.45×10^{4}	1.09×10^{5}	1.78×10^{4}	2.27×10^{3}	2.81×10^{6}	1.17×10^{6}	2.75×10^{7}	2.16×10^{4}	1.16×10^{5}	4.22×10^{3}	$\mathbf{1.90 \times 10^{3}}$
	Std	1.25×10^{4}	6.58×10^{4}	1.37×10^{4}	2.08×10^{2}	5.92×10^{6}	1.36×10^{6}	1.75×10^{7}	1.33×10^{4}	4.89×10^{4}	1.10×10^{4}	$\mathbf{1.19 \times 10^{3}}$
CEC2017_F14	Mean	1.56×10^{4}	3.48×10^{4}	2.37×10^{3}	1.49×10^{3}	7.24×10^{4}	3.83×10^{4}	8.63×10^{4}	1.74×10^{3}	2.06×10^{3}	1.48×10^{3}	$\mathbf{1.46 \times 10^{3}}$
	Std	1.62×10^{4}	3.23×10^{4}	1.53×10^{3}	7.27×10^{0}	9.64×10^{4}	6.00×10^{4}	4.94×10^{4}	1.91×10^{2}	3.11×10^{2}	$\mathbf{1.06 \times 10^{1}}$	1.47×10^{1}
CEC2017_F15	Mean	8.45×10^{3}	7.78×10^{4}	3.98×10^{3}	1.62×10^{3}	7.81×10^{4}	1.49×10^{5}	1.09×10^{6}	5.62×10^{3}	5.12×10^{3}	1.62×10^{3}	$\mathbf{1.58 \times 10^{3}}$
	Std	6.89×10^{3}	5.31×10^{4}	4.13×10^{3}	$\mathbf{2.34 \times 10^{1}}$	6.62×10^{4}	2.51×10^{5}	6.83×10^{5}	5.05×10^{3}	9.80×10^{2}	4.24×10^{1}	3.21×10^{1}
CEC2017_F16	Mean	2.58×10^{3}	2.76×10^{3}	2.55×10^{3}	3.46×10^{3}	3.10×10^{3}	2.51×10^{3}	3.02×10^{3}	2.67×10^{3}	3.04×10^{3}	2.46×10^{3}	$\mathbf{2.22 \times 10^{3}}$
	Std	3.96×10^{2}	3.32×10^{2}	3.19×10^{2}	1.94×10^{2}	4.48×10^{2}	3.06×10^{2}	1.57×10^{2}	1.53×10^{2}	$\mathbf{1.07 \times 10^{2}}$	1.72×10^{2}	1.85×10^{2}
CEC2017_F17	Mean	1.92×10^{3}	2.25×10^{3}	2.26×10^{3}	2.38×10^{3}	2.46×10^{3}	2.20×10^{3}	2.15×10^{3}	1.99×10^{3}	2.06×10^{3}	1.90×10^{3}	$\mathbf{1.85 \times 10^{3}}$
	Std	9.02×10^{1}	2.45×10^{2}	2.24×10^{2}	1.02×10^{2}	2.06×10^{2}	2.07×10^{2}	9.02×10^{1}	9.75×10^{1}	8.35×10^{1}	$\mathbf{5.95 \times 10^{1}}$	9.11×10^{1}
CEC2017_F18	Mean	5.79×10^{5}	5.13×10^{5}	3.28×10^{4}	2.27×10^{3}	1.85×10^{6}	1.23×10^{6}	1.04×10^{6}	4.50×10^{4}	5.47×10^{4}	3.94×10^{3}	$\mathbf{1.93 \times 10^{3}}$
	Std	4.43×10^{5}	3.80×10^{5}	2.10×10^{4}	1.82×10^{2}	2.72×10^{6}	1.36×10^{6}	3.76×10^{5}	2.09×10^{4}	1.59×10^{4}	3.41×10^{3}	$\mathbf{5.00 \times 10^{1}}$
CEC2017_F19	Mean	8.00×10^{3}	1.84×10^{6}	3.39×10^{3}	1.95×10^{3}	1.06×10^{6}	2.00×10^{5}	1.60×10^{6}	6.07×10^{3}	7.12×10^{3}	1.96×10^{3}	$\mathbf{1.93 \times 10^{3}}$
	Std	6.13×10^{3}	1.15×10^{6}	2.99×10^{3}	$\mathbf{5.65 \times 10^{0}}$	1.38×10^{6}	4.62×10^{5}	1.09×10^{6}	4.17×10^{3}	1.85×10^{3}	8.45×10^{1}	7.15×10^{0}
CEC2017_F20	Mean	2.36×10^{3}	2.54×10^{3}	2.50×10^{3}	2.83×10^{3}	2.63×10^{3}	2.43×10^{3}	2.50×10^{3}	2.33×10^{3}	2.48×10^{3}	2.25×10^{3}	$\mathbf{2.16 \times 10^{3}}$
	Std	1.35×10^{2}	2.00×10^{2}	1.65×10^{2}	1.06×10^{2}	1.80×10^{2}	1.54×10^{2}	9.13×10^{1}	$\mathbf{6.95 \times 10^{1}}$	8.06×10^{1}	1.02×10^{2}	9.22×10^{1}
CEC2017_F21	Mean	2.37×10^{3}	2.44×10^{3}	2.43×10^{3}	2.51×10^{3}	2.52×10^{3}	2.37×10^{3}	2.50×10^{3}	2.42×10^{3}	2.47×10^{3}	2.41×10^{3}	$\mathbf{2.35 \times 10^{3}}$
	Std	2.03×10^{1}	3.73×10^{1}	2.76×10^{1}	1.57×10^{1}	4.66×10^{1}	1.87×10^{1}	5.21×10^{1}	$\mathbf{1.41 \times 10^{1}}$	8.89×10^{0}	1.55×10^{1}	1.61×10^{1}
CEC2017_F22	Mean	2.56×10^{3}	3.32×10^{3}	4.58×10^{3}	3.32×10^{3}	5.22×10^{3}	3.73×10^{3}	3.04×10^{3}	2.31×10^{3}	2.43×10^{3}	2.50×10^{3}	$\mathbf{2.30 \times 10^{3}}$
	Std	1.35×10^{3}	1.88×10^{3}	2.10×10^{3}	1.46×10^{3}	1.94×10^{3}	2.03×10^{3}	2.34×10^{2}	8.07×10^{0}	2.53×10^{1}	1.12×10^{3}	$\mathbf{7.36 \times 10^{-1}}$
CEC2017_F23	Mean	2.74×10^{3}	2.79×10^{3}	2.82×10^{3}	2.83×10^{3}	2.91×10^{3}	2.75×10^{3}	2.85×10^{3}	2.77×10^{3}	2.87×10^{3}	2.74×10^{3}	$\mathbf{2.69 \times 10^{3}}$
	Std	2.30×10^{1}	3.89×10^{1}	5.39×10^{1}	5.74×10^{1}	7.42×10^{1}	3.19×10^{1}	4.38×10^{1}	$\mathbf{1.49 \times 10^{1}}$	1.50×10^{1}	2.32×10^{1}	1.67×10^{1}
CEC2017_F24	Mean	2.91×10^{3}	2.93×10^{3}	2.96×10^{3}	3.04×10^{3}	3.06×10^{3}	2.90×10^{3}	3.04×10^{3}	2.95×10^{3}	3.03×10^{3}	2.95×10^{3}	$\mathbf{2.88 \times 10^{3}}$
	Std	2.12×10^{1}	3.20×10^{1}	4.24×10^{1}	$\mathbf{1.49 \times 10^{1}}$	8.10×10^{1}	3.58×10^{1}	1.88×10^{1}	1.69×10^{1}	1.86×10^{1}	1.71×10^{1}	1.60×10^{1}
CEC2017_F25	Mean	2.90×10^{3}	2.91×10^{3}	2.90×10^{3}	2.89×10^{3}	2.95×10^{3}	2.89×10^{3}	3.07×10^{3}	2.89×10^{3}	2.99×10^{3}	2.89×10^{3}	$\mathbf{2.89 \times 10^{3}}$
	Std	1.66×10^{1}	2.15×10^{1}	1.84×10^{1}	6.11×10^{-1}	4.56×10^{1}	1.48×10^{1}	3.71×10^{1}	7.63×10^{0}	1.63×10^{1}	$\mathbf{1.02 \times 10^{0}}$	1.56×10^{0}
CEC2017_F26	Mean	4.36×10^{3}	4.77×10^{3}	5.50×10^{3}	4.13×10^{3}	6.35×10^{3}	4.87×10^{3}	4.74×10^{3}	4.23×10^{3}	4.36×10^{3}	4.36×10^{3}	$\mathbf{3.96 \times 10^{3}}$
	Std	1.09×10^{3}	8.63×10^{2}	1.33×10^{3}	$\mathbf{2.27 \times 10^{2}}$	1.12×10^{3}	3.95×10^{2}	6.38×10^{2}	9.30×10^{2}	7.63×10^{2}	3.04×10^{2}	4.04×10^{2}
CEC2017_F27	Mean	3.24×10^{3}	3.26×10^{3}	3.26×10^{3}	3.23×10^{3}	3.27×10^{3}	3.26×10^{3}	3.27×10^{3}	3.24×10^{3}	3.37×10^{3}	$\mathbf{3.20 \times 10^{3}}$	3.21×10^{3}
	Std	2.11×10^{1}	4.69×10^{1}	2.93×10^{1}	7.35×10^{0}	4.11×10^{1}	1.67×10^{1}	7.69×10^{0}	$\mathbf{5.13 \times 10^{0}}$	1.69×10^{1}	9.68×10^{0}	1.03×10^{1}
CEC2017_F28	Mean	3.23×10^{3}	3.26×10^{3}	3.18×10^{3}	3.22×10^{3}	3.38×10^{3}	3.27×10^{3}	3.50×10^{3}	3.24×10^{3}	3.38×10^{3}	$\mathbf{3.16 \times 10^{3}}$	3.19×10^{3}
	Std	2.37×10^{1}	3.04×10^{1}	5.01×10^{1}	$\mathbf{1.02 \times 10^{1}}$	7.41×10^{1}	4.35×10^{1}	5.20×10^{1}	1.72×10^{1}	3.07×10^{1}	6.24×10^{1}	6.24×10^{1}
CEC2017_F29	Mean	3.69×10^{3}	4.07×10^{3}	4.12×10^{3}	4.39×10^{3}	4.20×10^{3}	3.85×10^{3}	4.14×10^{3}	3.69×10^{3}	4.12×10^{3}	3.62×10^{3}	$\mathbf{3.53 \times 10^{3}}$
	Std	1.37×10^{2}	2.83×10^{2}	3.06×10^{2}	1.88×10^{2}	2.84×10^{2}	2.45×10^{2}	1.43×10^{2}	$\mathbf{8.32 \times 10^{1}}$	1.09×10^{2}	8.50×10^{1}	9.22×10^{1}
CEC2017_F30	Mean	1.10×10^{4}	5.51×10^{6}	8.87×10^{3}	1.80×10^{4}	2.42×10^{6}	4.18×10^{5}	3.60×10^{6}	2.28×10^{4}	3.63×10^{5}	6.82×10^{3}	$\mathbf{5.32 \times 10^{3}}$
	Std	4.44×10^{3}	3.42×10^{6}	3.35×10^{3}	5.35×10^{3}	4.31×10^{6}	1.10×10^{6}	1.51×10^{6}	1.34×10^{4}	1.57×10^{5}	9.44×10^{2}	$\mathbf{3.24 \times 10^{2}}$
MeanRank		4.97	7.24	5.72	6.38	9.72	5.90	9.66	4.90	7.55	2.76	**1.21**
FinalRank		4	8	5	7	11	6	10	3	9	2	**1**
Rank First		0	0	1	0	0	0	0	0	0	4	**24**

Table 7. Comparison of numerical results for the IEEE CEC2017 test function set for 50D.

Problems	Metric	TLBO	SSA	INFO	DE	DBO	SO	CLPSO	BDE	BESD	GO	CODGBGO
CEC2017_F1	Mean	3.63×10^{7}	7.02×10^{3}	7.88×10^{4}	4.29×10^{7}	4.41×10^{8}	1.90×10^{7}	1.13×10^{10}	4.35×10^{7}	4.59×10^{9}	$\mathbf{4.31 \times 10^{3}}$	7.58×10^{3}
	Std	1.09×10^{8}	8.47×10^{3}	3.99×10^{5}	1.49×10^{7}	2.97×10^{8}	4.48×10^{7}	1.54×10^{9}	1.34×10^{7}	9.72×10^{8}	$\mathbf{5.09 \times 10^{3}}$	8.08×10^{3}
CEC2017_F3	Mean	1.60×10^{5}	1.01×10^{5}	1.99×10^{4}	1.97×10^{5}	2.13×10^{5}	1.33×10^{5}	2.27×10^{5}	1.04×10^{5}	8.33×10^{4}	3.73×10^{4}	$\mathbf{1.31 \times 10^{4}}$
	Std	2.23×10^{4}	2.97×10^{4}	6.91×10^{3}	2.91×10^{4}	4.47×10^{4}	1.38×10^{4}	2.50×10^{4}	1.75×10^{4}	1.26×10^{4}	1.35×10^{4}	$\mathbf{4.70 \times 10^{3}}$
CEC2017_F4	Mean	5.99×10^{2}	5.89×10^{2}	5.38×10^{2}	6.00×10^{2}	8.12×10^{2}	6.60×10^{2}	1.89×10^{3}	6.47×10^{2}	1.23×10^{3}	5.19×10^{2}	$\mathbf{5.17 \times 10^{2}}$
	Std	5.49×10^{1}	4.67×10^{1}	5.47×10^{1}	3.84×10^{1}	1.70×10^{2}	6.83×10^{1}	1.74×10^{2}	$\mathbf{2.59 \times 10^{1}}$	1.26×10^{2}	5.87×10^{1}	4.12×10^{1}
CEC2017_F5	Mean	6.95×10^{2}	8.38×10^{2}	8.07×10^{2}	9.53×10^{2}	9.13×10^{2}	6.34×10^{2}	9.98×10^{2}	7.90×10^{2}	9.09×10^{2}	7.84×10^{2}	$\mathbf{6.32 \times 10^{2}}$
	Std	2.82×10^{1}	5.35×10^{1}	4.61×10^{1}	1.94×10^{1}	7.88×10^{1}	4.36×10^{1}	1.94×10^{1}	$\mathbf{1.58 \times 10^{1}}$	1.94×10^{1}	3.30×10^{1}	3.68×10^{1}
CEC2017_F6	Mean	6.21×10^{2}	6.56×10^{2}	6.40×10^{2}	6.26×10^{2}	6.56×10^{2}	6.04×10^{2}	6.40×10^{2}	6.03×10^{2}	6.41×10^{2}	$\mathbf{6.00 \times 10^{2}}$	$\mathbf{6.00 \times 10^{2}}$
	Std	4.80×10^{0}	9.77×10^{0}	9.47×10^{0}	5.75×10^{0}	8.02×10^{0}	2.01×10^{0}	3.96×10^{0}	5.95×10^{-1}	3.98×10^{0}	$\mathbf{1.01 \times 10^{-1}}$	1.45×10^{-1}

