Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6655614, 24 pages
https://doi.org/10.1155/2021/6655614

Research Article

Improved Biogeography-Based Optimization Algorithm by Hierarchical Tissue-Like P System with Triggering Ablation Rules

Xiao Sang,[1] **Xiyu Liu**,[1] **Zhe Zhang**,[1] **and Lin Wang**[2]

[1]*College of Business, Shandong Normal University, Jinan 250300, China*
[2]*School of Management Engineering, Shandong Jianzhu University, Jinan 250100, China*

Correspondence should be addressed to Xiyu Liu; xyliu@sdnu.edu.cn

Received 3 December 2020; Revised 18 February 2021; Accepted 11 March 2021; Published 24 March 2021

Academic Editor: Diego Oliva

BBO is one of the new metaheuristic optimization algorithms, which is based on the science of biogeography. It can be used to solve optimization problems through the migration and drift of species between habitats. Many improved BBO algorithms have been proposed, but there were still many shortcomings in global optimization, convergence speed, and algorithm complexity. In response to the above problems, this paper proposes an improved BBO algorithm (DCGBBO) by hierarchical tissue-like P system with triggering ablation rules. Membrane computing is a branch of natural computing that aims to abstract computational models (P system) from the structure and function of biological cells and from the collaboration of cell groups such as organs and tissues. In this paper, firstly, a dynamic crossover migration operator is generated to improve the global search ability and also increase the species diversity. Secondly, a dynamic Gaussian mutation operator is introduced to speed up convergence and improve local search capabilities. To guarantee the correctness and feasibility of the mutation, a unified maximum mutation rate is designed. Finally, a hierarchical tissue-like P system with triggering ablation rules is combined with the DCGBBO algorithm, making use of evolution rules and communication rules to achieve migration and mutation of solutions and reduce computational complexity. In the experiments, eight classic benchmark functions and CEC 2017 benchmark functions are applied to demonstrate the effect of our algorithm. We apply the proposed algorithm to segment four colour pictures, and the results proved to be better compared to other algorithms.

1. Introduction

Membrane computing is a new field of natural computing that aims to abstract computing models from the function and structure of living cells and from the coordination of tissues and organs [1