Table 7. *Cont.*

Problems	Metric	TLBO	SSA	INFO	DE	DBO	SO	CLPSO	BDE	BESD	GO	CODGBGO
CEC2017_F7	Mean	1.07×10^{3}	1.14×10^{3}	1.30×10^{3}	1.22×10^{3}	1.16×10^{3}	9.39×10^{2}	1.59×10^{3}	1.08×10^{3}	1.26×10^{3}	1.08×10^{3}	$\mathbf{8.98 \times 10^{2}}$
	Std	6.20×10^{1}	1.27×10^{2}	1.07×10^{2}	1.86×10^{1}	1.48×10^{2}	4.11×10^{1}	5.46×10^{1}	2.65×10^{1}	3.16×10^{1}	$\mathbf{1.84 \times 10^{1}}$	4.87×10^{1}
CEC2017_F8	Mean	1.01×10^{3}	1.11×10^{3}	1.10×10^{3}	1.25×10^{3}	1.23×10^{3}	9.49×10^{2}	1.30×10^{3}	1.09×10^{3}	1.21×10^{3}	1.09×10^{3}	$\mathbf{9.14 \times 10^{2}}$
	Std	2.68×10^{1}	7.77×10^{1}	5.42×10^{1}	2.33×10^{1}	6.39×10^{1}	5.28×10^{1}	3.12×10^{1}	$\mathbf{2.01 \times 10^{1}}$	2.07×10^{1}	2.16×10^{1}	2.99×10^{1}
CEC2017_F9	Mean	6.86×10^{3}	1.28×10^{4}	7.97×10^{3}	6.81×10^{3}	1.96×10^{4}	1.77×10^{3}	1.98×10^{4}	2.32×10^{3}	1.19×10^{4}	$\mathbf{9.49 \times 10^{2}}$	1.06×10^{3}
	Std	2.92×10^{3}	2.96×10^{3}	1.70×10^{3}	2.72×10^{3}	7.92×10^{3}	4.09×10^{2}	2.77×10^{3}	4.48×10^{2}	1.44×10^{3}	$\mathbf{4.57 \times 10^{1}}$	2.05×10^{2}
CEC2017_F10	Mean	1.46×10^{4}	8.04×10^{3}	8.10×10^{3}	1.52×10^{4}	8.85×10^{3}	1.01×10^{4}	1.29×10^{4}	9.85×10^{3}	1.23×10^{4}	1.35×10^{4}	$\mathbf{8.23 \times 10^{3}}$
	Std	3.93×10^{2}	1.18×10^{3}	9.15×10^{2}	3.74×10^{2}	1.84×10^{3}	2.57×10^{3}	$\mathbf{3.31 \times 10^{2}}$	4.10×10^{2}	3.38×10^{2}	3.94×10^{2}	9.60×10^{2}
CEC2017_F11	Mean	1.35×10^{3}	1.58×10^{3}	1.31×10^{3}	1.41×10^{3}	2.70×10^{3}	1.48×10^{3}	5.43×10^{3}	1.54×10^{3}	2.28×10^{3}	1.28×10^{3}	$\mathbf{1.22 \times 10^{3}}$
	Std	6.00×10^{1}	1.07×10^{2}	6.18×10^{1}	3.59×10^{1}	1.78×10^{3}	1.05×10^{2}	1.28×10^{3}	1.19×10^{2}	2.50×10^{2}	$\mathbf{2.57 \times 10^{1}}$	3.47×10^{1}
CEC2017_F12	Mean	3.55×10^{6}	7.64×10^{7}	1.94×10^{6}	7.53×10^{6}	4.43×10^{8}	5.73×10^{7}	1.93×10^{9}	1.80×10^{7}	3.49×10^{8}	5.66×10^{5}	$\mathbf{3.39 \times 10^{5}}$
	Std	6.30×10^{6}	3.87×10^{7}	1.10×10^{6}	3.68×10^{6}	3.73×10^{8}	6.76×10^{7}	5.54×10^{8}	8.58×10^{6}	9.08×10^{7}	4.14×10^{5}	$\mathbf{2.95 \times 10^{5}}$
CEC2017_F13	Mean	8.57×10^{3}	1.75×10^{5}	7.50×10^{3}	6.61×10^{4}	1.95×10^{7}	5.51×10^{6}	4.50×10^{8}	1.34×10^{5}	1.11×10^{7}	9.39×10^{3}	$\mathbf{5.66 \times 10^{3}}$
	Std	5.31×10^{3}	1.05×10^{5}	$\mathbf{3.80 \times 10^{3}}$	1.77×10^{4}	2.78×10^{7}	6.05×10^{6}	1.68×10^{8}	2.17×10^{5}	5.75×10^{6}	8.55×10^{3}	6.40×10^{3}
CEC2017_F14	Mean	1.36×10^{5}	1.94×10^{5}	1.31×10^{4}	1.67×10^{3}	1.50×10^{6}	4.78×10^{5}	2.15×10^{6}	3.68×10^{4}	6.42×10^{4}	1.71×10^{3}	$\mathbf{1.61 \times 10^{3}}$
	Std	1.10×10^{5}	1.35×10^{5}	1.50×10^{4}	$\mathbf{2.15 \times 10^{1}}$	1.94×10^{6}	1.19×10^{6}	9.50×10^{5}	2.53×10^{4}	2.81×10^{4}	7.07×10^{1}	3.67×10^{1}
CEC2017_F15	Mean	8.25×10^{3}	5.70×10^{4}	1.09×10^{4}	2.80×10^{3}	5.76×10^{6}	1.01×10^{6}	7.54×10^{7}	2.29×10^{4}	2.77×10^{5}	3.46×10^{3}	$\mathbf{3.22 \times 10^{3}}$
	Std	6.30×10^{3}	2.34×10^{4}	7.18×10^{3}	2.63×10^{2}	2.36×10^{7}	1.58×10^{6}	3.16×10^{7}	1.82×10^{4}	1.25×10^{5}	4.41×10^{3}	$\mathbf{3.47 \times 10^{3}}$
CEC2017_F16	Mean	3.13×10^{3}	3.69×10^{3}	3.50×10^{3}	5.35×10^{3}	4.40×10^{3}	3.59×10^{3}	4.36×10^{3}	3.86×10^{3}	4.20×10^{3}	3.79×10^{3}	$\mathbf{3.08 \times 10^{3}}$
	Std	4.59×10^{2}	5.27×10^{2}	4.75×10^{2}	2.23×10^{2}	5.98×10^{2}	7.99×10^{2}	3.15×10^{2}	$\mathbf{1.80 \times 10^{2}}$	2.69×10^{2}	3.34×10^{2}	4.17×10^{2}
CEC2017_F17	Mean	2.92×10^{3}	3.40×10^{3}	3.25×10^{3}	4.08×10^{3}	4.12×10^{3}	3.19×10^{3}	3.89×10^{3}	3.23×10^{3}	3.34×10^{3}	3.05×10^{3}	$\mathbf{2.74 \times 10^{3}}$
	Std	2.99×10^{2}	3.74×10^{2}	3.52×10^{2}	1.75×10^{2}	4.02×10^{2}	4.31×10^{2}	2.38×10^{2}	1.93×10^{2}	$\mathbf{1.35 \times 10^{2}}$	1.52×10^{2}	2.29×10^{2}
CEC2017_F18	Mean	1.95×10^{6}	2.48×10^{6}	1.25×10^{5}	8.20×10^{4}	5.88×10^{6}	5.31×10^{6}	7.48×10^{6}	5.90×10^{5}	7.89×10^{5}	2.51×10^{4}	$\mathbf{7.92 \times 10^{3}}$
	Std	1.32×10^{6}	2.21×10^{6}	9.12×10^{4}	2.83×10^{4}	6.86×10^{6}	4.90×10^{6}	4.05×10^{6}	5.11×10^{5}	3.60×10^{5}	1.51×10^{4}	$\mathbf{5.77 \times 10^{3}}$
CEC2017_F19	Mean	1.84×10^{4}	3.30×10^{6}	1.54×10^{4}	2.27×10^{3}	5.29×10^{6}	6.62×10^{5}	1.54×10^{7}	2.05×10^{4}	1.14×10^{5}	2.28×10^{3}	$\mathbf{2.08 \times 10^{3}}$
	Std	8.69×10^{3}	2.57×10^{6}	1.08×10^{4}	1.18×10^{2}	7.55×10^{6}	1.23×10^{6}	6.95×10^{6}	7.84×10^{3}	3.26×10^{4}	4.59×10^{2}	$\mathbf{7.73 \times 10^{1}}$
CEC2017_F20	Mean	3.73×10^{3}	3.18×10^{3}	3.31×10^{3}	4.19×10^{3}	3.61×10^{3}	3.11×10^{3}	3.40×10^{3}	3.29×10^{3}	3.51×10^{3}	3.22×10^{3}	$\mathbf{2.99 \times 10^{3}}$
	Std	2.42×10^{2}	3.99×10^{2}	3.53×10^{2}	1.71×10^{2}	4.12×10^{2}	4.90×10^{2}	1.53×10^{2}	1.59×10^{2}	$\mathbf{1.52 \times 10^{2}}$	1.87×10^{2}	2.10×10^{2}
CEC2017_F21	Mean	2.50×10^{3}	2.61×10^{3}	2.58×10^{3}	2.75×10^{3}	2.79×10^{3}	2.45×10^{3}	2.78×10^{3}	2.59×10^{3}	2.69×10^{3}	2.58×10^{3}	$\mathbf{2.41 \times 10^{3}}$
	Std	4.04×10^{1}	7.03×10^{1}	5.79×10^{1}	2.42×10^{1}	9.89×10^{1}	4.42×10^{1}	1.79×10^{1}	$\mathbf{1.65 \times 10^{1}}$	1.75×10^{1}	3.64×10^{1}	3.04×10^{1}
CEC2017_F22	Mean	1.40×10^{4}	9.51×10^{3}	9.70×10^{3}	1.66×10^{4}	1.09×10^{4}	1.32×10^{4}	1.40×10^{4}	1.11×10^{4}	1.30×10^{4}	1.45×10^{4}	$\mathbf{8.96 \times 10^{3}}$
	Std	5.15×10^{3}	1.63×10^{3}	9.94×10^{2}	$\mathbf{4.20 \times 10^{2}}$	1.31×10^{3}	2.56×10^{3}	1.99×10^{3}	2.27×10^{3}	1.68×10^{3}	2.37×10^{3}	2.41×10^{3}
CEC2017_F23	Mean	2.98×10^{3}	3.03×10^{3}	3.13×10^{3}	3.19×10^{3}	3.36×10^{3}	2.95×10^{3}	3.24×10^{3}	3.05×10^{3}	3.27×10^{3}	2.99×10^{3}	$\mathbf{2.84 \times 10^{3}}$
	Std	4.59×10^{1}	7.99×10^{1}	7.16×10^{1}	2.61×10^{1}	9.04×10^{1}	7.45×10^{1}	2.96×10^{1}	$\mathbf{2.07 \times 10^{1}}$	3.67×10^{1}	5.07×10^{1}	3.13×10^{1}
CEC2017_F24	Mean	3.17×10^{3}	3.15×10^{3}	3.27×10^{3}	3.34×10^{3}	3.43×10^{3}	3.09×10^{3}	3.41×10^{3}	3.26×10^{3}	3.42×10^{3}	3.20×10^{3}	$\mathbf{3.02 \times 10^{3}}$
	Std	8.19×10^{1}	5.26×10^{1}	9.83×10^{1}	2.52×10^{1}	9.34×10^{1}	8.84×10^{1}	$\mathbf{2.29 \times 10^{1}}$	3.16×10^{1}	3.05×10^{1}	5.69×10^{1}	2.88×10^{1}
CEC2017_F25	Mean	3.14×10^{3}	3.08×10^{3}	3.06×10^{3}	3.07×10^{3}	3.51×10^{3}	3.11×10^{3}	4.42×10^{3}	3.14×10^{3}	3.68×10^{3}	3.03×10^{3}	$\mathbf{3.03 \times 10^{3}}$
	Std	4.32×10^{1}	4.06×10^{1}	2.90×10^{1}	$\mathbf{2.09 \times 10^{1}}$	1.28×10^{3}	4.18×10^{1}	2.05×10^{2}	2.92×10^{1}	1.13×10^{2}	3.62×10^{1}	3.62×10^{1}
CEC2017_F26	Mean	7.81×10^{3}	5.57×10^{3}	9.22×10^{3}	8.14×10^{3}	1.00×10^{4}	6.25×10^{3}	8.30×10^{3}	7.01×10^{3}	8.76×10^{3}	5.63×10^{3}	$\mathbf{4.88 \times 10^{3}}$
	Std	1.74×10^{3}	2.36×10^{3}	2.39×10^{3}	6.20×10^{2}	1.61×10^{3}	5.67×10^{2}	1.01×10^{3}	$\mathbf{1.92 \times 10^{2}}$	1.11×10^{3}	6.90×10^{2}	2.95×10^{2}
CEC2017_F27	Mean	3.57×10^{3}	3.55×10^{3}	3.65×10^{3}	3.53×10^{3}	3.75×10^{3}	3.59×10^{3}	3.72×10^{3}	3.55×10^{3}	4.20×10^{3}	$\mathbf{3.25 \times 10^{3}}$	3.29×10^{3}
	Std	1.45×10^{2}	1.14×10^{2}	1.70×10^{2}	1.23×10^{2}	2.11×10^{2}	8.92×10^{1}	5.44×10^{1}	$\mathbf{4.87 \times 10^{1}}$	8.02×10^{1}	5.61×10^{1}	5.38×10^{1}
CEC2017_F28	Mean	3.42×10^{3}	3.35×10^{3}	3.32×10^{3}	3.34×10^{3}	5.39×10^{3}	3.41×10^{3}	5.03×10^{3}	3.47×10^{3}	4.38×10^{3}	$\mathbf{3.29 \times 10^{3}}$	3.30×10^{3}
	Std	5.39×10^{1}	2.83×10^{1}	3.43×10^{1}	3.02×10^{1}	2.19×10^{3}	4.49×10^{1}	2.79×10^{2}	4.42×10^{1}	1.57×10^{2}	2.37×10^{1}	$\mathbf{2.18 \times 10^{1}}$
CEC2017_F29	Mean	4.48×10^{3}	5.38×10^{3}	4.97×10^{3}	5.83×10^{3}	5.47×10^{3}	4.46×10^{3}	5.74×10^{3}	4.28×10^{3}	5.37×10^{3}	4.02×10^{3}	$\mathbf{3.75 \times 10^{3}}$
	Std	2.98×10^{2}	5.12×10^{2}	4.32×10^{2}	2.47×10^{2}	5.22×10^{2}	4.56×10^{2}	3.15×10^{2}	$\mathbf{1.61 \times 10^{2}}$	2.04×10^{2}	2.29×10^{2}	2.08×10^{2}
CEC2017_F30	Mean	9.43×10^{5}	9.89×10^{7}	9.06×10^{5}	6.57×10^{6}	2.41×10^{7}	1.59×10^{7}	1.18×10^{8}	3.91×10^{6}	4.65×10^{7}	7.96×10^{5}	$\mathbf{7.70 \times 10^{5}}$
	Std	1.42×10^{5}	2.55×10^{7}	2.09×10^{5}	1.49×10^{6}	1.74×10^{7}	2.24×10^{7}	4.28×10^{7}	7.51×10^{5}	8.21×10^{6}	$\mathbf{1.00 \times 10^{5}}$	1.39×10^{5}
MeanRank		5.17	6.00	4.86	6.83	9.41	5.24	10.03	5.55	8.24	3.34	**1.31**
FinalRank		4	7	3	8	10	5	11	6	9	2	**1**
Rank First		0	0	0	0	0	0	0	0	0	5	**24**

Table 8. Comparison of numerical results for the IEEE CEC2017 test function for 100D.

Problems	Metric	TLBO	SSA	INFO	DE	DBO	SO	CLPSO	BDE	BESD	GO	CODGBGO
CEC2017_F1	Mean	6.64×10^{9}	$\mathbf{1.82 \times 10^{5}}$	2.06×10^{7}	7.10×10^{9}	4.61×10^{10}	6.02×10^{7}	7.23×10^{10}	2.18×10^{9}	4.57×10^{10}	2.47×10^{7}	2.21×10^{7}
	Std	3.12×10^{9}	$\mathbf{1.96 \times 10^{5}}$	6.46×10^{7}	1.84×10^{9}	4.83×10^{10}	1.07×10^{8}	6.83×10^{9}	4.04×10^{8}	5.52×10^{9}	9.71×10^{6}	1.24×10^{7}
CEC2017_F3	Mean	4.80×10^{5}	3.96×10^{5}	1.71×10^{5}	6.53×10^{5}	5.70×10^{5}	3.32×10^{5}	6.45×10^{5}	3.25×10^{5}	2.66×10^{5}	3.01×10^{5}	$\mathbf{1.70 \times 10^{5}}$
	Std	5.54×10^{4}	1.11×10^{5}	2.44×10^{4}	9.49×10^{4}	2.04×10^{5}	1.93×10^{4}	5.03×10^{4}	3.01×10^{4}	$\mathbf{1.55 \times 10^{4}}$	6.52×10^{4}	2.53×10^{4}
CEC2017_F4	Mean	1.65×10^{3}	8.84×10^{2}	8.94×10^{2}	1.48×10^{3}	6.19×10^{3}	1.01×10^{3}	1.04×10^{4}	1.29×10^{3}	6.04×10^{3}	7.74×10^{2}	$\mathbf{7.60 \times 10^{2}}$
	Std	3.22×10^{2}	6.50×10^{1}	1.48×10^{2}	1.91×10^{2}	6.09×10^{3}	1.54×10^{2}	1.49×10^{3}	9.55×10^{1}	6.34×10^{2}	$\mathbf{4.17 \times 10^{1}}$	4.72×10^{1}
CEC2017_F5	Mean	1.14×10^{3}	1.36×10^{3}	1.26×10^{3}	1.66×10^{3}	1.64×10^{3}	$\mathbf{9.94 \times 10^{2}}$	1.81×10^{3}	1.35×10^{3}	1.64×10^{3}	1.36×10^{3}	1.04×10^{3}
	Std	8.68×10^{1}	1.32×10^{2}	6.52×10^{1}	4.38×10^{1}	2.03×10^{2}	1.71×10^{2}	4.11×10^{1}	3.51×10^{1}	$\mathbf{2.35 \times 10^{1}}$	3.42×10^{1}	1.07×10^{2}
CEC2017_F6	Mean	6.42×10^{2}	6.67×10^{2}	6.54×10^{2}	6.60×10^{2}	6.68×10^{2}	6.15×10^{2}	6.69×10^{2}	6.14×10^{2}	6.69×10^{2}	$\mathbf{6.05 \times 10^{2}}$	6.07×10^{2}
	Std	4.98×10^{0}	6.82×10^{0}	6.68×10^{0}	7.82×10^{0}	7.97×10^{0}	3.02×10^{0}	3.86×10^{0}	1.44×10^{0}	3.38×10^{0}	$\mathbf{9.15 \times 10^{-1}}$	2.01×10^{0}
CEC2017_F7	Mean	2.09×10^{3}	2.02×10^{3}	2.61×10^{3}	2.25×10^{3}	2.40×10^{3}	$\mathbf{1.37 \times 10^{3}}$	3.58×10^{3}	1.81×10^{3}	2.42×10^{3}	1.76×10^{3}	1.52×10^{3}
	Std	1.66×10^{2}	1.97×10^{2}	2.32×10^{2}	9.08×10^{1}	3.89×10^{2}	5.48×10^{1}	1.59×10^{2}	4.45×10^{1}	1.03×10^{2}	$\mathbf{4.03 \times 10^{1}}$	1.02×10^{2}
CEC2017_F8	Mean	1.52×10^{3}	1.65×10^{3}	1.63×10^{3}	1.99×10^{3}	2.02×10^{3}	$\mathbf{1.31 \times 10^{3}}$	2.11×10^{3}	1.66×10^{3}	1.98×10^{3}	1.66×10^{3}	1.36×10^{3}
	Std	7.93×10^{1}	1.50×10^{2}	9.69×10^{1}	5.32×10^{1}	1.78×10^{2}	1.75×10^{2}	5.30×10^{1}	3.96×10^{1}	$\mathbf{2.88 \times 10^{1}}$	3.69×10^{1}	1.11×10^{2}
CEC2017_F9	Mean	4.52×10^{4}	3.00×10^{4}	2.17×10^{4}	4.69×10^{4}	4.66×10^{4}	6.21×10^{3}	9.05×10^{4}	2.45×10^{4}	4.75×10^{4}	$\mathbf{5.21 \times 10^{3}}$	7.28×10^{3}
	Std	8.32×10^{3}	3.55×10^{3}	2.40×10^{3}	8.12×10^{3}	2.03×10^{4}	$\mathbf{1.58 \times 10^{3}}$	8.77×10^{3}	4.25×10^{3}	3.95×10^{3}	1.94×10^{3}	3.42×10^{3}
CEC2017_F10	Mean	3.18×10^{4}	$\mathbf{1.62 \times 10^{4}}$	1.68×10^{4}	3.25×10^{4}	1.93×10^{4}	3.15×10^{4}	2.79×10^{4}	2.41×10^{4}	2.82×10^{4}	3.14×10^{4}	2.22×10^{4}
	Std	5.55×10^{2}	2.06×10^{3}	1.70×10^{3}	5.48×10^{2}	3.83×10^{3}	9.29×10^{2}	6.95×10^{2}	5.79×10^{2}	$\mathbf{4.35 \times 10^{2}}$	6.53×10^{2}	1.43×10^{3}
CEC2017_F11	Mean	4.46×10^{4}	1.88×10^{4}	3.63×10^{3}	3.93×10^{4}	1.43×10^{5}	8.18×10^{4}	1.30×10^{5}	3.13×10^{4}	4.91×10^{4}	4.03×10^{3}	$\mathbf{2.62 \times 10^{3}}$
	Std	1.08×10^{4}	6.80×10^{3}	5.47×10^{2}	7.47×10^{3}	4.98×10^{4}	1.99×10^{4}	2.50×10^{4}	7.28×10^{3}	8.33×10^{3}	6.43×10^{2}	$\mathbf{1.81 \times 10^{2}}$
CEC2017_F12	Mean	2.33×10^{8}	6.00×10^{8}	2.93×10^{7}	2.19×10^{8}	2.15×10^{9}	2.50×10^{8}	1.89×10^{10}	4.92×10^{8}	6.13×10^{9}	2.16×10^{7}	$\mathbf{1.22 \times 10^{7}}$
	Std	2.10×10^{8}	2.91×10^{8}	1.60×10^{7}	8.45×10^{7}	7.57×10^{8}	2.85×10^{8}	3.65×10^{9}	1.22×10^{8}	9.88×10^{8}	$\mathbf{6.63 \times 10^{6}}$	6.70×10^{6}
CEC2017_F13	Mean	2.07×10^{4}	8.79×10^{4}	2.19×10^{4}	1.04×10^{6}	1.21×10^{8}	1.28×10^{7}	2.17×10^{9}	1.21×10^{6}	3.06×10^{8}	$\mathbf{5.63 \times 10^{3}}$	8.05×10^{3}
	Std	8.46×10^{3}	3.31×10^{4}	3.61×10^{4}	5.75×10^{5}	1.30×10^{8}	1.31×10^{7}	5.39×10^{8}	6.14×10^{5}	7.45×10^{7}	$\mathbf{3.36 \times 10^{3}}$	6.08×10^{3}
CEC2017_F14	Mean	1.38×10^{6}	2.92×10^{6}	3.54×10^{5}	2.91×10^{5}	1.01×10^{7}	7.61×10^{6}	2.27×10^{7}	2.33×10^{6}	2.17×10^{6}	2.19×10^{4}	$\mathbf{9.00 \times 10^{3}}$
	Std	6.81×10^{5}	1.61×10^{6}	1.80×10^{5}	9.76×10^{4}	9.56×10^{6}	6.51×10^{6}	7.02×10^{6}	1.31×10^{6}	6.00×10^{5}	1.80×10^{4}	$\mathbf{1.15 \times 10^{4}}$
CEC2017_F15	Mean	5.68×10^{3}	7.87×10^{4}	4.69×10^{3}	1.74×10^{5}	1.44×10^{7}	2.87×10^{6}	5.02×10^{8}	1.00×10^{5}	2.40×10^{7}	5.38×10^{3}	$\mathbf{3.27 \times 10^{3}}$
	Std	3.51×10^{3}	2.97×10^{4}	2.75×10^{3}	6.10×10^{4}	4.14×10^{7}	4.44×10^{6}	1.75×10^{8}	7.34×10^{4}	9.34×10^{6}	5.19×10^{3}	$\mathbf{1.69 \times 10^{3}}$
CEC2017_F16	Mean	$\mathbf{5.64 \times 10^{3}}$	6.89×10^{3}	6.00×10^{3}	1.09×10^{4}	8.75×10^{3}	8.44×10^{3}	1.03×10^{4}	8.53×10^{3}	1.04×10^{4}	8.94×10^{3}	5.86×10^{3}
	Std	9.37×10^{2}	9.34×10^{2}	6.61×10^{2}	4.20×10^{2}	9.88×10^{2}	2.03×10^{3}	7.67×10^{2}	3.99×10^{2}	$\mathbf{3.63 \times 10^{2}}$	5.64×10^{2}	6.87×10^{2}

Table 8. *Cont.*

Problems	Metric	TLBO	SSA	INFO	DE	DBO	SO	CLPSO	BDE	BESD	GO	CODGBGO
CEC2017_F17	Mean	$\mathbf{4.93 \times 10^3}$	5.75×10^3	5.81×10^3	7.50×10^3	8.46×10^3	5.90×10^3	9.89×10^3	6.37×10^3	6.81×10^3	6.23×10^3	5.11×10^3
	Std	5.64×10^2	6.17×10^2	5.49×10^2	3.18×10^2	1.16×10^3	1.47×10^3	8.83×10^2	2.80×10^2	$\mathbf{2.65 \times 10^2}$	4.14×10^2	6.93×10^2
CEC2017_F18	Mean	2.75×10^6	4.21×10^6	4.11×10^5	2.61×10^6	2.12×10^7	1.01×10^7	2.90×10^7	2.82×10^6	2.61×10^6	2.32×10^5	$\mathbf{1.65 \times 10^5}$
	Std	1.44×10^6	2.26×10^6	2.20×10^5	6.22×10^5	2.05×10^7	1.21×10^7	7.53×10^6	9.69×10^5	6.06×10^5	9.28×10^4	$\mathbf{7.46 \times 10^4}$
CEC2017_F19	Mean	4.36×10^3	1.39×10^7	6.72×10^3	3.98×10^5	2.29×10^7	5.97×10^6	6.08×10^8	9.75×10^4	2.93×10^7	4.61×10^3	$\mathbf{3.31 \times 10^3}$
	Std	2.54×10^3	9.33×10^6	4.37×10^3	1.92×10^5	2.39×10^7	7.51×10^6	1.75×10^8	5.47×10^4	9.69×10^6	3.62×10^3	$\mathbf{1.86 \times 10^3}$
CEC2017_F20	Mean	7.09×10^3	$\mathbf{5.40 \times 10^3}$	5.55×10^3	7.84×10^3	6.06×10^3	6.88×10^3	6.26×10^3	6.36×10^3	6.84×10^3	6.81×10^3	5.57×10^3
	Std	3.13×10^2	6.23×10^2	5.44×10^2	3.10×10^2	7.10×10^2	4.56×10^2	4.14×10^2	$\mathbf{2.73 \times 10^2}$	2.81×10^2	3.47×10^2	5.70×10^2
CEC2017_F21	Mean	3.06×10^3	3.12×10^3	3.19×10^3	3.46×10^3	3.80×10^3	$\mathbf{2.83 \times 10^3}$	3.64×10^3	3.20×10^3	3.47×10^3	3.16×10^3	2.83×10^3
	Std	1.18×10^2	1.38×10^2	1.47×10^2	5.06×10^1	1.35×10^2	1.90×10^2	3.98×10^1	$\mathbf{3.65 \times 10^1}$	3.80×10^1	4.23×10^1	1.16×10^2
CEC2017_F22	Mean	3.34×10^4	$\mathbf{1.92 \times 10^4}$	1.98×10^4	3.48×10^4	2.22×10^4	3.17×10^4	3.03×10^4	2.67×10^4	3.09×10^4	3.33×10^4	2.50×10^4
	Std	4.79×10^3	1.54×10^3	1.93×10^3	5.24×10^2	3.05×10^3	2.55×10^3	7.53×10^2	5.89×10^2	$\mathbf{3.98 \times 10^2}$	6.27×10^2	1.15×10^3
CEC2017_F23	Mean	3.60×10^3	3.59×10^3	3.79×10^3	4.07×10^3	4.35×10^3	3.38×10^3	4.07×10^3	3.65×10^3	4.37×10^3	3.36×10^3	$\mathbf{3.21 \times 10^3}$
	Std	1.27×10^2	1.64×10^2	1.28×10^2	7.13×10^1	1.94×10^2	1.16×10^2	6.18×10^1	$\mathbf{3.14 \times 10^1}$	5.86×10^1	8.32×10^1	4.91×10^1
CEC2017_F24	Mean	4.35×10^3	4.14×10^3	4.59×10^3	4.52×10^3	5.26×10^3	4.05×10^3	4.79×10^3	4.23×10^3	5.06×10^3	4.03×10^3	$\mathbf{3.70 \times 10^3}$
	Std	1.68×10^2	1.59×10^2	2.56×10^2	6.90×10^1	2.41×10^2	1.67×10^2	$\mathbf{5.08 \times 10^1}$	5.11×10^1	8.05×10^1	1.02×10^2	7.94×10^1
CEC2017_F25	Mean	4.24×10^3	3.60×10^3	3.51×10^3	4.15×10^3	6.35×10^3	3.73×10^3	1.24×10^4	4.08×10^3	7.09×10^3	3.44×10^3	$\mathbf{3.41 \times 10^3}$
	Std	2.25×10^2	7.97×10^1	5.82×10^1	1.43×10^2	4.65×10^3	1.41×10^2	1.15×10^3	1.41×10^2	3.49×10^2	$\mathbf{4.57 \times 10^1}$	6.71×10^1
CEC2017_F26	Mean	2.27×10^4	1.44×10^4	2.26×10^4	1.87×10^4	2.47×10^4	1.36×10^4	2.14×10^4	1.57×10^4	2.47×10^4	1.28×10^4	$\mathbf{1.00 \times 10^4}$
	Std	3.09×10^3	3.15×10^3	4.12×10^3	8.12×10^2	2.53×10^3	1.73×10^3	8.55×10^2	$\mathbf{5.41 \times 10^2}$	1.14×10^3	1.11×10^3	8.84×10^2
CEC2017_F27	Mean	4.22×10^3	3.87×10^3	3.91×10^3	3.86×10^3	4.18×10^3	3.84×10^3	4.51×10^3	4.01×10^3	5.31×10^3	$\mathbf{3.42 \times 10^3}$	3.43×10^3
	Std	2.45×10^2	1.40×10^2	2.73×10^2	2.03×10^2	2.15×10^2	1.21×10^2	1.08×10^2	7.65×10^1	1.37×10^2	$\mathbf{3.67 \times 10^1}$	4.26×10^1
CEC2017_F28	Mean	5.13×10^3	3.68×10^3	3.62×10^3	4.41×10^3	1.76×10^4	4.31×10^3	1.58×10^4	5.05×10^3	1.04×10^4	$\mathbf{3.53 \times 10^3}$	$\mathbf{3.53 \times 10^3}$
	Std	6.61×10^2	9.50×10^1	7.47×10^1	3.36×10^2	5.79×10^3	3.45×10^2	1.33×10^3	3.29×10^2	5.32×10^2	5.10×10^1	$\mathbf{3.76 \times 10^1}$
CEC2017_F29	Mean	7.82×10^3	9.56×10^3	7.66×10^3	1.06×10^4	1.03×10^4	7.28×10^3	1.46×10^4	8.35×10^3	1.14×10^4	8.25×10^3	$\mathbf{6.72 \times 10^3}$
	Std	6.19×10^2	9.87×10^2	6.62×10^2	3.48×10^2	1.46×10^3	1.07×10^3	1.01×10^3	$\mathbf{2.88 \times 10^2}$	3.46×10^2	5.06×10^2	7.14×10^2
CEC2017_F30	Mean	4.26×10^5	1.22×10^8	9.77×10^4	1.80×10^6	8.40×10^7	1.04×10^6	1.06×10^9	4.06×10^6	3.47×10^8	5.73×10^4	$\mathbf{3.34 \times 10^4}$
	Std	3.61×10^5	5.26×10^7	1.05×10^5	5.53×10^5	1.33×10^8	6.90×10^5	2.67×10^8	1.17×10^6	1.08×10^8	2.85×10^4	$\mathbf{1.15 \times 10^4}$
MeanRank		5.76	4.90	4.07	7.55	8.72	5.07	9.93	5.86	8.79	3.59	**1.76**
FinalRank		6	4	3	8	9	5	11	7	10	2	**1**
Rank First		2	4	0	0	0	4	0	0	0	5	**14**

Meanwhile, when the test dimension is 50D, CODGBGO also demonstrates strong solution performance, ranking first on 82.7% of the test functions. Among them, it dominates 85.7% of the simple multimodal test functions, which also shows that the exploration strategy proposed in this paper can still effectively improve the algorithm's ability to jump out of the local optimum with an increase in test dimensions. Meanwhile, on the complex multimodal function, it occupies 90% of the solution advantage, which also confirms that the exploitation strategy and exploration strategy proposed in this paper still have effectiveness with the increase of the test dimension. Meanwhile, from the perspective of the first overall ranking, it can be seen that the strategy proposed in this paper is still effective in promoting the algorithm as the problem dimension increases.

Finally, when the test dimension is 100D, the solution performance of all algorithms shows a decline. Inevitably, the solving performance of CODGBGO also shows a slight decline, but the overall ranking shows that the proposed CODGBGO still has a certain advantage in solving high-dimensional test problems. Specifically, it achieves a winning rate of 60% on the complex multimodal problem. This also indirectly reflects that the exploration strategy and exploitation strategy still have a certain contribution to the performance of the algorithm when solving high-dimensional problems.

To summarize, by testing different dimensions at CEC2017, it is confirmed that the exploration strategy and exploitation strategy proposed in this paper can promote the algorithm effectively. In addition, the disadvantage is that the facilitation will gradually decrease as the test dimensions rise, but this is within an acceptable range.

4.3.4. Convergence and Stability Analysis

In addition to the solution accuracy, the convergence speed and stability of the algorithm are also important. This subsection will analyze the convergence and stability of the algorithm. Experiments are conducted using the IEEE CEC2017 test function set with a test dimension of 30 D. The experimental results are shown in Figures 6 and 7. From Figure 6, it can be seen that in most cases, CODGBGO is in the lead after 30,000 function evaluations, with faster convergence speed and convergence accuracy. Meanwhile, from Figure 7, it can be seen that CODGBGO has higher solution stability.

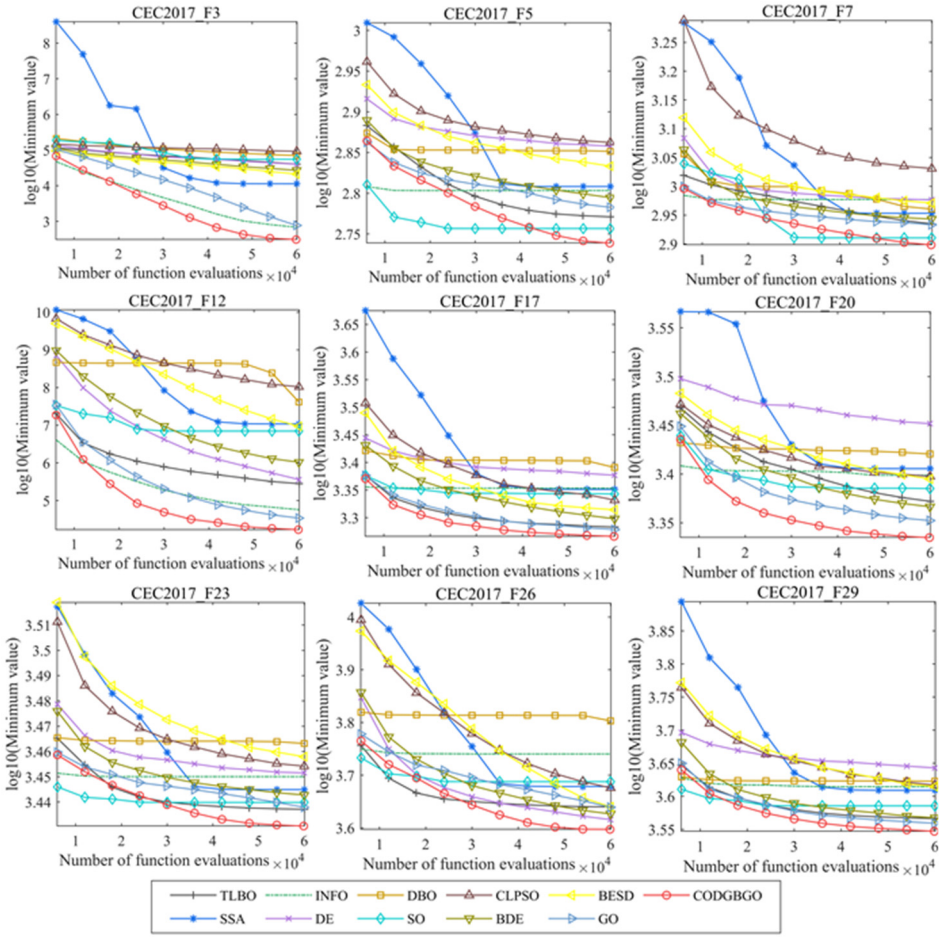

Figure 6. Comparison of convergence curves on the IEEE CEC2017 test function set.

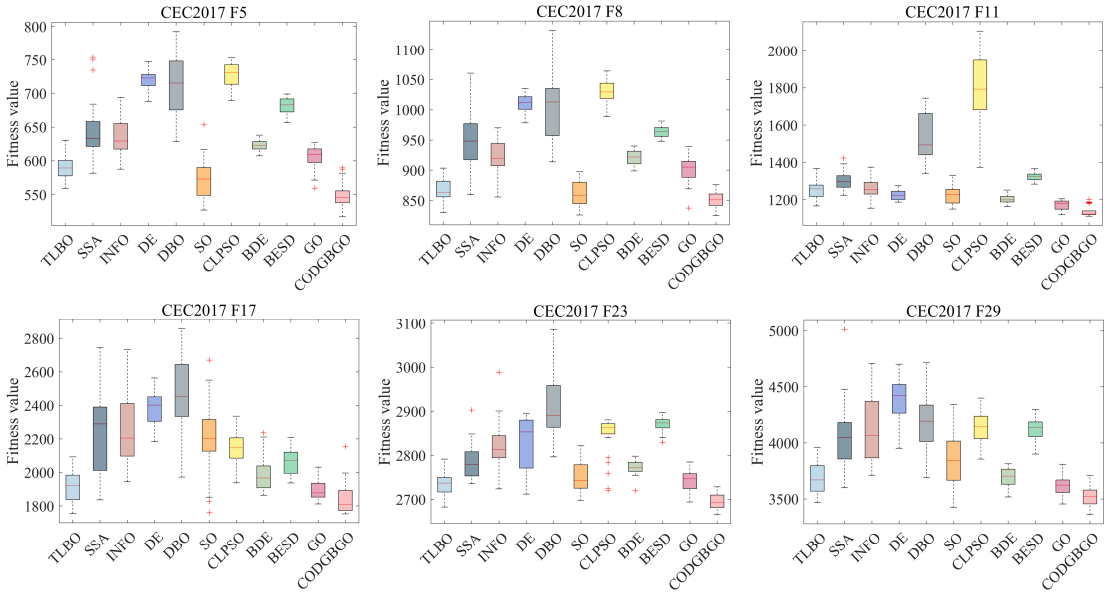

Figure 7. Comparison of stability on the IEEE CEC2017 test function set.

4.3.5. Nonparametric Analysis

In this subsection, non-parametric tests are used to compare the differences between CODGBGO and competitive algorithms, and experiments are conducted using the IEEE CEC2017 test function set with test dimensions of 30D, 50D, and 100D.First, the Wilcoxon

rank sum test is used to compare the differences between CODGBGO and the competitive algorithms, and the results of the experiments are shown in Table 9, Table 10, and Table 11, respectively. Where the significance factor $p < 0.05$ indicates that there is a significant difference between the competitive algorithms and CODGBGO, and on the contrary, it indicates that there is no significant difference between the competitive algorithms and CODGBGO. When there is a significant difference between the algorithms, the competitive algorithms are determined to be superior or weaker than CODGBGO by comparing the means. where '+' indicates that the competitive algorithms are significantly better than CODGBGO, '−' indicates that the competitive algorithms are significantly weaker than CODGBGO, and '=' indicates that there is no significant difference between the competitive algorithm and CODGBGO.As can be seen from Tables 9–11, CODGBGO significantly outperforms the other competing algorithms in most cases, demonstrating strong comprehensive solution performance. Secondly, the experimental results are ranked using the Friedman mean rank test, and the results are shown in Table 12, from which it can be seen that CODGBGO is ahead of the competing algorithms in terms of average rank on different test dimensions. In summary, it can be shown that the CODGBGO proposed in this paper is more competitive than other competitive algorithms on the IEEE CEC2017 test set.

Table 9. *p*-values of the IEEE CEC2017 test function set for 30D.

Problems	TLBO	SSA	INFO	DE	DBO	SO	CLPSO	BDE	BESD	GO
CEC2017_F1	2.67×10^{-9}/−	4.18×10^{-9}/−	1.50×10^{-2}/+	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	1.70×10^{-2}/−
CEC2017_F3	3.02×10^{-11}/−	3.02×10^{-11}/−	1.22×10^{-2}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−
CEC2017_F4	3.82×10^{-9}/−	3.02×10^{-11}/−	4.69×10^{-8}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	2.15×10^{-10}/−	3.02×10^{-11}/−	7.39×10^{-11}/−	3.02×10^{-11}/−	3.27×10^{-2}/−
CEC2017_F5	1.86×10^{-9}/−	4.50×10^{-11}/−	3.69×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.01×10^{-4}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	1.78×10^{-10}/−
CEC2017_F6	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	2.01×10^{-4}/+
CEC2017_F7	1.55×10^{-9}/−	7.39×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	2.37×10^{-10}/−	4.03×10^{-3}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	4.08×10^{-11}/−
CEC2017_F8	4.71×10^{-4}/−	6.70×10^{-11}/−	2.15×10^{-10}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	8.50×10^{-2}/=	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	5.57×10^{-10}/−
CEC2017_F9	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	4.08×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	6.55×10^{-4}/+
CEC2017_F10	3.02×10^{-11}/−	2.51×10^{-2}/−	1.30×10^{-3}/−	3.02×10^{-11}/−	9.51×10^{-6}/−	5.59×10^{-1}/=	3.02×10^{-11}/−	1.61×10^{-6}/−	3.02×10^{-11}/−	3.02×10^{-11}/−
CEC2017_F11	1.61×10^{-10}/−	3.02×10^{-11}/−	1.33×10^{-10}/−	7.39×10^{-11}/−	3.02×10^{-11}/−	9.26×10^{-9}/−	3.02×10^{-11}/−	2.92×10^{-9}/−	3.02×10^{-11}/−	1.64×10^{-5}/−
CEC2017_F12	4.50×10^{-11}/−	3.02×10^{-11}/−	4.57×10^{-9}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	2.62×10^{-3}/−
CEC2017_F13	3.47×10^{-10}/−	3.02×10^{-11}/−	2.61×10^{-10}/−	1.43×10^{-8}/−	3.02×10^{-11}/−	6.07×10^{-11}/−	3.02×10^{-11}/−	4.50×10^{-11}/−	3.02×10^{-11}/−	4.22×10^{-4}/−
CEC2017_F14	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.20×10^{-9}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	1.19×10^{-6}/−
CEC2017_F15	3.02×10^{-11}/−	3.02×10^{-11}/−	6.07×10^{-11}/−	3.57×10^{-6}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	4.71×10^{-4}/−
CEC2017_F16	1.17×10^{-4}/−	1.31×10^{-8}/−	1.34×10^{-5}/−	3.02×10^{-11}/−	2.61×10^{-10}/−	7.20×10^{-5}/−	3.02×10^{-11}/−	2.15×10^{-10}/−	3.02×10^{-11}/−	1.53×10^{-5}/−
CEC2017_F17	1.24×10^{-3}/−	7.77×10^{-9}/−	1.33×10^{-10}/−	3.02×10^{-11}/−	4.50×10^{-11}/−	1.31×10^{-8}/−	2.15×10^{-10}/−	3.01×10^{-7}/−	1.17×10^{-9}/−	3.67×10^{-3}/−
CEC2017_F18	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	4.08×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	1.33×10^{-10}/−
CEC2017_F19	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	7.39×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	1.31×10^{-8}/−
CEC2017_F20	2.20×10^{-7}/−	1.78×10^{-10}/−	1.69×10^{-9}/−	3.02×10^{-11}/−	1.33×10^{-10}/−	1.85×10^{8}/−	5.49×10^{-11}/−	2.83×10^{-8}/−	5.49×10^{-11}/−	2.27×10^{-3}/−
CEC2017_F21	6.53×10^{-7}/−	3.69×10^{-11}/−	3.69×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	1.25×10^{-5}/−	6.70×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	6.70×10^{-11}/−
CEC2017_F22	1.29×10^{-9}/−	1.86×10^{-9}/−	1.11×10^{-4}/−	3.02×10^{-11}/−	5.57×10^{-10}/−	1.61×10^{-10}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	5.00×10^{-9}/−
CEC2017_F23	1.20×10^{-8}/−	3.02×10^{-11}/−	3.34×10^{-11}/−	4.98×10^{-11}/−	3.02×10^{-11}/−	4.20×10^{-10}/−	4.08×10^{-11}/−	4.08×10^{-11}/−	3.02×10^{-11}/−	9.26×10^{-9}/−
CEC2017_F24	2.00×10^{-5}/−	8.48×10^{-9}/−	3.47×10^{-10}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	1.30×10^{-3}/−	3.02×10^{-11}/−	4.50×10^{-11}/−	3.02×10^{-11}/−	6.07×10^{-11}/−
CEC2017_F25	9.06×10^{-8}/−	5.07×10^{-10}/−	3.57×10^{-6}/−	1.17×10^{-3}/−	3.02×10^{-11}/−	5.60×10^{-7}/−	3.02×10^{-11}/−	9.92×10^{-11}/−	3.02×10^{-11}/−	1.45×10^{-1}/=
CEC2017_F26	1.03×10^{-2}/−	9.53×10^{-7}/−	1.86×10^{-6}/−	1.58×10^{-1}/=	3.82×10^{-9}/−	1.21×10^{-10}/−	9.06×10^{-8}/−	2.71×10^{-2}/−	8.77×10^{-2}/=	8.15×10^{-5}/−
CEC2017_F27	7.12×10^{-9}/−	2.87×10^{-10}/−	1.21×10^{-10}/−	2.87×10^{-10}/−	7.39×10^{-11}/−	6.07×10^{-11}/−	3.02×10^{-11}/−	4.98×10^{-11}/−	3.02×10^{-11}/−	1.68×10^{-3}/+
CEC2017_F28	4.22×10^{-4}/−	9.83×10^{-8}/−	2.77×10^{-1}/=	7.62×10^{-3}/−	3.34×10^{-11}/−	3.08×10^{-8}/−	3.02×10^{-11}/−	5.97×10^{-5}/−	3.02×10^{-11}/−	3.39×10^{2}/+
CEC2017_F29	1.25×10^{-5}/−	8.15×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	4.08×10^{-11}/−	2.38×10^{-7}/−	3.02×10^{-11}/−	8.35×10^{-8}/−	3.02×10^{-11}/−	2.25×10^{-4}/−
CEC2017_F30	4.08×10^{-11}/−	3.02×10^{-11}/−	1.61×10^{-10}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	9.92×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.02×10^{-11}/−	3.16×10^{-10}/−
+/−/=	0/29/0	0/29/0	1/27/1	0/28/1	0/29/0	0/27/2	0/29/0	0/29/0	0/28/1	4/24/1

Table 10. *p*-values of the IEEE CEC2017 test function set for 50D.

Problems	TLBO	SSA	INFO	DE	DBO	SO	CLPSO	BDE	BESD	GO
CEC2017_F1	$3.02\times10^{-11}/-$	$4.73\times10^{-1}/=$	$6.31\times10^{-1}/=$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.09\times10^{-1}/=$
CEC2017_F3	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.83\times10^{-5}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.55\times10^{-9}/-$
CEC2017_F4	$2.78\times10^{-7}/-$	$8.84\times10^{-7}/-$	$4.36\times10^{-2}/-$	$1.01\times10^{-8}/-$	$3.02\times10^{-11}/-$	$1.96\times10^{-10}/-$	$3.02\times10^{-11}/-$	$3.69\times10^{-11}/-$	$3.02\times10^{-11}/-$	$8.07\times10^{-1}/=$
CEC2017_F5	$7.69\times10^{-8}/-$	$3.02\times10^{-11}/-$	$3.69\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.69\times10^{-11}/-$	$9.47\times10^{-1}/=$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$4.08\times10^{-11}/-$
CEC2017_F6	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.00\times10^{-3}/+$
CEC2017_F7	$1.46\times10^{-10}/-$	$8.15\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$5.49\times10^{-11}/-$	$3.77\times10^{-4}/-$	$3.02\times10^{-11}/-$	$3.69\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$
CEC2017_F8	$1.21\times10^{-10}/-$	$5.49\times10^{-11}/-$	$3.34\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.84\times10^{-2}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$
CEC2017_F9	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.07\times10^{-9}/-$	$3.02\times10^{-11}/-$	$4.08\times10^{-11}/-$	$3.02\times10^{-11}/-$	$4.46\times10^{-4}/+$
CEC2017_F10	$3.02\times10^{-11}/-$	$5.20\times10^{-1}/=$	$5.69\times10^{-1}/=$	$3.02\times10^{-11}/-$	$2.46\times10^{-1}/=$	$6.20\times10^{4}/-$	$3.02\times10^{-11}/-$	$8.48\times10^{-9}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$
CEC2017_F11	$3.82\times10^{-10}/-$	$3.02\times10^{-11}/-$	$3.08\times10^{-8}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$5.53\times10^{-8}/-$
CEC2017_F12	$8.10\times10^{-10}/-$	$3.02\times10^{-11}/-$	$3.16\times10^{-10}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.03\times10^{-2}/-$
CEC2017_F13	$5.56\times10^{-4}/-$	$3.02\times10^{-11}/-$	$9.52\times10^{-4}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.82\times10^{-10}/-$	$3.02\times10^{-11}/-$	$9.92\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.03\times10^{-2}/-$
CEC2017_F14	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.16\times10^{-7}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.96\times10^{-8}/-$
CEC2017_F15	$1.09\times10^{-5}/-$	$3.34\times10^{-11}/-$	$1.73\times10^{-7}/-$	$2.53\times10^{-4}/+$	$3.02\times10^{-11}/-$	$3.82\times10^{-10}/-$	$3.02\times10^{-11}/-$	$5.07\times10^{-10}/-$	$3.02\times10^{-11}/-$	$3.18\times10^{-1}/=$
CEC2017_F16	$6.31\times10^{-1}/=$	$1.09\times10^{-5}/-$	$2.68\times10^{-4}/-$	$3.02\times10^{-11}/-$	$8.89\times10^{-10}/-$	$4.03\times10^{-3}/-$	$1.33\times10^{-10}/-$	$5.00\times10^{-9}/-$	$3.16\times10^{-10}/-$	$1.47\times10^{-7}/-$
CEC2017_F17	$2.42\times10^{-2}/-$	$9.76\times10^{-10}/-$	$6.01\times10^{-8}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$2.77\times10^{-5}/-$	$3.02\times10^{-11}/-$	$1.17\times10^{-9}/-$	$4.08\times10^{-11}/-$	$1.60\times10^{-7}/-$
CEC2017_F18	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$2.20\times10^{-7}/-$
CEC2017_F19	$5.49\times10^{-11}/-$	$3.02\times10^{-11}/-$	$5.49\times10^{-11}/-$	$6.52\times10^{-9}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.89\times10^{-4}/-$
CEC2017_F20	$1.78\times10^{-10}/-$	$4.36\times10^{-2}/-$	$3.18\times10^{-4}/-$	$3.02\times10^{-11}/-$	$9.83\times10^{-8}/-$	$3.95\times10^{-1}/=$	$1.01\times10^{-8}/-$	$1.29\times10^{-6}/-$	$3.82\times10^{-10}/-$	$9.21\times10^{-5}/-$
CEC2017_F21	$1.86\times10^{-9}/-$	$3.34\times10^{-11}/-$	$3.69\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$6.20\times10^{-4}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.34\times10^{-11}/-$
CEC2017_F22	$3.32\times10^{-6}/-$	$4.04\times10^{-1}/=$	$3.63\times10^{-1}/=$	$3.02\times10^{-11}/-$	$1.43\times10^{-5}/-$	$3.08\times10^{-8}/-$	$1.07\times10^{-9}/-$	$9.06\times10^{-8}/-$	$2.02\times10^{-8}/-$	$5.07\times10^{-10}/-$
CEC2017_F23	$6.70\times10^{-11}/-$	$3.34\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$2.03\times10^{-9}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.78\times10^{-10}/-$
CEC2017_F24	$1.78\times10^{-10}/-$	$6.70\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.04\times10^{-4}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$4.98\times10^{-11}/-$
CEC2017_F25	$1.96\times10^{-10}/-$	$3.09\times10^{-6}/-$	$1.06\times10^{-3}/-$	$4.35\times10^{-5}/-$	$9.92\times10^{-11}/-$	$1.29\times10^{-9}/-$	$3.02\times10^{-11}/-$	$5.49\times10^{-11}/-$	$3.02\times10^{-11}/-$	$7.51\times10^{-1}/=$
CEC2017_F26	$8.48\times10^{-9}/-$	$1.86\times10^{-1}/=$	$8.48\times10^{-9}/-$	$3.02\times10^{-11}/-$	$5.57\times10^{-10}/-$	$6.70\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$2.43\times10^{-5}/-$
CEC2017_F27	$8.15\times10^{-11}/-$	$5.49\times10^{-11}/-$	$7.39\times10^{-11}/-$	$1.46\times10^{-10}/-$	$4.50\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$2.68\times10^{-4}/+$
CEC2017_F28	$6.70\times10^{-11}/-$	$2.83\times10^{-8}/-$	$4.43\times10^{-3}/-$	$3.32\times10^{-6}/-$	$3.02\times10^{-11}/-$	$3.69\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$2.77\times10^{-1}/=$
CEC2017_F29	$5.49\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$5.00\times10^{-9}/-$	$3.02\times10^{-11}/-$	$7.39\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.25\times10^{-4}/-$
CEC2017_F30	$1.09\times10^{-5}/-$	$3.02\times10^{-11}/-$	$1.03\times10^{-2}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$4.50\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$8.50\times10^{-2}/=$
+/−/=	0/28/1	0/25/4	0/26/3	1/28/0	0/28/1	0/27/2	0/29/0	0/29/0	0/29/0	3/20/6

Table 11. *p*-values of the IEEE CEC2017 test function set for 100D.

Problems	TLBO	SSA	INFO	DE	DBO	SO	CLPSO	BDE	BESD	GO
CEC2017_F1	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/+$	$7.74\times10^{-6}/+$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$7.04\times10^{-7}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.05\times10^{-1}/=$
CEC2017_F3	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$9.12\times10^{-1}/=$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.69\times10^{-11}/-$
CEC2017_F4	$3.02\times10^{-11}/-$	$1.41\times10^{-9}/-$	$1.47\times10^{-7}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.96\times10^{-10}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$7.73\times10^{-2}/=$
CEC2017_F5	$6.91\times10^{-4}/-$	$9.76\times10^{-10}/-$	$3.47\times10^{-10}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.33\times10^{-1}/=$	$3.02\times10^{-11}/-$	$3.34\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$
CEC2017_F6	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$7.39\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.34\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.56\times10^{-4}/+$
CEC2017_F7	$3.02\times10^{-11}/-$	$6.70\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.08\times10^{-8}/+$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.34\times10^{-11}/-$
CEC2017_F8	$4.11\times10^{-7}/-$	$2.44\times10^{-9}/-$	$8.89\times10^{-10}/-$	$3.02\times10^{-11}/-$	$3.69\times10^{-11}/-$	$3.18\times10^{-1}/=$	$3.02\times10^{-11}/-$	$5.49\times10^{-11}/-$	$3.02\times10^{-11}/-$	$7.39\times10^{-11}/-$
CEC2017_F9	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$5.40\times10^{-1}/=$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$2.32\times10^{-2}/+$
CEC2017_F10	$3.02\times10^{-11}/-$	$8.99\times10^{-11}/+$	$4.50\times10^{-11}/+$	$3.02\times10^{-11}/-$	$4.31\times10^{-8}/+$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.87\times10^{7}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$
CEC2017_F11	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.07\times10^{-9}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.34\times10^{-11}/-$
CEC2017_F12	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.96\times10^{-8}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.69\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$1.61\times10^{-6}/-$
CEC2017_F13	$2.83\times10^{-8}/-$	$3.02\times10^{-11}/-$	$5.61\times10^{-5}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$2.01\times10^{-1}/=$
CEC2017_F14	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$6.28\times10^{-6}/-$
CEC2017_F15	$7.20\times10^{-5}/-$	$3.02\times10^{-11}/-$	$3.34\times10^{-3}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$7.39\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$4.06\times10^{-2}/-$
CEC2017_F16	$2.12\times10^{-1}/=$	$2.43\times10^{-5}/-$	$4.55\times10^{-1}/=$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$2.15\times10^{-6}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$
CEC2017_F17	$2.34\times10^{-1}/=$	$1.60\times10^{-3}/-$	$1.11\times10^{-4}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$9.05\times10^{-2}/=$	$3.02\times10^{-11}/-$	$2.87\times10^{-10}/-$	$3.34\times10^{-11}/-$	$1.01\times10^{-8}/-$
CEC2017_F18	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$4.80\times10^{-7}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$4.86\times10^{-3}/-$
CEC2017_F19	$2.61\times10^{-2}/-$	$3.02\times10^{-11}/-$	$8.29\times10^{-6}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$4.98\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$3.02\times10^{-11}/-$	$4.29\times10^{-1}/=$
CEC2017_F20	$4.98\times10^{-11}/-$	$3.79\times10^{-1}/=$	$7.17\times10^{-1}/=$	$3.02\times10^{-11}/-$	$6.38\times10^{-3}/-$	$9.76\times10^{-10}/-$	$1.34\times10^{5}/-$	$1.60\times10^{7}/-$	$1.61\times10^{-10}/-$	$2.15\times10^{-10}/-$

Table 11. *Cont.*

Problems	TLBO	SSA	INFO	DE	DBO	SO	CLPSO	BDE	BESD	GO
CEC2017_F21	$3.08 \times 10^{-8}/-$	$9.76 \times 10^{-10}/-$	$2.61 \times 10^{-10}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.63 \times 10^{-1}/=$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.34 \times 10^{-11}/-$
CEC2017_F22	$5.57 \times 10^{-10}/-$	$3.02 \times 10^{-11}/+$	$1.21 \times 10^{-10}/+$	$3.02 \times 10^{-11}/-$	$5.53 \times 10^{-8}/+$	$3.34 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$1.20 \times 10^{-8}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$
CEC2017_F23	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$1.25 \times 10^{-7}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$2.92 \times 10^{-9}/-$
CEC2017_F24	$3.02 \times 10^{-11}/-$	$3.69 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$1.61 \times 10^{-10}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$1.09 \times 10^{-10}/-$
CEC2017_F25	$3.02 \times 10^{-11}/-$	$1.29 \times 10^{-9}/-$	$3.81 \times 10^{-7}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$6.07 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$1.50 \times 10^{-2}/-$
CEC2017_F26	$3.02 \times 10^{-11}/-$	$8.48 \times 10^{-9}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$2.15 \times 10^{-10}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$2.37 \times 10^{-10}/-$
CEC2017_F27	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$2.84 \times 10^{1}/=$
CEC2017_F28	$3.02 \times 10^{-11}/-$	$6.52 \times 10^{-9}/-$	$2.32 \times 10^{-6}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$4.55 \times 10^{-1}/=$
CEC2017_F29	$4.11 \times 10^{-7}/-$	$5.49 \times 10^{-11}/-$	$1.64 \times 10^{-5}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$5.19 \times 10^{-2}/=$	$3.02 \times 10^{-11}/-$	$6.70 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$1.17 \times 10^{-9}/-$
CEC2017_F30	$3.34 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$7.04 \times 10^{-7}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$3.02 \times 10^{-11}/-$	$2.84 \times 10^{-4}/-$
+/−/=	0/27/2	3/25/1	3/23/3	0/29/0	2/27/0	1/22/6	0/29/0	0/29/0	0/29/0	2/21/6

Table 12. Friedman mean rank test results for the IEEE CEC2017 test function set.

Dimensions	30D		50D		100D	
Algorithms	**Mean Rank**	**Final Rank**	**Mean Rank**	**Final Rank**	**Mean Rank**	**Final Rank**
TLBO	5.05	3	5.31	5	5.77	6
SSA	7.08	8	6.08	7	5.13	5
INFO	5.51	5	4.82	3	3.99	3
DE	6.51	7	7.02	8	7.67	8
DBO	8.91	10	8.69	10	8.49	9
SO	5.58	6	5.10	4	5.01	4
CLPSO	9.60	11	9.84	11	9.87	11
BDE	5.30	4	5.61	6	5.78	7
BESD	7.79	9	8.39	9	8.77	10
GO	3.03	2	3.38	2	3.63	2
CODGBGO	1.65	1	1.75	1	1.87	1

4.3.6. Expanded Analysis

In this section, the performance differences between CODGBGO and the superior algorithms are primarily tested. The IEEE CEC2020 test function set is used for the experiments, with test dimensions of 10D and 20D.

Firstly, the algorithm's performance is evaluated using the mean and standard deviation metrics, and the results are shown in Table 13. We can see from Table 13 that when the test dimension is 10D, CODGBGO ranks first on 80% of the test functions and achieves good solution performance. Among them, it is weaker than ALSHADE on test function F8 and ALSHADE, IMODE, and LSHADE on test function F9, which shows that CODGBGO still suffers from low convergence performance when dealing with specific problems compared to the best existing optimization algorithms, and this is attributed to the fact that there is still room for improvement in the exploration strategy and the exploitation strategy proposed in this paper. However, from a comprehensive point of view, CODGBGO is undeniably an excellent algorithm. Meanwhile, when the test is 20D, consistent with the above analysis, CODGBGO still has the best comprehensive performance, but there is still some room for improvement.

Secondly, the Wilcoxon rank sum test is used to compare the differences between CODGBGO and the superior algorithms, and the experimental results are shown in Table 14. From Table 14, it can be observed that for a test dimension of 10D, CODGBGO significantly outperforms ALSHADE in five test functions and significantly outperforms IMODE and LSHADE in four test functions. For a test dimension of 20D, CODGBGO significantly outperforms ALSHADE and LSHADE in seven test functions and significantly outperforms IMODE in six test functions, demonstrating its strong competitiveness. Secondly, the Friedman mean rank test is used to rank the experimental results, and the results are

presented in Table 15. From Table 15, it can be seen that CODGBGO achieves the top ranking in all test dimensions, outperforming other superior algorithms and demonstrating higher solution stability and performance.

Table 13. Comparison of numerical results on the IEEE CEC2020 test function.

Dimensions		10D				20D			
		Algorithms				Algorithms			
Problems		ALSHADE	IMODE	LSHADE	CODGBGO	ALSHADE	IMODE	LSHADE	CODGBGO
CEC2020_F1	Mean	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$	$\mathbf{1.00 \times 10^{2}}$
	Std	0.00×10^{0}	$\mathbf{2.64 \times 10^{-15}}$	0.00×10^{0}	1.50×10^{-10}	$\mathbf{2.04 \times 10^{-14}}$	5.91×10^{-12}	1.20×10^{-12}	9.49×10^{-5}
CEC2020_F2	Mean	1.15×10^{3}	1.16×10^{3}	1.12×10^{3}	$\mathbf{1.11 \times 10^{3}}$	1.21×10^{3}	1.30×10^{3}	1.28×10^{3}	$\mathbf{1.12 \times 10^{3}}$
	Std	6.99×10^{1}	5.43×10^{1}	2.25×10^{1}	$\mathbf{5.19 \times 10^{0}}$	4.01×10^{1}	9.85×10^{1}	8.29×10^{1}	$\mathbf{8.88 \times 10^{0}}$
CEC2020_F3	Mean	7.13×10^{2}	7.17×10^{2}	7.12×10^{2}	$\mathbf{7.12 \times 10^{2}}$	7.26×10^{2}	7.29×10^{2}	7.27×10^{2}	$\mathbf{7.26 \times 10^{2}}$
	Std	8.45×10^{-1}	2.28×10^{0}	9.45×10^{-1}	$\mathbf{1.36 \times 10^{1}}$	1.43×10^{1}	1.98×10^{0}	7.90×10^{-1}	$\mathbf{3.47 \times 10^{-13}}$
CEC2020_F4	Mean	1.90×10^{3}	1.90×10^{3}	1.90×10^{3}	$\mathbf{1.90 \times 10^{3}}$	1.90×10^{3}	1.90×10^{3}	1.90×10^{3}	$\mathbf{1.90 \times 10^{3}}$
	Std	1.07×10^{-1}	2.25×10^{-1}	$\mathbf{7.01 \times 10^{-2}}$	7.91×10^{-2}	1.89×10^{-1}	2.84×10^{-1}	2.13×10^{-1}	$\mathbf{4.63 \times 10^{-13}}$
CEC2020_F5	Mean	1.72×10^{3}	1.73×10^{3}	1.70×10^{3}	$\mathbf{1.70 \times 10^{3}}$	2.16×10^{3}	2.07×10^{3}	2.01×10^{3}	$\mathbf{1.93 \times 10^{3}}$
	Std	3.07×10^{1}	4.53×10^{1}	1.42×10^{0}	$\mathbf{3.33 \times 10^{-1}}$	2.20×10^{2}	1.02×10^{2}	$\mathbf{2.75 \times 10^{0}}$	2.82×10^{0}
CEC2020_F6	Mean	1.60×10^{3}	1.60×10^{3}	1.60×10^{3}	$\mathbf{1.60 \times 10^{3}}$	1.60×10^{3}	1.60×10^{3}	1.60×10^{3}	$\mathbf{1.60 \times 10^{3}}$
	Std	3.00×10^{-1}	2.03×10^{0}	3.00×10^{-1}	$\mathbf{2.88 \times 10^{-1}}$	$\mathbf{2.59 \times 10^{-1}}$	3.44×10^{-1}	3.53×10^{-1}	0.00×10^{0}
CEC2020_F7	Mean	2.10×10^{3}	2.10×10^{3}	2.10×10^{3}	$\mathbf{2.10 \times 10^{3}}$	2.22×10^{3}	2.16×10^{3}	$\mathbf{2.15 \times 10^{3}}$	2.16×10^{3}
	Std	2.60×10^{-1}	3.11×10^{-1}	3.18×10^{1}	$\mathbf{1.79 \times 10^{-1}}$	1.02×10^{2}	6.29×10^{1}	5.23×10^{1}	$\mathbf{3.01 \times 10^{0}}$
CEC2020_F8	Mean	$\mathbf{2.30 \times 10^{3}}$	2.31×10^{3}	2.31×10^{3}	2.31×10^{3}	$\mathbf{2.31 \times 10^{3}}$	$\mathbf{2.31 \times 10^{3}}$	$\mathbf{2.31 \times 10^{3}}$	$\mathbf{2.31 \times 10^{3}}$
	Std	2.50×10^{1}	4.55×10^{-13}	$\mathbf{4.31 \times 10^{-13}}$	$\mathbf{4.31 \times 10^{-13}}$	0.00×10^{0}	7.97×10^{-13}	$\mathbf{6.86 \times 10^{-13}}$	0.00×10^{0}
CEC2020_F9	Mean	2.72×10^{3}	$\mathbf{2.66 \times 10^{3}}$	2.68×10^{3}	2.74×10^{3}	2.81×10^{3}	2.82×10^{3}	$\mathbf{2.81 \times 10^{3}}$	2.84×10^{3}
	Std	4.26×10^{1}	1.13×10^{2}	9.39×10^{1}	$\mathbf{4.35 \times 10^{0}}$	4.83×10^{0}	6.04×10^{1}	$\mathbf{1.66 \times 10^{0}}$	1.33×10^{1}
CEC2020_F10	Mean	2.92×10^{3}	2.91×10^{3}	2.91×10^{3}	$\mathbf{2.90 \times 10^{3}}$	2.92×10^{3}	2.92×10^{3}	2.91×10^{3}	$\mathbf{2.91 \times 10^{3}}$
	Std	2.33×10^{1}	1.72×10^{1}	1.72×10^{1}	$\mathbf{4.54 \times 10^{-1}}$	1.09×10^{1}	1.69×10^{1}	$\mathbf{1.72 \times 10^{-2}}$	3.05×10^{-2}
Mean Rank		2.4	2.8	2.1	**1.4**	2.3	2.9	2.2	**1.4**
Final Rank		3	4	2	**1**	3	4	2	**1**
Rank First		2	2	1	**8**	1	2	4	**8**

Table 14. *p*-values of the IEEE CEC2020 test function set.

Dimensions	10D			20D		
	Algorithms			Algorithms		
Problems	ALSHADE	IMODE	LSHADE	ALSHADE	IMODE	LSHADE
CEC2020_F1	$1.63 \times 10^{-11}/+$	$1.63 \times 10^{-11}/+$	$1.63 \times 10^{-11}/+$	$1.21 \times 10^{-12}/+$	$3.16 \times 10^{-12}/+$	$7.82 \times 10^{-12}/+$
CEC2020_F2	$1.42 \times 10^{-4}/-$	$2.36 \times 10^{-9}/-$	$7.39 \times 10^{-4}/-$	$1.37 \times 10^{-8}/-$	$9.26 \times 10^{-10}/-$	$2.15 \times 10^{-12}/-$
CEC2020_F3	$4.87 \times 10^{-4}/-$	$7.41 \times 10^{-11}/-$	$4.34 \times 10^{-2}/-$	$1.22 \times 10^{-12}/-$	$1.01 \times 10^{-12}/-$	$3.90 \times 10^{-13}/-$
CEC2020_F4	$2.40 \times 10^{-11}/-$	$2.40 \times 10^{-11}/-$	$5.93 \times 10^{-11}/-$	$7.15 \times 10^{-9}/-$	$3.36 \times 10^{-11}/-$	$1.21 \times 10^{-12}/-$
CEC2020_F5	$5.18 \times 10^{-5}/-$	$3.87 \times 10^{-4}/-$	$2.21 \times 10^{-3}/-$	$2.34 \times 10^{-5}/-$	$3.73 \times 10^{-11}/-$	$4.29 \times 10^{-14}/-$
CEC2020_F6	$3.47 \times 10^{-1}/=$	$7.47 \times 10^{-2}/=$	$6.93 \times 10^{-1}/=$	$1.81 \times 10^{-7}/-$	$2.05 \times 10^{-5}/-$	$1.21 \times 10^{-12}/-$
CEC2020_F7	$1.47 \times 10^{-1}/=$	$1.22 \times 10^{-1}/=$	$3.53 \times 10^{-1}/=$	$2.06 \times 10^{-2}/-$	$1.55 \times 10^{-1}/=$	$2.34 \times 10^{-5}/+$
CEC2020_F8	$1.61 \times 10^{-1}/=$	$3.34 \times 10^{-1}/=$	$3.34 \times 10^{-1}/=$	$3.34 \times 10^{-1}/=$	$3.34 \times 10^{-1}/=$	$3.34 \times 10^{-1}/=$
CEC2020_F9	$1.55 \times 10^{-8}/+$	$1.64 \times 10^{-6}/+$	$3.61 \times 10^{-11}/+$	$9.92 \times 10^{-11}/+$	$1.22 \times 10^{-2}/+$	$3.02 \times 10^{-11}/+$
CEC2020_F10	$6.70 \times 10^{-3}/-$	$2.71 \times 10^{-1}/=$	$6.25 \times 10^{-1}/=$	$1.72 \times 10^{-12}/-$	$1.07 \times 10^{-9}/-$	$1.35 \times 10^{-12}/-$
$+/-/=$	2/5/3	2/4/4	2/4/4	2/7/1	2/6/2	3/7/0

Table 15. Friedman mean rank test results for IEEE CEC2020 test function set.

Dimensions	10D		20D	
Algorithms	Mean Rank	Final Rank	Mean Rank	Final Rank
ALSHADE	2.58	3	2.55	2
IMODE	2.78	4	2.87	4
SHADE	2.35	2	2.59	3
CODGBGO	2.28	1	1.99	1

4.4. Results for Feature Selection Problems

It has been verified in previous experiments that the proposed CODGBGO possesses efficient performance in dealing with numerical optimization problems. In this section, the performance of the proposed CODGBGO for discretized optimization problems is evaluated by using CODGBGO for solving feature selection problems. The feature selection problem can be defined as the dimensionality reduction of the original dataset features to improve the classification accuracy. The feature selection problem can be viewed as a

discretized optimization problem and the CODGBGO algorithm is proposed in this paper to solve this challenging problem.

4.4.1. Establishment of an Optimization Model

The feature selection problem is a multidimensional optimization problem where the objective is to obtain the highest classification accuracy using the minimum number of features. The fitness function of the feature selection model is defined as follows:

$$min\ f(\mathrm{x}) = \lambda_1 \cdot error + \lambda_2 \cdot R/n, \tag{25}$$

where x denotes the selected feature subset, *error* denotes the classification error of the selected feature subset, R denotes the selected feature subset size, and n denotes the number of features in the original dataset, $\lambda_1 \in [0, 1]$, $\lambda_2 = 1 - \lambda_1$, In this paper, λ_1 takes the value of 0.9.

To deal with the feature selection problem, CODGBGO was discretized by defining each individual in the population as a binary variable. For example, an individual in the population $X_i = (x_{i,1}, x_{i,j}, \ldots x_{i,D})$, where D denotes the dimension of the problem to be solved and its value is the number of features in the original dataset, $x_{i,j}$ is a binary variable and $x_{i,j}$ equals 1 indicates that the jth feature is selected, otherwise, the jth feature is not selected. During the initialization phase of the population, generate N individuals, variables are randomly generated in the interval [0, 1] and then converted to binary variables by a threshold of 0.5. The definition is as follows:

$$x_{i,j} = \begin{cases} 1, & x_{i,j} > 0.5 \\ 0, & x_{i,j} \leq 0.5 \end{cases} \quad i = 1, 2, \ldots, N;\ j = 1, 2, \ldots, D. \tag{26}$$

4.4.2. Experimental Analyses

UCI is a machine learning database proposed by the University of California Irvine. It can be obtained at http://archive.ics.uci.edu/mL/index.php (accessed on 4 March 2024), In this paper, 18 datasets were selected to evaluate the performance of the proposed CODGBGO in dealing with discrete optimization problems. The dataset description information is shown in Table 16. The comparison algorithms are PSO, DE, GWO, MVO, WOA, ABO, BOA, HHO, EO, and GO.

Table 16. The information from 18 datasets.

Datasets	Number of Features	Number of Classifications	Dataset Size
IonosphereEW	34	2	351
Breastcancer	9	2	699
BreastEW	30	2	569
Congress	16	2	435
Wine	13	3	178
Vote	16	2	435
Vehicle	18	4	846
Exactly	13	2	1000
Glass	9	7	214
HeartEW	13	2	270
Landsat	36	6	2000
Lymphography	18	4	148
Zoo	16	7	101
WDBC	30	2	569
SonarEW	60	2	208
Libras	90	15	360
Spectf	44	2	267
MUSK	166	2	476

In this paper, the selected subset of features is used to calculate the classification accuracy using the K-nearest neighbor (KNN) algorithm, with K set to 5. Subsequently, the performance was evaluated using 5-fold cross-validation and the dataset is divided into five parts: one for testing and four for training. The accuracy of feature subset prediction can be effectively calculated by K-fold cross-validation. Meanwhile, 10 algorithms were used to experimentally compare with CODGBGO to evaluate the performance of CODGBGO. The population size was set to 60, the maximum number of iterations was set to 100, and each experiment was executed independently for 30 times. The following metrics were used to evaluate the performance of CODGBGO:

Fitness values: They are obtained from 30 independent runs of the experiment and contain the best value, the worst value, and the mean value.

Classification accuracy: it is obtained by calculating the average classification accuracy of 30 independent experiments. The classification accuracy for each experiment is obtained by solving the classification accuracy by experimenting with KNN on a selected subset of features.

Feature subset size: it is obtained by calculating the average feature subset size of 30 independent experiments.

Rank: It represents the ranking in different evaluation criteria (e.g., mean, best, worst) and is used to visualize the performance of the algorithm.

Table 17 shows the best fitness value, mean fitness value, worst fitness value, and ranking on each metric for 30 independent runs of the algorithm on different datasets. As can be seen in Table 17, CODGBGO ranked first on all 18 datasets in the best fitness value metric, with a 100% win rate. Meanwhile, on the mean fitness value metric, it ranked first on 17 datasets and was weaker than DE only on the Zoo dataset, with an average ranking of 1.0556. On the worst fitness value metric, it ranked first on 16 datasets, weaker than GWO on the IonosphereEW dataset, and weaker than GO on the Landsat dataset, with an average ranking of 1.1111. The final ranking of the proposed CODGBGO is first in all three metrics: best fitness value, average fitness value, and worst fitness value. The above analysis confirms that, due to the introduction of exploitation strategy and exploration strategy, CODGBGO has more efficient performance in dealing with feature selection problems. Meanwhile, compared with the comparison algorithms, CODGBGO possesses higher expandability, which also means that CODGBGO may also be very competitive when dealing with other engineering optimization problems. However, it inevitably falls into a local optimum when solving a specific problem and loses its global optimization capability. This also indicates that there is still room for improvement in the exploration strategy and exploitation strategy in this paper.

Figure 8 shows the distribution of the best fitness values generated by CODGBGO over 30 independently run experiments. As can be seen in Figure 8, the stability of CODGBGO outperforms the comparison algorithms on most of the datasets. This also shows that CODGBGO has higher solution stability in solving the feature selection problem and can be used as an effective method to solve the feature selection problem. In addition to the accuracy and stability of the solution, the speed of convergence of the algorithm in solving real-world problems is crucial. Figure 9 shows the average convergence curve of CODGBGO in solving the feature selection problem. From Figure 9, it can be seen that its convergence speed and accuracy are better than those of the other comparison algorithms on most of the datasets.

Through the above analysis, CODGBGO has higher computational accuracy, better stability, and faster convergence speed in solving the feature selection problem. Table 18 shows the Wilcoxon's rank sum test results of the algorithm in solving different datasets, where '+', '−', and '=' have the meanings as described earlier with a significance factor of 0.05. As can be seen from Table 18, CODGBGO significantly outperforms PSO, MVO, WOA, ABO, BOA, HHO, and EO on 18 datasets, DE and GWO on 17 datasets, and GO on 15 datasets. This also shows that CODGBGO is an effective method for solving the feature selection problem.

Table 17. Comparison of algorithms for feature selection.

Datasets	Metric	PSO	DE	GWO	MVO	WOA	ABO	BOA	HHO	EO	GO	CODGBGO
IonosphereEW	Best	0.0878	0.0562	0.0345	0.0492	0.0573	0.0503	0.0375	0.0503	0.0603	0.0375	0.0246
	Mean	0.1278	0.0858	0.0545	0.0826	0.0919	0.0865	0.0641	0.1017	0.0714	0.0708	0.0476
	Worst	0.1797	0.1143	0.0750	0.1025	0.1245	0.1183	0.0977	0.1521	0.0889	0.0977	0.0790
	Rank	(11/11/11)	(8/7/7)	(2/2/1)	(5/6/6)	(9/9/9)	(6/8/8)	(4/3/5)	(6/10/10)	(10/5/3)	(3/4/4)	(1/1/2)
Breastcancer	Best	0.0592	0.0528	0.0592	0.0592	0.0611	0.0592	0.0639	0.0416	0.0463	0.0546	0.0398
	Mean	0.0626	0.0547	0.0603	0.0607	0.0684	0.0612	0.0696	0.0672	0.0468	0.0569	0.0422
	Worst	0.0805	0.0592	0.0722	0.0722	0.0916	0.0740	0.0787	0.0879	0.0481	0.0657	0.0463
	Rank	(6/8/9)	(4/3/3)	(6/5/5)	(6/6/5)	(10/10/11)	(6/7/7)	(11/11/8)	(2/9/10)	(3/2/2)	(5/4/4)	(1/1/1)
BreastEW	Best	0.0439	0.0333	0.0472	0.0413	0.0419	0.0406	0.0452	0.0333	0.0293	0.0346	0.0246
	Mean	0.0532	0.0488	0.0535	0.0602	0.0597	0.0574	0.0601	0.0531	0.0415	0.0489	0.0325
	Worst	0.0672	0.0672	0.0632	0.0752	0.0798	0.0704	0.0719	0.0706	0.0619	0.0719	0.0413
	Rank	(9/6/4)	(3/3/4)	(11/7/3)	(7/11/10)	(8/9/11)	(6/8/6)	(10/10/8)	(3/5/7)	(2/2/2)	(5/4/8)	(1/1/1)
Congress	Best	0.0166	0.0250	0.0394	0.0601	0.0269	0.0623	0.0582	0.0476	0.0313	0.0353	0.0063
	Mean	0.0215	0.0389	0.0452	0.0747	0.0275	0.0744	0.0788	0.0551	0.0371	0.0371	0.0063
	Worst	0.0313	0.0623	0.0519	0.0914	0.0353	0.0787	0.0974	0.0996	0.0373	0.0394	0.0063
	Rank	(2/2/2)	(3/6/7)	(7/7/6)	(10/10/9)	(4/3/3)	(11/9/8)	(9/11/10)	(8/8/11)	(5/4/4)	(6/5/5)	(1/1/1)
Wine	Best	0.0308	0.0231	0.0565	0.0231	0.0411	0.0565	0.0308	0.0231	0.0231	0.0565	0.0231
	Mean	0.0717	0.0427	0.0711	0.0449	0.0687	0.0606	0.0639	0.0664	0.0420	0.0656	0.0259
	Worst	0.1336	0.0642	0.1079	0.1053	0.1851	0.0745	0.1079	0.1387	0.0899	0.0899	0.0385
	Rank	(6/11/9)	(1/3/2)	(9/10/7)	(1/4/6)	(8/9/11)	(9/5/3)	(6/6/7)	(1/8/10)	(1/2/4)	(9/7/4)	(1/1/1)
Vote	Best	0.0332	0.0269	0.0269	0.0228	0.0332	0.0166	0.0746	0.0394	0.0269	0.0250	0.0063
	Mean	0.0399	0.0334	0.0269	0.0388	0.0367	0.0175	0.0932	0.0557	0.0269	0.0280	0.0067
	Worst	0.0541	0.0438	0.0269	0.0769	0.0435	0.0435	0.1388	0.0996	0.0269	0.0500	0.0188
	Rank	(8/9/8)	(5/6/6)	(5/3/2)	(3/8/9)	(8/7/4)	(2/2/4)	(11/11/11)	(10/10/10)	(5/3/2)	(4/5/7)	(1/1/1)
Vehicle	Best	0.2575	0.2519	0.2563	0.2468	0.2568	0.2626	0.2621	0.2572	0.2568	0.2514	0.2308
	Mean	0.2730	0.2710	0.2770	0.2660	0.2933	0.2857	0.2988	0.2966	0.2687	0.2746	0.2463
	Worst	0.3100	0.2950	0.3047	0.2947	0.3272	0.3105	0.3427	0.3334	0.3052	0.2943	0.2623
	Rank	(9/5/7)	(4/4/4)	(5/7/5)	(2/2/3)	(6/9/9)	(11/8/8)	(10/11/11)	(8/10/10)	(6/3/6)	(3/6/2)	(1/1/1)
Exactly	Best	0.0462	0.0462	0.0462	0.0462	0.0462	0.0462	0.2177	0.0628	0.0462	0.0462	0.0462
	Mean	0.1256	0.1047	0.2058	0.2006	0.2537	0.2183	0.2886	0.2662	0.1711	0.0781	0.0603
	Worst	0.3040	0.2704	0.2912	0.3143	0.2912	0.2912	0.2912	0.3387	0.2905	0.2912	0.2635
	Rank	(1/4/9)	(1/3/2)	(1/7/4)	(1/6/10)	(1/9/4)	(1/8/4)	(11/11/4)	(10/10/11)	(1/5/3)	(1/2/4)	(1/1/1)
Glass	Best	0.2373	0.2905	0.2587	0.2698	0.2690	0.2373	0.2373	0.2484	0.2802	0.2690	0.1833
	Mean	0.2438	0.2982	0.2745	0.2941	0.2952	0.2469	0.2890	0.2799	0.2952	0.2838	0.1894
	Worst	0.2810	0.3230	0.3016	0.3452	0.3333	0.2802	0.3341	0.3238	0.3119	0.3016	0.2476
	Rank	(2/2/3)	(11/11/7)	(6/4/4)	(9/8/11)	(7/10/9)	(2/3/2)	(2/7/10)	(5/5/8)	(10/9/6)	(7/6/4)	(1/1/1)
HeartEW	Best	0.1897	0.1641	0.1231	0.1231	0.1038	0.1564	0.1551	0.1808	0.1308	0.1641	0.0974
	Mean	0.2287	0.2052	0.1332	0.1754	0.1324	0.1928	0.2210	0.2291	0.1421	0.1742	0.1038
	Worst	0.2808	0.2615	0.1821	0.2218	0.1731	0.2385	0.2654	0.3038	0.1731	0.3308	0.1385
	Rank	(11/10/9)	(8/8/7)	(3/3/4)	(3/6/5)	(2/2/2)	(7/7/6)	(6/9/8)	(10/11/10)	(5/4/2)	(8/5/11)	(1/1/1)
Landsat	Best	0.1117	0.1194	0.1122	0.1088	0.1061	0.1059	0.1292	0.1183	0.1083	0.1048	0.1010
	Mean	0.1214	0.1325	0.1196	0.1201	0.1204	0.1160	0.1395	0.1436	0.1160	0.1122	0.1109
	Worst	0.1343	0.1495	0.1268	0.1339	0.1386	0.1240	0.1498	0.1623	0.1257	0.1228	0.1240
	Rank	(7/8/7)	(10/9/9)	(8/5/5)	(6/6/6)	(4/7/8)	(3/4/2)	(11/10/10)	(9/11/11)	(5/3/4)	(2/2/1)	(1/1/2)
Lymphography	Best	0.1375	0.0898	0.1042	0.1065	0.0699	0.1264	0.1153	0.0755	0.0644	0.0644	0.0533
	Mean	0.1890	0.1189	0.1181	0.1581	0.1152	0.1598	0.1741	0.1314	0.0860	0.0929	0.0649
	Worst	0.2506	0.1741	0.1519	0.2251	0.1885	0.1941	0.2418	0.2052	0.1153	0.1320	0.0954
	Rank	(11/11/11)	(6/6/5)	(7/5/4)	(8/8/9)	(4/4/6)	(10/9/7)	(9/10/10)	(5/7/8)	(2/2/2)	(2/3/3)	(1/1/1)
Zoo	Best	0.0375	0.0438	0.0375	0.0375	0.0375	0.0375	0.0625	0.0500	0.0438	0.0438	0.0375
	Mean	0.0875	0.0505	0.0667	0.0690	0.0668	0.0607	0.1323	0.0938	0.0746	0.0614	0.0516
	Worst	0.1475	0.0950	0.1213	0.1525	0.1075	0.0888	0.1788	0.1725	0.0825	0.0825	0.0825
	Rank	(1/9/8)	(7/1/5)	(1/5/7)	(1/7/9)	(1/6/6)	(1/3/4)	(11/11/11)	(10/10/10)	(7/8/1)	(7/4/1)	(1/2/1)
WDBC	Best	0.0372	0.0791	0.0385	0.0552	0.0465	0.0532	0.0419	0.0545	0.0498	0.0498	0.0372
	Mean	0.0527	0.0877	0.0457	0.0943	0.0642	0.0601	0.0575	0.0781	0.0528	0.0631	0.0393
	Worst	0.0778	0.1004	0.0678	0.1269	0.0830	0.0750	0.0758	0.0971	0.0704	0.0771	0.0578
	Rank	(1/3/7)	(11/10/10)	(3/2/2)	(10/11/11)	(5/8/8)	(8/6/4)	(4/5/5)	(9/9/9)	(6/4/3)	(6/7/6)	(1/1/1)
SonarEW	Best	0.0536	0.0620	0.0167	0.1075	0.0822	0.0789	0.0859	0.0672	0.0320	0.0470	0.0117
	Mean	0.1225	0.0922	0.0609	0.1457	0.1127	0.1293	0.1111	0.1295	0.0546	0.0893	0.0192
	Worst	0.1887	0.1209	0.1195	0.1937	0.1584	0.1684	0.1381	0.1867	0.0995	0.1364	0.0300
	Rank	(5/8/10)	(6/5/4)	(2/3/3)	(11/11/11)	(9/7/7)	(8/9/8)	(10/6/6)	(7/10/9)	(3/2/2)	(4/4/5)	(1/1/1)
Libras	Best	0.1389	0.1625	0.1269	0.0992	0.1428	0.1644	0.1303	0.1494	0.1156	0.1447	0.0667
	Mean	0.1770	0.1979	0.1581	0.1376	0.1885	0.2185	0.1727	0.1984	0.1435	0.1783	0.0913
	Worst	0.2103	0.2122	0.1833	0.1569	0.2439	0.2492	0.1939	0.2411	0.1781	0.2050	0.1164
	Rank	(6/6/7)	(10/9/8)	(4/4/4)	(2/2/2)	(7/8/10)	(11/11/11)	(5/5/5)	(9/10/9)	(3/3/3)	(8/7/6)	(1/1/1)
Spectf	Best	0.0681	0.0602	0.0544	0.1076	0.0963	0.0985	0.1145	0.1076	0.0669	0.0511	0.0511
	Mean	0.1073	0.1140	0.0926	0.1416	0.1506	0.1213	0.1484	0.1469	0.0930	0.0817	0.0748
	Worst	0.1496	0.1496	0.1360	0.1745	0.1913	0.1551	0.1689	0.1936	0.1280	0.1304	0.1111
	Rank	(6/5/5)	(4/6/5)	(3/3/4)	(9/8/9)	(7/11/10)	(8/7/7)	(11/10/8)	(9/9/11)	(5/4/2)	(1/2/3)	(1/1/1)
MUSK	Best	0.0610	0.0792	0.0342	0.0510	0.0503	0.0692	0.0963	0.0606	0.0328	0.0676	0.0199
	Mean	0.1020	0.0981	0.0681	0.0843	0.0816	0.0960	0.1424	0.0894	0.0516	0.0913	0.0366
	Worst	0.1411	0.1272	0.1106	0.1175	0.1138	0.1243	0.1703	0.1234	0.0728	0.1210	0.0591
	Rank	(7/10/10)	(10/9/9)	(3/3/3)	(5/5/5)	(4/4/4)	(9/8/8)	(11/11/11)	(6/6/7)	(2/2/2)	(8/7/6)	(1/1/1)
Mean Rank	Best	6.0556	6.2222	4.7778	5.5000	5.7778	6.6111	8.4444	7.0556	4.5000	4.9444	1.0000
	Mean	7.1111	6.0556	4.7222	6.9444	7.3333	6.7778	8.7778	8.7778	3.7222	4.6667	1.0556
	Worst	7.5556	5.7778	4.0556	7.5556	7.3333	5.9444	8.2222	9.5000	2.9444	4.6667	1.1111
Final Rank	Best	7	8	3	5	6	9	11	10	2	4	1
	Mean	8	5	4	7	9	6	10	10	2	3	1
	Worst	8	5	3	8	7	6	10	11	2	4	1

In addition, Tables 19 and 20 show the average classification accuracy and the average feature subset size of the algorithms when solving different datasets, respectively. Figure 10 shows the average accuracy ranking of the algorithms when solving different datasets. Table 21 shows the running time for each dataset. As can be seen in Table 19 and Figure 10, CODGBGO finished first on 14 datasets, second on 3 datasets, and just 8th on the BreastEW dataset, with an average ranking of 1.5556 for average accuracy, resulting in a first-place ranking. As can be seen in Table 20, CODGBGO has an average ranking of 4 in terms of the average selected feature subset size, and ultimately ranks second behind EO, but the average classification accuracy of EO is weaker than that of CODGBGO. This shows that EO removes strongly correlated feature attributes in the feature selection process, which leads to a reduction in classification accuracy, which is attributed to the fact that EO's global optimization ability is not as good as CODGBGO's. While CODGBGO is weaker than EO

in terms of the feature subset size, its classification accuracy is better than that of the other algorithms, which confirms that, due to the introduction of the strategies, CODGBGO's has achieved a better balance between the global and local searches and has a stronger global optimization ability.

Table 18. *p*-value of the Wilcoxon rank sum test on different datasets.

Datasets	PSO	DE	GWO	MVO	WOA	ABO	BOA	HHO	EO	GO
IonosphereEW	$2.80 \times 10^{-11}/-$	$6.88 \times 10^{-10}/-$	$1.56 \times 10^{-2}/-$	$1.04 \times 10^{-9}/-$	$7.07 \times 10^{-11}/-$	$1.48 \times 10^{-9}/-$	$6.83 \times 10^{-5}/-$	$2.42 \times 10^{-10}/-$	$1.26 \times 10^{-8}/-$	$7.75 \times 10^{-7}/-$
Breastcancer	$6.85 \times 10^{-12}/-$	$6.83 \times 10^{-12}/-$	$1.86 \times 10^{-12}/-$	$3.38 \times 10^{-12}/-$	$7.62 \times 10^{-12}/-$	$6.81 \times 10^{-12}/-$	$9.99 \times 10^{-12}/-$	$4.77 \times 10^{-8}/-$	$5.61 \times 10^{-8}/-$	$6.29 \times 10^{-12}/-$
BreastEW	$2.21 \times 10^{-11}/-$	$7.23 \times 10^{-10}/-$	$2.05 \times 10^{-11}/-$	$2.78 \times 10^{-11}/-$	$2.33 \times 10^{-11}/-$	$2.83 \times 10^{-11}/-$	$2.30 \times 10^{-11}/-$	$3.80 \times 10^{-10}/-$	$1.24 \times 10^{-6}/-$	$2.67 \times 10^{-10}/-$
Congress	$8.56 \times 10^{-13}/-$	$1.10 \times 10^{-12}/-$	$5.51 \times 10^{-13}/-$	$1.10 \times 10^{-12}/-$	$6.14 \times 10^{-14}/-$	$5.29 \times 10^{-13}/-$	$9.74 \times 10^{-13}/-$	$7.32 \times 10^{-13}/-$	$2.71 \times 10^{-14}/-$	$2.03 \times 10^{-13}/-$
Wine	$2.85 \times 10^{-10}/-$	$5.29 \times 10^{-8}/-$	$7.33 \times 10^{-12}/-$	$1.99 \times 10^{-6}/-$	$1.06 \times 10^{-11}/-$	$3.21 \times 10^{-12}/-$	$1.99 \times 10^{-11}/-$	$2.39 \times 10^{-9}/-$	$3.69 \times 10^{-5}/-$	$4.43 \times 10^{-12}/-$
Vote	$1.33 \times 10^{-12}/-$	$1.50 \times 10^{-12}/-$	$2.71 \times 10^{-14}/-$	$1.57 \times 10^{-12}/-$	$3.00 \times 10^{-13}/-$	$1.61 \times 10^{-12}/-$	$1.63 \times 10^{-12}/-$	$1.54 \times 10^{-12}/-$	$2.71 \times 10^{-14}/-$	$3.86 \times 10^{-13}/-$
Vehicle	$8.02 \times 10^{-10}/-$	$2.58 \times 10^{-10}/-$	$1.28 \times 10^{-10}/-$	$5.52 \times 10^{-8}/-$	$8.10 \times 10^{-11}/-$	$2.71 \times 10^{-11}/-$	$3.84 \times 10^{-11}/-$	$4.48 \times 10^{-11}/-$	$3.56 \times 10^{-7}/-$	$5.60 \times 10^{-11}/-$
Exactly	$7.22 \times 10^{-5}/-$	$4.90 \times 10^{-6}/-$	$1.38 \times 10^{-6}/-$	$1.68 \times 10^{-7}/-$	$2.80 \times 10^{-11}/-$	$2.25 \times 10^{-9}/-$	$3.54 \times 10^{-13}/-$	$1.93 \times 10^{-12}/-$	$1.04 \times 10^{-4}/-$	$5.87 \times 10^{-1}/=$
Glass	$2.42 \times 10^{-10}/-$	$1.25 \times 10^{-12}/-$	$2.12 \times 10^{-12}/-$	$2.34 \times 10^{-12}/-$	$2.61 \times 10^{-12}/-$	$2.87 \times 10^{-10}/-$	$8.94 \times 10^{-12}/-$	$2.96 \times 10^{-12}/-$	$1.17 \times 10^{-12}/-$	$2.01 \times 10^{-12}/-$
HeartEW	$1.08 \times 10^{-11}/-$	$1.13 \times 10^{-11}/-$	$4.53 \times 10^{-11}/-$	$1.72 \times 10^{-11}/-$	$4.17 \times 10^{-9}/-$	$1.11 \times 10^{-11}/-$	$1.12 \times 10^{-11}/-$	$1.16 \times 10^{-11}/-$	$3.47 \times 10^{-11}/-$	$5.04 \times 10^{-12}/-$
Landsat	$1.52 \times 10^{-7}/-$	$4.05 \times 10^{-11}/-$	$9.38 \times 10^{-8}/-$	$1.86 \times 10^{-5}/-$	$9.13 \times 10^{-6}/-$	$6.00 \times 10^{-4}/-$	$2.99 \times 10^{-11}/-$	$3.85 \times 10^{-11}/-$	$6.17 \times 10^{-4}/-$	$3.55 \times 10^{-1}/=$
Lymphography	$2.16 \times 10^{-11}/-$	$3.18 \times 10^{-11}/-$	$1.91 \times 10^{-11}/-$	$2.15 \times 10^{-11}/-$	$1.13 \times 10^{-10}/-$	$2.12 \times 10^{-11}/-$	$2.20 \times 10^{-11}/-$	$3.78 \times 10^{-11}/-$	$5.95 \times 10^{-6}/-$	$3.37 \times 10^{-7}/-$
Zoo	$3.01 \times 10^{-7}/-$	$6.93 \times 10^{-2}/=$	$7.37 \times 10^{-2}/=$	$2.38 \times 10^{-3}/-$	$9.68 \times 10^{-4}/-$	$4.64 \times 10^{-2}/-$	$6.34 \times 10^{-11}/-$	$9.31 \times 10^{-9}/-$	$5.00 \times 10^{-6}/-$	$2.52 \times 10^{-2}/-$
WDBC	$4.35 \times 10^{-9}/-$	$2.99 \times 10^{-12}/-$	$7.54 \times 10^{-9}/-$	$4.21 \times 10^{-12}/-$	$2.89 \times 10^{-11}/-$	$2.38 \times 10^{-11}/-$	$3.48 \times 10^{-10}/-$	$4.37 \times 10^{-12}/-$	$5.34 \times 10^{-9}/-$	$2.26 \times 10^{-11}/-$
SonarEW	$2.88 \times 10^{-11}/-$	$2.87 \times 10^{-11}/-$	$1.06 \times 10^{-9}/-$	$2.88 \times 10^{-11}/-$	$2.89 \times 10^{-11}/-$	$2.89 \times 10^{-11}/-$	$2.89 \times 10^{-11}/-$	$2.89 \times 10^{-11}/-$	$2.87 \times 10^{-11}/-$	$2.84 \times 10^{-11}/-$
Libras	$2.99 \times 10^{-11}/-$	$2.99 \times 10^{-11}/-$	$2.98 \times 10^{-11}/-$	$7.31 \times 10^{-11}/-$	$3.00 \times 10^{-11}/-$	$3.00 \times 10^{-11}/-$	$2.97 \times 10^{-11}/-$	$3.00 \times 10^{-11}/-$	$3.64 \times 10^{-11}/-$	$2.99 \times 10^{-11}/-$
Spectf	$4.06 \times 10^{-8}/-$	$2.26 \times 10^{-8}/-$	$3.75 \times 10^{-4}/-$	$3.66 \times 10^{-11}/-$	$7.28 \times 10^{-11}/-$	$6.79 \times 10^{-11}/-$	$2.97 \times 10^{-11}/-$	$4.01 \times 10^{-11}/-$	$1.88 \times 10^{-4}/-$	$9.16 \times 10^{-2}/=$
MUSK	$3.00 \times 10^{-11}/-$	$3.00 \times 10^{-11}/-$	$2.38 \times 10^{-8}/-$	$4.95 \times 10^{-11}/-$	$1.04 \times 10^{-10}/-$	$3.00 \times 10^{-11}/-$	$3.00 \times 10^{-11}/-$	$3.01 \times 10^{-11}/-$	$3.43 \times 10^{-6}/-$	$3.00 \times 10^{-11}/-$
(+/−/=)	(0/18/0)	(0/17/1)	(0/17/1)	(0/18/0)	(0/18/0)	(0/18/0)	(0/18/0)	(0/18/0)	(0/18/0)	(0/15/3)

Table 19. The mean classification accuracy of the comparison algorithm.

Datasets	PSO	DE	GWO	MVO	WOA	ABO	BOA	HHO	EO	GO	CODGBGO
IonosphereEW	89.1429	94.7143	95.5714	94.5714	91.0476	92.4286	94.9524	90.3333	93.0952	93.9524	96.1905
	11	4	2	5	9	8	3	10	7	6	1
Breastcancer	96.5468	97.7458	96.8825	97.2422	95.2038	96.8585	95.5635	96.0671	98.1775	96.5947	99.0168
	8	3	4	3	8	5	8	8	7	8	8
BreastEW	97.4631	99.2035	96.5782	97.6696	96.1357	96.7257	96.2242	97.4631	97.6696	98.1121	99.1150
	6	1	9	4	11	8	10	6	4	3	2
Congress	99.6935	97.9693	96.4368	94.2912	97.7778	93.3716	93.6782	95.3640	96.6667	97.0115	100.0000
	2	3	7	9	4	11	10	8	6	5	1
Wine	95.9048	99.8095	95.5238	98.2857	95.6190	96.5714	96.8571	96.3810	98.0952	96.1905	100.0000
	9	2	11	3	10	6	5	7	4	8	1
Vote	97.8161	99.2720	97.7011	98.0077	96.7816	98.7739	91.9540	96.0536	97.7011	99.6169	100.0000
	6	3	7	5	9	4	11	10	7	2	1
Vehicle	74.3393	75.0099	72.1893	74.9310	71.4596	71.7949	70.3550	71.7357	73.3136	73.8462	76.2919
	4	2	7	3	10	8	11	9	6	5	1
Exactly	90.8833	94.6667	80.1000	82.1000	73.7167	78.6833	69.3333	72.3000	85.4333	96.0000	98.4000
	4	3	7	6	9	8	11	10	5	2	1
Glass	78.1746	71.2698	73.5714	72.3810	70.0794	77.3016	72.6190	74.2063	72.1429	72.4603	82.6984
	2	10	5	8	11	3	6	4	9	7	1
HeartEW	77.8395	81.7284	87.6543	84.0123	89.5679	82.2222	78.5802	77.1605	87.2840	83.8889	91.9136
	10	8	3	5	2	7	9	11	4	6	1
Landsat	90.3833	89.4500	89.3167	90.6083	89.2417	90.3417	87.4333	87.2000	89.5917	91.0417	90.7000
	4	7	8	3	9	5	10	11	6	1	2
Lymphography	83.2184	91.7241	89.0805	86.3218	90.8046	86.3218	84.2529	88.2759	93.7931	93.5632	96.2069
	11	4	6	8	5	8	10	7	2	3	1
Zoo	94.3333	99.6667	96.5000	97.3333	98.0000	97.3333	90.0000	94.6667	95.6667	96.8333	98.5000
	10	1	7	4	3	4	11	9	8	6	2
WDBC	96.5782	93.3038	95.9292	92.2714	94.1003	94.0708	95.9587	93.3628	95.2212	94.3068	97.0796
	2	10	4	11	7	8	3	9	5	6	1
SonarEW	90.2439	95.2846	95.4472	88.0488	89.9187	88.9431	90.8130	88.6992	95.6911	93.0081	100.0000
	7	4	3	11	8	9	6	10	2	5	1
Libras	84.5833	84.3519	85.0463	89.4444	81.5741	78.5648	83.5648	80.5556	85.8333	83.7037	92.4537
	5	6	4	2	9	11	8	10	3	7	1
Spectf	92.8302	93.1447	92.9560	88.9937	84.7170	88.4906	86.2893	85.7862	92.5157	94.8428	95.5975
	5	3	4	7	11	8	9	10	6	2	1
MUSK	93.5088	96.1053	95.0526	95.4737	95.2982	92.6316	86.9474	93.8596	96.8070	94.2807	99.2281
	9	3	6	4	5	10	11	8	2	7	1
Mean Rank	6.3889	4.2778	5.7778	5.6111	7.7778	7.2778	8.4444	8.7222	5.1667	4.9444	1.5556
Final Rank	7	2	6	5	9	8	10	11	4	3	1

Table 20. The mean size of feature subsets selecting by the comparison algorithm.

Datasets	PSO	DE	GWO	MVO	WOA	ABO	BOA	HHO	EO	GO	CODGBGO
IonosphereEW	10.2333	13.0000	4.9667	11.4667	3.8667	6.2333	6.3333	5.0000	3.1333	5.5667	4.5333
	9	11	4	10	2	7	8	5	1	6	3
Breastcancer	2.8333	3.1000	2.9000	3.2333	2.2667	2.9667	2.6667	2.8667	2.7333	2.3667	3.0000
	5	10	7	11	1	8	3	6	4	2	9
BreastEW	9.1000	12.5000	6.8000	11.7667	7.4667	8.3667	7.8333	9.0667	6.1667	9.5667	7.3667
	8	11	2	10	4	6	5	7	1	9	3
Congress	3.0000	3.3000	2.1000	3.7333	1.2000	2.3667	3.5000	2.1333	1.1333	1.6333	1.0000
	8	9	5	11	3	7	10	6	2	4	1
Wine	4.5333	5.3333	4.0000	3.8333	3.8000	3.8667	4.6333	4.4000	3.2333	4.0667	3.3667
	9	11	6	4	3	5	10	8	1	7	2
Vote	3.2333	4.3000	1.0000	3.3333	1.2333	1.0333	3.3333	3.2333	1.0000	3.9333	1.0667
	6	11	1	8	5	3	8	6	1	10	4
Vehicle	7.5667	8.3000	4.8000	7.2667	6.5667	5.7333	5.7667	7.6000	5.1333	7.0667	5.9333
	9	11	1	8	6	3	4	10	2	7	5
Exactly	5.6667	7.3667	3.4667	5.1333	2.2333	3.4333	1.6333	2.2000	5.2000	5.4667	5.9667
	9	11	5	6	3	4	1	2	7	8	10
Glass	4.2667	3.5667	3.3000	4.1000	2.3333	3.8333	3.8333	4.3000	4.0000	3.2333	3.0333
	2	6	6	2	7	3	3	1	1	1	1
HeartEW	3.8000	5.3000	2.8667	4.1000	5.0000	4.2667	3.6667	3.0667	3.6000	3.8000	4.0333
	6	1	9	3	1	1	3	4	3	2	1
Landsat	12.5333	13.5000	8.4333	12.8000	8.5000	10.4667	9.5000	10.2333	8.0333	11.3667	9.8000
	3	1	8	1	6	2	4	2	3	1	1
Lymphography	6.8333	8.0000	3.5667	6.3000	5.8333	6.6000	5.8333	4.6667	5.4333	6.3000	5.5333
	10	11	1	7	5	9	5	2	3	7	4
Zoo	5.8333	7.6000	5.6333	7.2000	7.8000	5.8667	6.7667	7.3333	5.7000	5.2667	6.1000
	4	10	2	8	11	5	7	9	3	1	6
WDBC	6.5667	8.2333	2.7333	7.4333	3.3333	2.0333	6.3333	5.5000	2.9333	3.5667	3.9000
	9	11	2	10	4	1	8	7	3	5	6
SonarEW	20.8333	29.8333	11.9333	22.8667	13.2000	17.8667	17.0333	16.7000	9.4667	15.8333	11.5000
	9	11	3	10	4	8	7	6	1	5	2
Libras	34.4000	51.3667	21.1667	38.3667	20.3667	23.0667	22.3333	21.0667	14.3667	28.5000	21.0000
	9	11	5	10	2	7	6	4	1	8	3
Spectf	18.8333	23.0000	12.8333	18.7000	5.7333	7.8000	11.0000	8.3667	11.2667	15.5333	15.5000
	10	11	6	9	1	2	4	3	5	8	7
MUSK	72.2667	104.7333	39.2000	72.3333	65.2000	49.2667	41.3667	56.7000	37.9333	66.0333	49.2000
	9	11	2	10	7	5	3	6	1	8	4
Mean Rank	7.4444	9.3889	4.1667	7.6667	4.1667	4.7778	5.5000	5.2222	2.3889	5.5000	4.0000
Final Rank	9	11	3	10	3	5	7	6	1	7	2

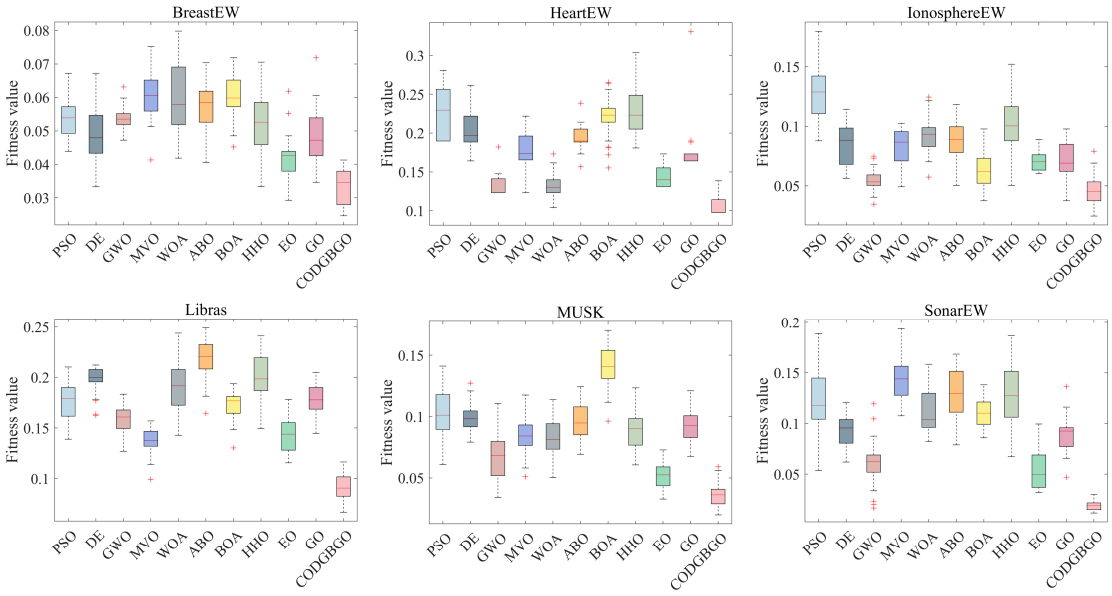

Figure 8. The boxplot of comparison algorithms on feature selection.

Finally, Figure 11 shows the combined performance on six evaluation criteria, which are best fitness value, average fitness value, worst fitness value, average accuracy, average running time, and average feature subset size. As can be seen in Figure 11, CODGBGO does not dominate in terms of average runtime, but it is within an acceptable range. CODGBGO is weaker than EO in average feature subset size, but CODGBGO outperforms EO in classification accuracy. CODGBGO outperforms the comparison algorithm in all other metrics. In summary, it can be shown that the comprehensive performance of CODGBGO is better than the comparison algorithm, and it is an effective method to solve the feature selection problem.

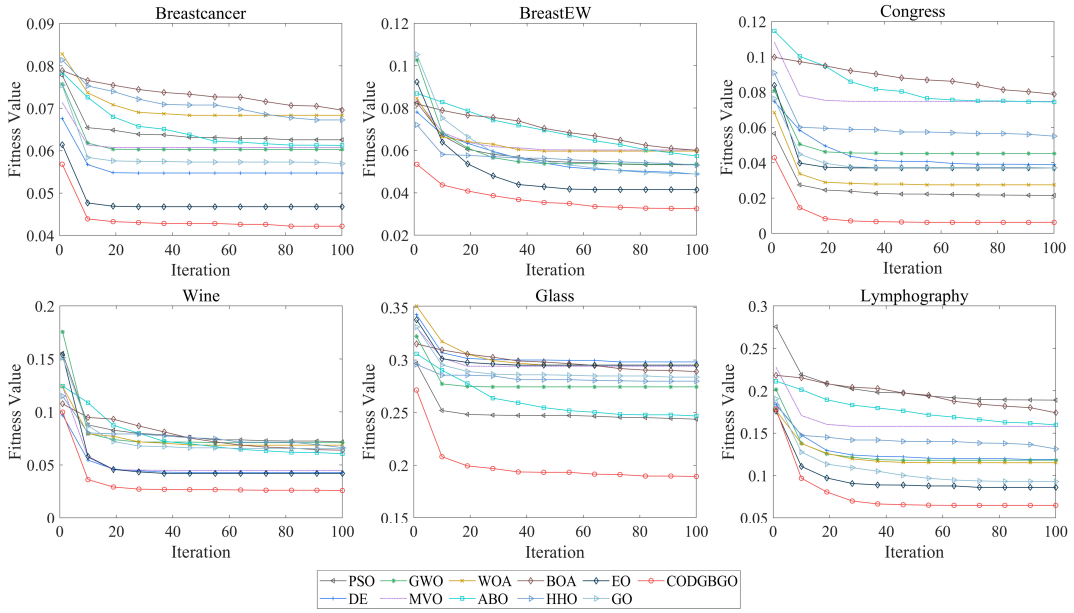

Figure 9. The convergence curve plots of comparison algorithms on feature selection.

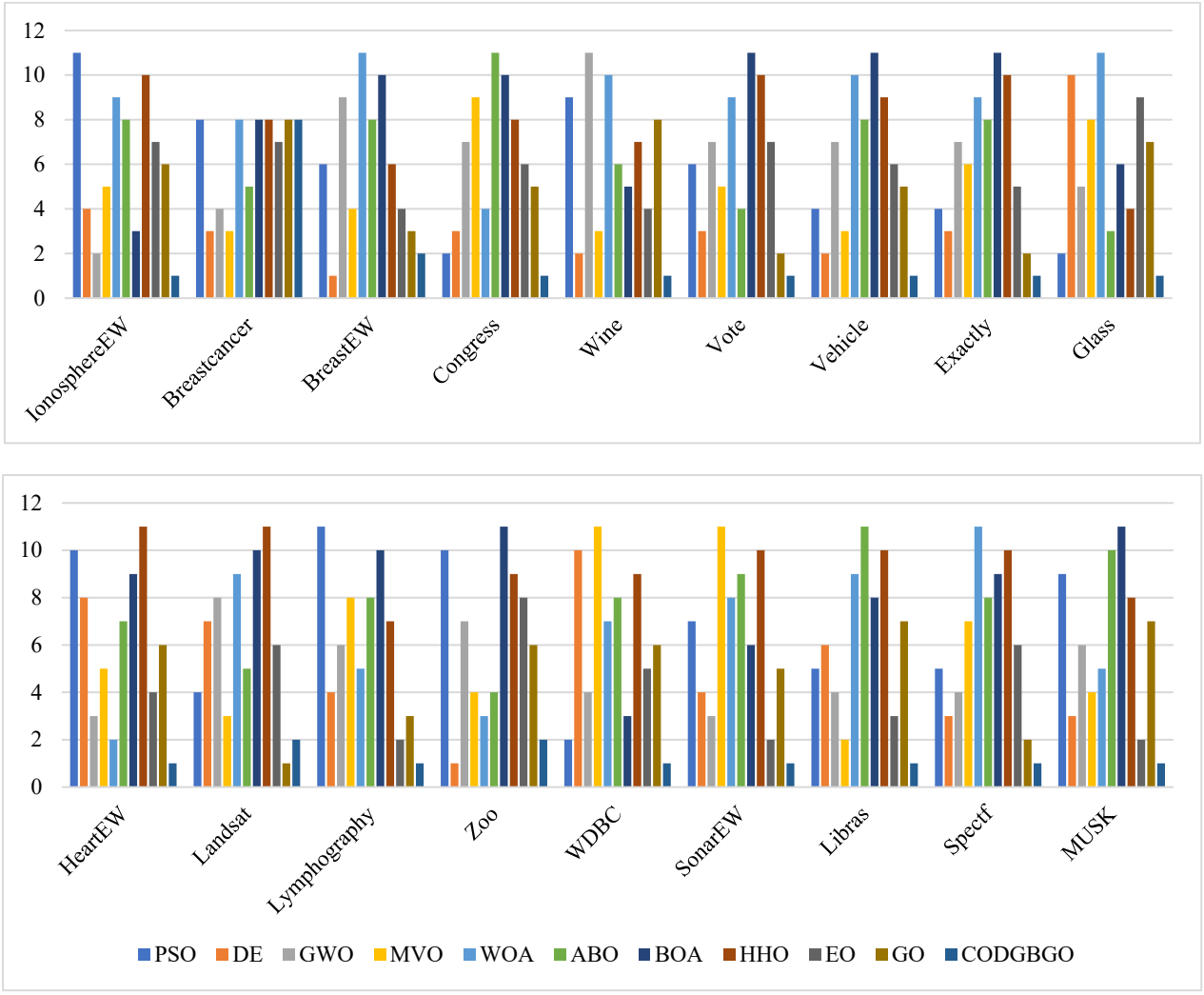

Figure 10. Comparison of average classification accuracy.

Table 21. Mean runtime of feature selection based on different algorithms.

Datasets	PSO	DE	GWO	MVO	WOA	ABO	BOA	HHO	EO	GO	CODGBGO
IonosphereEW	3.3108	3.1722	3.3264	3.1784	2.9800	2.1202	3.2604	4.9315	3.3270	6.5616	6.6642
Breastcancer	3.6709	3.5429	3.4315	3.4680	3.0807	2.2683	3.4019	5.7813	3.4697	6.8237	6.9120
BreastEW	3.3533	3.2018	3.5022	3.2243	3.1798	2.2524	3.3439	5.2125	3.4673	6.6059	6.9163
Congress	3.3494	3.2961	3.2631	3.3033	2.6290	2.0781	3.2823	4.5191	3.1994	6.5350	6.4676
Wine	3.1881	3.1710	3.1694	3.1464	2.9201	2.1440	3.1045	4.6576	3.1517	6.3121	6.3179
Vote	3.3267	3.3109	3.2309	3.2795	2.6670	2.0366	3.2974	4.6792	3.3187	6.5316	6.6233
Vehicle	3.6556	3.6239	3.5830	3.6427	3.3595	2.4563	3.5264	5.4494	3.5911	7.4301	7.4114
Exactly	3.7490	3.7573	3.6516	3.7003	3.0622	2.2741	3.5822	5.0219	3.6353	7.3639	7.6153
Glass	3.2112	3.1554	3.1348	3.1633	2.7608	2.1511	2.9145	4.8530	3.1383	6.2397	6.2764
HeartEW	3.2382	3.2001	3.2068	3.1889	2.9947	2.1126	3.0886	4.7290	3.2077	6.4120	6.5637
Landsat	4.3829	4.4849	4.6258	4.2721	4.2646	2.9833	4.4782	7.0426	4.6445	8.6532	9.0210
Lymphography	3.1431	3.1904	3.1270	3.1329	2.8913	2.1470	3.1233	4.5939	3.1329	6.2383	6.2533
Zoo	3.2060	3.0961	3.0762	3.0675	2.9377	2.0992	3.1017	4.8683	3.2445	6.1870	6.2997
WDBC	3.3966	3.2996	3.3730	3.3750	2.9942	2.0821	3.3059	5.0062	3.3910	6.8395	6.8839
SonarEW	3.0436	3.0531	3.0037	3.0109	2.9500	2.1247	3.0362	4.6112	3.0677	6.0004	6.1562
Libras	3.2948	3.3294	3.1570	3.2474	3.0417	2.2046	3.1727	4.8582	3.1632	6.3859	6.3510
Spectf	3.0568	3.0503	3.0780	3.0571	2.8979	2.0055	3.0874	4.7757	3.1155	5.9824	6.1543
MUSK	3.6771	3.8192	3.3378	3.5578	3.3299	2.3570	3.3974	5.3402	3.3292	7.1061	6.8958
Mean time	3.4030	3.3753	3.3488	3.3342	3.0523	2.2165	3.3058	5.0517	3.3664	6.6782	6.7657
Mean Rank	8	7	5	4	2	1	3	9	6	10	11

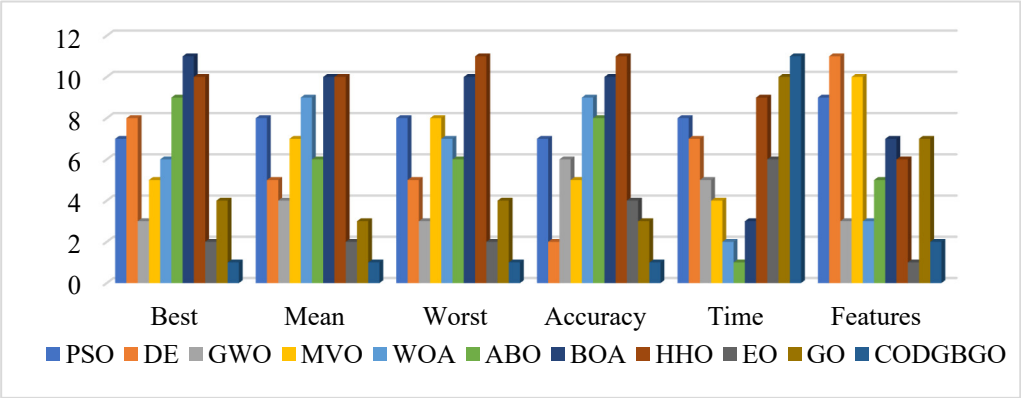

Figure 11. Combined performance on six evaluation criteria.

4.5. Results for Constrained Engineering Application Problems

In the previous experiments, the ability of CODGBGO to solve numerical optimization problems and discretized optimization problems was verified, and in this section, the proposed CODGBGO is mainly applied to solve four constrained engineering optimization problems in order to evaluate its performance in solving constrained engineering optimization problems. The selected engineering problems are 10-bar truss design, tension/compression spring design, weight minimization of a speed reducer, and welded beam design. The comparison algorithms are GO, INFO, SSA, TLBO, BESD, DE, PSO, SO, FOPSO, and ICGWO. The population size was set to 60, the maximum number of function evaluations was set to 60,000, and each experiment was run independently 30 times.

4.5.1. Results on 10-Bar Truss Design [61]

The primary objective of this problem is to minimize the weight of the truss structure while ensuring that certain frequency constraints are met. The structure is shown in Figure 12, and the mathematical formulation for addressing this problem can be described as follows:

$$\begin{aligned}
&\text{Minimize :}\\
&f(\overline{x}) = \sum_{i=1}^{10} L_i(x_i)\rho_i A_i\\
&\text{subject to :}\\
&g_1(\overline{x}) = \frac{7}{\omega_1(\overline{x})} - 1 \le 0,\\
&g_2(\overline{x}) = \frac{15}{\omega_2(\overline{x})} - 1 \le 0,\\
&g_3(\overline{x}) = \frac{20}{\omega_3(\overline{x})} - 1 \le 0,\\
&\text{with bounds :}\\
&6.45 \times 10^{-5} \le A_i \le 5 \times 10^{-3}, i = 1, 2, \ldots, 10.\\
&\text{where,}\\
&\overline{x} = \{A_1, A_2, \ldots, A_{10}\}, \rho = 2770.
\end{aligned} \tag{27}$$

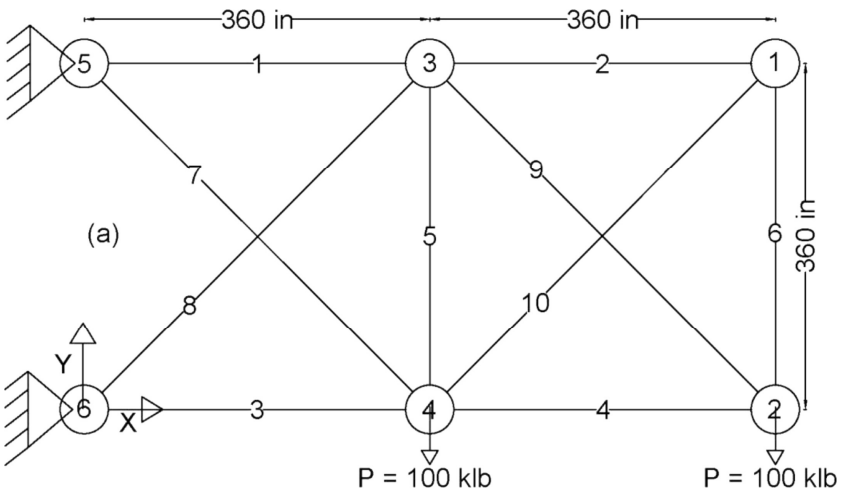

Figure 12. The 10-bar truss design structure.

The optimal solution of the proposed CODGBGO and comparative algorithms for the 10-bar truss design problem is shown in Table 22, and the statistical results are presented in Table 23. From Table 22, it can be observed that CODGBGO ranks first and obtains the optimal solution X = (0.003515, 0.001471, 0.003511, 0.001472, 0.000065, 0.000456, 0.002368, 0.002371, 0.001244, 0.001241) with a fitness function value of 524.4511. Furthermore, Table 23 demonstrates that CODGBGO outperforms other algorithms in terms of the best fitness function value, worst fitness function value, average fitness function value, and standard deviation, ranking first in all categories. In conclusion, CODGBGO is an effective method for solving the 10-bar truss design problem.

Table 22. The optimal solution of the proposed CODGBGO and comparative algorithms for the 10-bar truss design problem.

Algorithms	***X***										**Optimal**
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	
GO	0.003526	0.001473	0.003511	0.001466	0.000065	0.000456	0.002384	0.002362	0.001224	0.001250	524.4727
INFO	0.003560	0.001477	0.003443	0.001485	0.000065	0.000455	0.002361	0.002359	0.001228	0.001283	524.5944
SSA	0.003571	0.001596	0.003558	0.001489	0.000065	0.000447	0.002432	0.002200	0.001206	0.001235	525.0305
TLBO	0.003493	0.001466	0.003537	0.001454	0.000065	0.000457	0.002394	0.002368	0.001233	0.001243	524.4853
BESD	0.003514	0.001515	0.003449	0.001451	0.000115	0.000455	0.002411	0.002330	0.001239	0.001297	526.6039
DE	0.003536	0.001470	0.003509	0.001511	0.000065	0.000454	0.002315	0.002394	0.001233	0.001245	524.5510
PSO	0.003578	0.001473	0.003455	0.001521	0.000065	0.000453	0.002297	0.002400	0.001267	0.001224	524.5765
SO	0.003552	0.001476	0.003445	0.001433	0.000065	0.000458	0.002433	0.002334	0.001236	0.001266	524.5727
FOPSO	0.003599	0.001420	0.003452	0.001539	0.000065	0.000455	0.002234	0.002503	0.001292	0.001176	524.7809
ICGWO	0.003211	0.001594	0.003734	0.001469	0.000065	0.000495	0.002890	0.002145	0.001243	0.001277	538.3172
CODGBGO	0.003515	0.001471	0.003511	0.001472	0.000065	0.000456	0.002368	0.002371	0.001244	0.001241	524.4511

Table 23. Statistical results of the proposed CODGBGO and comparative algorithms for the 10-bar truss design problem.

Algorithms	**Best**	**Worst**	**Mean**	**Std**
GO	524.472692	530.626625	524.972147	1.205804
INFO	524.594363	536.501563	527.520947	3.520433
SSA	525.030465	536.997795	529.502118	3.881703
TLBO	524.485260	530.858513	525.462996	2.132384
BESD	526.603949	531.093229	529.159895	1.163149
DE	524.550988	525.206518	524.754413	0.154388
PSO	524.576532	533.737282	528.496080	3.450720
SO	524.572671	535.307127	527.344681	2.970005
FOPSO	524.780922	537.102061	529.847788	3.819223
ICGWO	541.278051	573.614056	552.856617	7.635963
CODGBGO	524.451125	524.602817	524.463803	0.034718

4.5.2. Results on Tension/Compression Spring Design [62]

The primary aim of this problem is to optimize the mass of a tension or compression spring. This problem involves four constraints, and three variables are employed to compute the mass: the wire diameter (x_1), the average coil diameter (x_2), and the number of active coils (x_3). The structure is shown in Figure 13, and the formulation of this problem is as follows:

$$\begin{array}{l}\text{Minimize :}\\ f(\overline{x}) = x_1^2 x_2(2+x_3)\\ \text{subject to :}\\ g_1(\overline{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \le 0\\ g_2(\overline{x}) = \frac{4x_2^2 - x_1 x_2}{12566(x_2 x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} - 1 \le 0\\ g_3(\overline{x}) = 1 - \frac{140.45x_1}{x_2^2 x_3} \le 0\\ g_4(\overline{x}) = \frac{x_1+x_2}{1.5} - 1 \le 0\\ \text{with bounds :}\\ 0.05 \le x_1 \le 2.00\\ 0.25 \le x_2 \le 1.30\\ 2.00 \le x_3 \le 15.0\end{array} \tag{28}$$

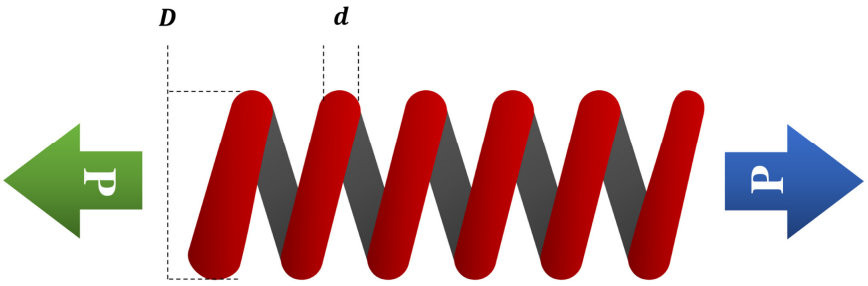

Figure 13. The tension/compression spring design structure.

The optimal solution of the proposed CODGBGO and comparative algorithms for the tension/compression spring design problem are presented in Table 24, while the corresponding statistical results are shown in Table 25. From Table 24, it can be observed that GO, DE, SO, and CODGBGO all rank first simultaneously. Specifically, CODGBGO achieves the best solution with a fitness function value of 0.012665, corresponding to X = (0.051701, 0.356996, 11.272677). Moreover, from Table 25, it is evident that CODGBGO ranks second in terms of the worst fitness function value, being weaker than DE, but ranks first in terms of the average fitness function value. Therefore, based on the above analysis, CODGBGO is considered suitable for solving the tension/compression spring design problem.

Table 24. The optimal solution of the proposed CODGBGO and comparative algorithms for the tension/compression spring design.

Algorithms		***X***		**Optimal**
	x_1	x_2	x_3	
GO	0.051610	0.354815	11.401536	0.012665
INFO	0.051879	0.361303	11.025125	0.012666
SSA	0.050000	0.315802	14.245195	0.012826
TLBO	0.051520	0.352659	11.531001	0.012666
BESD	0.051768	0.358525	11.191061	0.012674
DE	0.051696	0.356874	11.279811	0.012665
PSO	0.051861	0.360861	11.050136	0.012666
SO	0.051636	0.355443	11.364087	0.012665
FOPSO	0.051654	0.355874	11.338620	0.012665
ICGWO	0.050000	0.317212	14.230810	0.012872
CODGBGO	0.051701	0.356996	11.272677	0.012665

Table 25. Statistical results of the proposed CODGBGO and comparative algorithms for the tension/compression spring design.

Algorithms	Best	Worst	Mean	Std
GO	0.012665	0.012742	0.012672	1.423895×10^{-5}
INFO	0.012666	0.012734	0.012686	1.642346×10^{-5}
SSA	0.012728	0.013977	0.012920	2.356985×10^{-4}
TLBO	0.012666	0.012732	0.012683	1.585319×10^{-5}
BESD	0.012674	0.012792	0.012727	2.427097×10^{-5}
DE	0.012665	0.012666	0.012665	2.013640×10^{-7}
PSO	0.012666	0.014329	0.013268	5.261813×10^{-4}
SO	0.012665	0.013907	0.012835	2.417478×10^{-4}
FOPSO	0.012665	0.014304	0.013264	5.067277×10^{-4}
ICGWO	0.013111	0.016786	0.014086	1.306659×10^{-3}
CODGBGO	0.012665	0.012676	0.012666	1.957407×10^{-6}

4.5.3. Results on Weight Minimization of a Speed Reducer [63]

This task encompasses the development of a speed reduction mechanism for a compact aircraft engine. The structure is shown in Figure 14, and the optimization problem derived from this task can be expressed in the following manner:

$$
\begin{aligned}
&\text{Minimize :}\\
&f(\overline{x}) = 0.7854x_2^2x_1(14.9334x_3 - 43.0934 + 3.3333x_3^2) + 0.7854(x_5x_7^2 + x_4x_6^2)\\
&\quad -1.508x_1(x_7^2 + x_6^2) + 7.477(x_7^3 + x_6^3)\\
&\text{subject to :}\\
&g_1(\overline{x}) = -x_1x_2^2x_3 + 27 \le 0,\\
&g_2(\overline{x}) = -x_1x_2^2x_3^2 + 397.5 \le 0,\\
&g_3(\overline{x}) = -x_2x_6^4x_3x_4^{-3} + 1.93 \le 0,\\
&g_4(\overline{x}) = -x_2x_7^4x_3x_5^{-3} + 1.93 \le 0,\\
&g_5(\overline{x}) = 10x_6^{-3}\sqrt{16.91 \times 10^6 + (745x_4x_2^{-1}x_3^{-1})^2} - 1100 \le 0,\\
&g_6(\overline{x}) = 10x_7^{-3}\sqrt{157.5 \times 10^6 + (745x_5x_2^{-1}x_3^{-1})^2} - 850 \le 0,\\
&g_7(\overline{x}) = x_2x_3 - 40 \le 0,\\
&g_8(\overline{x}) = -x_1x_2^{-1} + 5 \le 0,\\
&g_9(\overline{x}) = x_1x_2^{-1} - 12 \le 0,\\
&g_{10}(\overline{x}) = 1.5x_6 - x_4 + 1.9 \le 0,\\
&g_{11}(\overline{x}) = 1.1x_7 - x_5 + 1.9 \le 0,\\
&\text{with bounds :}\\
&0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 2.6 \le x_1 \le 3.6, 5 \le x_7 \le 5.5, 7.3 \le x_5, x_4 \le 8.3, 2.9 \le x_6 \le 3.9.
\end{aligned}
\tag{29}
$$

The optimal solution of the proposed CODGBGO and comparative algorithms for the weight minimization of a speed reducer problem is shown in Table 26, and the statistical results are presented in Table 27. From Table 26, it can be observed that GO, INFO, TLBO, PSO, SO, and CODGBGO all ranked first simultaneously. CODGBGO obtained the best solution with a fitness function value of 2994.424466, where X = (3.500000, 0.700000, 17.000000, 7.300000, 7.715320, 3.350541, 5.286654). Additionally, Table 27 reveals that CODGBGO shares the top rank with GO, INFO, TLBO, and PSO regarding the worst and average fitness function values. Based on these findings, this study concludes that CODGBGO is well-suited for solving the weight minimization problem of a speed reducer.

Table 26. The optimal solution of the proposed CODGBGO and comparative algorithms for the weight minimization of a speed reducer.

Algorithms	***X***							**Optimal**
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	
GO	3.500000	0.700000	17.000000	7.300000	7.715320	3.350541	5.286654	2994.424466
INFO	3.500000	0.700000	17.000000	7.300000	7.715320	3.350541	5.286654	2994.424466
SSA	3.500072	0.700000	17.000000	7.660034	8.087044	3.358114	5.286784	3007.818687
TLBO	3.500000	0.700000	17.000000	7.300000	7.715320	3.350541	5.286654	2994.424466
BESD	3.503719	0.700002	17.001660	7.340884	7.755843	3.351224	5.288070	2998.504231
DE	3.500000	0.700000	17.000000	7.300000	7.715320	3.350541	5.286654	2994.424479
PSO	3.500000	0.700000	17.000000	7.300000	7.715320	3.350541	5.286654	2994.424466
SO	3.500000	0.700000	17.000000	7.300000	7.715320	3.350541	5.286654	2994.424466
FOPSO	3.500000	0.700000	17.000000	7.300000	7.715320	3.350541	5.286654	2994.424466
ICGWO	3.600000	0.700000	17.000000	8.300000	8.300000	3.353812	5.500000	3197.929838
CODGBGO	3.500000	0.700000	17.000000	7.300000	7.715320	3.350541	5.286654	2994.424466

Table 27. Statistical results of the proposed CODGBGO and comparative algorithms for the weight minimization of a speed reducer.

Algorithms	**Best**	**Worst**	**Mean**	**Std**
GO	2994.424466	2994.424466	2994.424466	1.850085×10^{-12}
INFO	2994.424466	2994.424466	2994.424466	1.850085×10^{-12}
SSA	3007.818687	3124.932806	3035.033900	2.585991×10^{1}
TLBO	2994.424466	2994.424466	2994.424466	1.850085×10^{-12}
BESD	2998.504231	3005.944928	3002.199777	1.751670×10^{0}
DE	2994.424479	2994.424576	2994.424525	3.035569×10^{-5}
PSO	2994.424466	2994.424466	2994.424466	1.850085×10^{-12}
SO	2994.424466	2996.185353	2994.499642	3.245885×10^{-1}
FOPSO	2994.424466	3033.701596	2995.733703	7.050461×10^{0}
ICGWO	3218.393654	3218.393654	3218.393654	4.547474×10^{-13}
CODGBGO	2994.424466	2994.424466	2994.424466	1.850085×10^{-12}

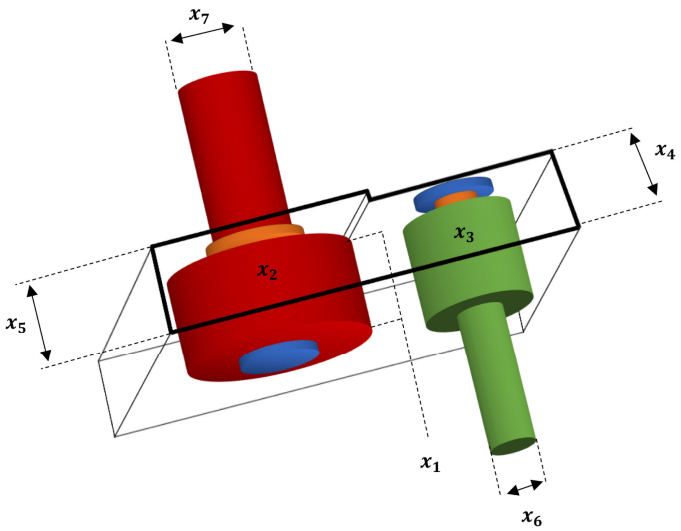

Figure 14. The speed reducer design structure.

4.5.4. Results on Welded Beam Design [64]

The primary aim of this problem is to optimize the design of a welded beam while minimizing costs. The problem involves five constraints, and four variables are utilized

in the development of the welded beam. The structure is shown in Figure 15, and the mathematical formulation of this problem can be outlined as follows:

$$
\begin{aligned}
&\text{Minimize :}\\
&\mathrm{f}(\overline{x}) = 0.04811x_3x_4(x_2+14) + 1.10471x_1^2x_2\\
&\text{subject to :}\\
&g_1(\overline{x}) = x_1 - x_4 \le 0,\\
&g_2(\overline{x}) = \delta(\overline{x}) - \delta_{max} \le 0,\\
&g_3(\overline{x}) = P \le P_c(\overline{x})\\
&g_4(\overline{x}) = \tau_{max} \ge \tau(\overline{x}),\\
&g_5(\overline{x}) = \sigma(\overline{x}) - \sigma_{max} \le 0,\\
&\text{where,}\\
&\tau = \sqrt{\tau'^2 + \tau''^2 + 2\tau'\tau''\frac{x_2}{2R}}, \tau'' = \frac{RM}{J}, \tau' = \frac{P}{\sqrt{2}x_2x_1},\\
&M = p\left(\frac{x_2}{2} + L\right),\\
&R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}, J = 2\left(\left(\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2\right)\sqrt{2}x_1x_2\right),\\
&\sigma(\overline{x}) = \frac{6PL}{x_4x_3^2},\\
&\delta(\overline{x}) = \frac{6PL^3}{Ex_3^2X_4},\\
&P_c(\overline{x}) = \frac{4.013Ex_3x_4^3}{6L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)\\
&L = 14\text{in}, P = 6000\text{lb}, E = 30.10^6\text{psi}, \sigma_{max} = 30,000\text{psi},\\
&\tau_{max} = 13,600\text{psi}, G = 12.10^6\text{psi}, \delta_{max} = 0.25\text{in},.
\end{aligned}
\tag{30}
$$

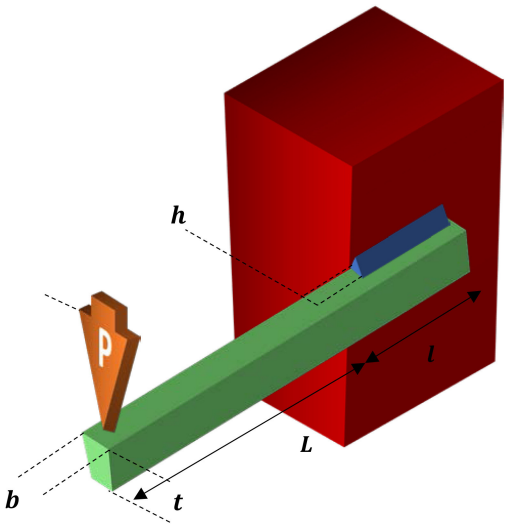

Figure 15. The welded beam design structure.

The optimal solution of the proposed CODGBGO algorithm and the comparative algorithm on the Welded beam design problem is shown in Table 28, and the statistical results are presented in Table 29. From Table 28, it can be observed that GO, INFO, TLBO, DE, PSO, SO, and CODGBGO are all ranked first simultaneously. CODGBGO achieves the best solution with a fitness function value of 1.670218, where X = (0.198832, 3.337365, 9.192024, 0.198832). Additionally, from Table 29, it can be seen that CODGBGO shares the top rank with GO, INFO, and TLBO in terms of the worst fitness function value. Furthermore, CODGBGO is jointly ranked first with GO, INFO, TLBO, and DE in terms of the average fitness function value. In conclusion, based on the above analysis, this study concludes that CODGBGO is suitable for solving the welded beam design problem.

Table 28. The optimal solution of the proposed CODGBGO and comparative algorithms for the welded beam design.

Algorithms	X				Optimal
	x_1	x_2	x_3	x_4	
GO	0.198832	3.337365	9.192024	0.198832	1.670218
INFO	0.198832	3.337365	9.192024	0.198832	1.670218
SSA	0.196408	3.378769	9.210326	0.198747	1.674474
TLBO	0.198832	3.337365	9.192024	0.198832	1.670218
BESD	0.198250	3.368697	9.188929	0.199399	1.677316
DE	0.198832	3.337367	9.192024	0.198832	1.670218
PSO	0.198832	3.337365	9.192024	0.198832	1.670218
SO	0.198832	3.337365	9.192024	0.198832	1.670218
FOPSO	0.198832	3.337365	9.192024	0.198832	1.670218
ICGWO	0.198308	3.457363	9.478214	0.201282	1.752510
CODGBGO	0.198832	3.337365	9.192024	0.198832	1.670218

Table 29. Statistical results of the proposed CODGBGO and comparative algorithms for the welded beam design.

Algorithms	Best	Worst	Mean	Std
GO	1.670218	1.670218	1.670218	1.086776×10^{-8}
INFO	1.670218	1.670218	1.670218	1.129203×10^{-15}
SSA	1.674474	1.831619	1.749027	5.575776×10^{-2}
TLBO	1.670218	1.670218	1.670218	1.129203×10^{-15}
BESD	1.677316	1.691348	1.683443	3.948120×10^{-3}
DE	1.670218	1.670219	1.670218	2.127735×10^{-7}
PSO	1.670218	1.711166	1.672234	8.154641×10^{-3}
SO	1.670218	1.679417	1.670570	1.673049×10^{-3}
FOPSO	1.670218	2.330363	1.860656	2.038525×10^{-1}
ICGWO	1.752510	2.140073	1.888851	7.733192×10^{-2}
CODGBGO	1.670218	1.670218	1.670218	9.433468×10^{-12}

5. Conclusions and Future Directions

In this work, the Circle-OBL initialization strategy, exploration strategy, and exploitation strategy are integrated into the GO to form an enhanced GO, which is called CODGBGO. In CODGBGO, the first step is to provide a good initial population using the Circle-OBL initialization strategy, which is instrumental in enhancing the search quality. Secondly, the exploration strategy is adopted to enhance population diversity and the ability of the algorithm to escape local optima traps. Finally, the exploitation strategy is used to improve the exploitation ability of the algorithm and enhance its convergence performance. Through a large number of experiments, it has been confirmed that CODGBGO has good advantages and scalability in dealing with continuous optimization, discrete optimization, and engineering optimization, which is specifically reflected in the comprehensive evaluation due to the good balance between the exploitation phase and the exploration phase in CODGBGO, which makes the algorithm have a better performance in global optimization search. However, by analyzing the unimodal problems, we find that the exploitation strategy proposed in this paper needs to be further improved to enhance the exploitation performance. Meanwhile, when dealing with discrete combinatorial problems with high dimensionality, the CODGBGO proposed in this paper has weaker performance than the traditional optimization algorithms in specific cases, despite its superiority in comprehensive performance. Finally, although it shows good results on engineering optimization problems, it needs to be further tested in prospective applications, such as the field of nonlinear systems.

Therefore, our next work will focus on the following aspects: 1. further improving the CODGBGO proposed in this paper to further enhance the solution's performance. 2. developing specific algorithms for high-dimensional combinatorial optimization problems that are applicable to such problems. 3. incorporating complex optimization problems in nonlinear systems into the testing of the algorithms in order to evaluate their performance more comprehensively.

Author Contributions: Conceptualization, R.X. and L.Y.; methodology, R.X. and L.Y.; software, R.X. and S.L.; validation, R.X. and F.W.; formal analysis, F.W.; investigation, T.Z.; resources, S.L.; data curation, R.X. and L.Y.; writing—original draft preparation, R.X.; writing—review and editing, R.X. and P.Y.; visualization, F.W., T.Z. and P.Y.; supervision, L.Y. and S.L.; project administration, L.Y. and S.L.; funding acquisition, L.Y. and S.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded in part by the National Key Research and Development Program of China, grant number 2023YFB3308802; and in part by the National Natural Science Foundation of China's Top-Level Program, grant number 52275480; in part by the Reserve Projects for Centralized Guidance of Local Science and Technology Development Funds, grant number QKHZYD [2023]002.

Data Availability Statement: All data in this paper can be obtained by contacting the corresponding author.

Acknowledgments: We are grateful to the publisher for formatting the paper and to the editor and reviewers for their valuable suggestions.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Hussien, A.G.; Oliva, D.; Houssein, E.H.; Juan, A.A.; Yu, X. Binary Whale Optimization Algorithm for Dimensionality Reduction. *Mathematics* **2020**, *8*, 1821. [CrossRef]
2. Hao, Y.; Helo, P.; Shamsuzzoha, A. Virtual Factory System Design and Implementation: Integrated Sustainable Manufacturing. *Int. J. Syst. Sci. Oper. Logist.* **2018**, *5*, 116–132. [CrossRef]
3. Simpson, A.R.; Dandy, G.C.; Murphy, L.J. Genetic algorithms compared to other techniques for pipe optimization. *J. Water Resour. Plan. Manag.* **1994**, *120*, 423–443. [CrossRef]
4. Gharaei, A.; Hoseini Shekarabi, S.A.; Karimi, M. Modelling And Optimal Lot-Sizing of the Replenishments in Constrained, Multi-Product and Bi-Objective EPQ Models with Defective Products: Generalised Cross Decomposition. *Int. J. Syst. Sci. Oper. Logist.* **2020**, *7*, 262–274. [CrossRef]
5. Hussien, A.G.; Hassanien, A.E.; Houssein, E.H.; Amin, M.; Azar, A.T. New Binary Whale Optimization Algorithm for Discrete Optimization Problems. *Eng. Optim.* **2020**, *52*, 945–959. [CrossRef]
6. Sayyadi, R.; Awasthi, A. An Integrated Approach Based on System Dynamics and ANP for Evaluating Sustainable Transportation Policies. *Int. J. Syst. Sci. Oper. Logist.* **2020**, *7*, 182–191. [CrossRef]
7. Schwefel, H.-P.; Beyer, H.-G. Evolution Strategies-A Comprehensive Introduction Evolution Strategies A Comprehensive Introduction. *ACM Comput. Classif.* **2002**, *1*, 3–52.
8. Simon, D. Biogeography-Based Optimization. *IEEE Trans. Evol. Comput.* **2008**, *12*, 702–713. [CrossRef]
9. Fleming, P.J.; Fonseca, C.M. Genetic algorithms in control systems engineering. *IFAC Proc. Vol.* **1993**, *26*, 605–612. [CrossRef]
10. Azizi, M.; Talatahari, S.; Gandomi, A.H. Fire Hawk Optimizer: A Novel Metaheuristic Algorithm. *Artif. Intell. Rev.* **2023**, *56*, 287–363. [CrossRef]
11. Mirjalili, S.; Lewis, A. The Whale Optimization Algorithm. *Adv. Eng. Softw.* **2016**, *95*, 51–67. [CrossRef]
12. Mohammed, H.; Rashid, T. FOX: A FOX-Inspired Optimization Algorithm. *Appl. Intell.* **2023**, *53*, 1030–1050. [CrossRef]
13. Chopra, N.; Mohsin Ansari, M. Golden Jackal Optimization: A Novel Nature-Inspired Optimizer for Engineering Applications. *Expert Syst. Appl.* **2022**, *198*, 116924. [CrossRef]
14. Rashedi, E.; Nezamabadi-pour, H.; Saryazdi, S. GSA: A Gravitational Search Algorithm. *Inf. Sci.* **2009**, *179*, 2232–2248. [CrossRef]
15. Erol, O.K.; Eksin, I. A New Optimization Method: Big Bang-Big Crunch. *Adv. Eng. Softw.* **2006**, *37*, 106–111. [CrossRef]
16. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by Simulated Annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
17. Kushwaha, N.; Pant, M.; Kant, S.; Jain, V.K. Magnetic Optimization Algorithm for Data Clustering. *Pattern Recognit. Lett.* **2018**, *115*, 59–65. [CrossRef]
18. Kaveh, A.; Bakhshpoori, T. Water Evaporation Optimization: A Novel Physically Inspired Optimization Algorithm. *Comput. Struct.* **2016**, *167*, 69–85. [CrossRef]
19. Zhao, W.; Wang, L.; Zhang, Z. Atom Search Optimization and Its Application to Solve a Hydrogeologic Parameter Estimation Problem. *Knowl. Based Syst.* **2019**, *163*, 283–304. [CrossRef]

20. Shabani, A.; Asgarian, B.; Salido, M.; Asil Gharebaghi, S. Search and Rescue Optimization Algorithm: A New Optimization Method for Solving Constrained Engineering Optimization Problems. *Expert Syst. Appl.* **2020**, *161*, 113698. [CrossRef]
21. Mousavirad, S.J.; Ebrahimpour-Komleh, H. Human Mental Search: A New Population-Based Metaheuristic Optimization Algorithm. *Appl. Intell.* **2017**, *47*, 850–887. [CrossRef]
22. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The Arithmetic Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
23. Shaheen, A.M.; Elsayed, A.M.; El-Sehiemy, R.A.; Abdelaziz, A.Y. Equilibrium Optimization Algorithm for Network Reconfiguration and Distributed Generation Allocation in Power Systems. *Appl. Soft Comput.* **2021**, *98*, 106867. [CrossRef]
24. Turkoglu, B.; Uymaz, S.A.; Kaya, E. Binary Artificial Algae Algorithm for Feature Selection [Formula Presented]. *Appl. Soft Comput.* **2022**, *120*, 108630. [CrossRef]
25. Hu, G.; Du, B.; Wang, X.; Wei, G. An Enhanced Black Widow Optimization Algorithm for Feature Selection. *Knowl. Based Syst.* **2022**, *235*, 107638. [CrossRef]
26. Xu, M.; Song, Q.; Xi, M.; Zhou, Z. Binary Arithmetic Optimization Algorithm for Feature Selection. *Soft Comput.* **2023**, *27*, 11395–11429. [CrossRef]
27. Hu, G.; Zhong, J.; Du, B.; Wei, G. An Enhanced Hybrid Arithmetic Optimization Algorithm for Engineering Applications. *Comput. Methods Appl. Mech. Eng.* **2022**, *394*, 114901. [CrossRef]
28. Zhang, Q.; Gao, H.; Zhan, Z.H.; Li, J.; Zhang, H. Growth Optimizer: A Powerful Metaheuristic Algorithm for Solving Con-tinuous and Discrete Global Optimization Problems. *Knowl. Based Syst.* **2023**, *261*, 110206. [CrossRef]
29. Aribia, H.B.; El-Rifaie, A.M.; Tolba, M.A.; Shaheen, A.; Moustafa, G.; Elsayed, F.; Elshahed, M. Growth Optimizer for Parameter Identification of Solar Photovoltaic Cells and Modules. *Sustainability* **2023**, *15*, 7896. [CrossRef]
30. Hakmi, S.H.; Alnami, H.; Moustafa, G.; Ginidi, A.R.; Shaheen, A.M. Modified Rime-Ice Growth Optimizer with Polynomial Differential Learning Operator for Single- and Double-Diode PV Parameter Estimation Problem. *Electronics* **2024**, *13*, 1611. [CrossRef]
31. Gao, H.; Zhang, Q.; Bu, X.; Zhang, H. Quadruple Parameter Adaptation Growth Optimizer with Integrated Distribution, Confrontation, and Balance Features for Optimization. *Expert Syst. Appl.* **2024**, *235*, 121218. [CrossRef]
32. Fatani, A.; Dahou, A.; Abd Elaziz, M.; Al-qaness, M.A.A.; Lu, S.; Alfadhli, S.A.; Alresheedi, S.S. Enhancing Intrusion Detection Systems for IoT and Cloud Environments Using a Growth Optimizer Algorithm and Conventional Neural Networks. *Sensors* **2023**, *23*, 4430. [CrossRef]
33. Altay, O. Chaotic Slime Mould Optimization Algorithm for Global Optimization. *Artif. Intell. Rev.* **2022**, *55*, 3979–4040. [CrossRef]
34. Qaraad, M.; Amjad, S.; Hussein, N.K.; Elhosseini, M.A. Large Scale Salp-Based Grey Wolf Optimization for Feature Selection and Global Optimization. *Neural Comput. Appl.* **2022**, *34*, 8989–9014. [CrossRef]
35. Ahmad, M.F.; Isa, N.A.M.; Lim, W.H.; Ang, K.M. Differential Evolution with Modified Initialization Scheme Using Chaotic Oppositional Based Learning Strategy. *Alex. Eng. J.* **2022**, *61*, 11835–11858. [CrossRef]
36. Li, X.D.; Wang, J.S.; Hao, W.K.; Zhang, M.; Wang, M. Chaotic Arithmetic Optimization Algorithm. *Appl. Intell.* **2022**, *52*, 16718–16757. [CrossRef]
37. Zhang, H.; Liu, T.; Ye, X.; Heidari, A.A.; Liang, G.; Chen, H.; Pan, Z. Differential Evolution-Assisted Salp Swarm Algorithm with Chaotic Structure for Real-World Problems. *Eng. Comput.* **2023**, *39*, 1735–1769. [CrossRef]
38. Dehghani, M.; Hubalovsky, S.; Trojovsky, P. Northern Goshawk Optimization: A New Swarm-Based Algorithm for Solving Optimization Problems. *IEEE Access.* **2021**, *9*, 162059–162080. [CrossRef]
39. Yang, Q.; Yan, J.Q.; Gao, X.D.; Xu, D.D.; Lu, Z.Y.; Zhang, J. Random Neighbor Elite Guided Differential Evolution for Global Numerical Optimization. *Inf. Sci.* **2022**, *607*, 1408–1438. [CrossRef]
40. Kennedy, J.; Eberhart, R. Particle Swarm Optimization. In Proceedings of the ICNN'95-International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995.
41. Storn, R.; Price, K. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *J. Glob. Optim.* **1997**, *11*, 341–359. [CrossRef]
42. Liang, J.J.; Qin, A.K.; Suganthan, P.N.; Baskar, S. Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions. *IEEE Trans. Evol. Comput.* **2006**, *10*, 281–295. [CrossRef]
43. Civicioglu, P.; Besdok, E. Bezier Search Differential Evolution Algorithm for Numerical Function Optimization: A Comparative Study with CRMLSP, MVO, WA, SHADE and LSHADE. *Expert Syst. Appl.* **2021**, *165*, 113875. [CrossRef]
44. Civicioglu, P.; Besdok, E. Bernstein-Levy Differential Evolution Algorithm for Numerical Function Optimization. *Neural Comput. Appl.* **2023**, *35*, 6603–6621. [CrossRef]
45. Malik, N.A.; Chang, C.L.; Chaudhary, N.I.; Raja, M.A.Z.; Cheema, K.M.; Shu, C.M.; Alshamrani, S.S. Knacks of Fractional Order Swarming Intelligence for Parameter Estimation of Harmonics in Electrical Systems. *Mathematics* **2022**, *10*, 1570. [CrossRef]
46. Mehmood, K.; Chaudhary, N.I.; Khan, Z.A.; Cheema, K.M.; Raja, M.A.Z. Variants of Chaotic Grey Wolf Heuristic for Robust Identification of Control Autoregressive Model. *Biomimetics* **2023**, *8*, 141. [CrossRef] [PubMed]
47. Rao, R.V.; Savsani, V.J.; Vakharia, D.P. Teaching-Learning-Based Optimization: A Novel Method for Constrained Mechanical Design Optimization Problems. *CAD Comput. Aided Des.* **2011**, *43*, 303–315. [CrossRef]
48. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]

49. Mirjalili, S.; Mirjalili, S.M.; Hatamlou, A. Multi-Verse Optimizer: A Nature-Inspired Algorithm for Global Optimization. *Neural Comput. Appl.* **2016**, *27*, 495–513. [CrossRef]
50. Mirjalili, S.; Gandomi, A.H.; Mirjalili, S.Z.; Saremi, S.; Faris, H.; Mirjalili, S.M. Salp Swarm Algorithm: A Bio-Inspired Optimizer for Engineering Design Problems. *Adv. Eng. Softw.* **2017**, *114*, 163–191. [CrossRef]
51. Heidari, A.A.; Mirjalili, S.; Faris, H.; Aljarah, I.; Mafarja, M.; Chen, H. Harris Hawks Optimization: Algorithm and Applications. *Future Gener. Comput. Syst.* **2019**, *97*, 849–872. [CrossRef]
52. Qi, X.; Zhu, Y.; Zhang, H. A New Meta-Heuristic Butterfly-Inspired Algorithm. *J. Comput. Sci.* **2017**, *23*, 226–239. [CrossRef]
53. Arora, S.; Singh, S. Butterfly Optimization Algorithm: A Novel Approach for Global Optimization. *Soft Comput.* **2019**, *23*, 715–734. [CrossRef]
54. Faramarzi, A.; Heidarinejad, M.; Stephens, B.; Mirjalili, S. Equilibrium optimizer: A novel optimization algorithm. *Knowl. Based Syst.* **2020**, *191*, 105190. [CrossRef]
55. Xue, J.; Shen, B. Dung Beetle Optimizer: A New Meta-Heuristic Algorithm for Global Optimization. *J. Supercomput.* **2023**, *79*, 7305–7336. [CrossRef]
56. Ahmadianfar, I.; Heidari, A.A.; Noshadian, S.; Chen, H.; Gandomi, A.H. INFO: An Efficient Optimization Algorithm Based on Weighted Mean of Vectors. *Expert Syst. Appl.* **2022**, *195*, 116516. [CrossRef]
57. Hashim, F.A.; Hussien, A.G. Snake Optimizer: A Novel Meta-Heuristic Optimization Algorithm. *Knowl. Based Syst.* **2022**, *242*, 108320. [CrossRef]
58. IEEE Computational Intelligence Society; Institute of Electrical and Electronics Engineers. Behavioral Study of the Surrogate Model-aware Evolutionary Search Framework. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation, Beijing, China, 6–11 July 2014; ISBN 9781479914883.
59. Institute of Electrical and Electronics Engineers; IEEE Computational Intelligence Society. Hybrid Single and Multiobjective Optimization for Engineering Design without Exact Specifications. In Proceedings of the 2020 IEEE Congress on Evolutionary Computation (CEC): 2020 Conference Proceedings, Glasgow, UK, 19–24 July 2020; ISBN 9781728169293.
60. Li, Y.; Han, T.; Zhou, H.; Tang, S.; Zhao, H. A Novel Adaptive L-SHADE Algorithm and Its Application in UAV Swarm Resource Configuration Problem. *Inf. Sci.* **2022**, *606*, 350–367. [CrossRef]
61. Azizi, M.; Aickelin, U.; Khorshidi, H.A.; Shishehgarkhaneh, M.B. Shape and Size Optimization of Truss Structures by Chaos Game Optimization Considering Frequency Constraints. *J. Adv. Res.* **2022**, *41*, 89–100. [CrossRef] [PubMed]
62. Tzanetos, A.; Blondin, M. A Qualitative Systematic Review of Metaheuristics Applied to Tension/Compression Spring Design Problem: Current Situation, Recommendations, and Research Direction. *Eng. Appl. Artif. Intell.* **2023**, *118*, 105521. [CrossRef]
63. Kumar, A.; Wu, G.; Ali, M.Z.; Mallipeddi, R.; Suganthan, P.N.; Das, S. A Test-Suite of Non-Convex Constrained Optimization Problems from the Real-World and Some Baseline Results. *Swarm Evol. Comput.* **2020**, *56*, 100693. [CrossRef]
64. Akay, B.; Karaboga, D. Artificial Bee Colony Algorithm for Large-Scale Problems and Engineering Design Optimization. *J. Intell. Manuf.* **2012**, *23*, 1001–1014. [CrossRef]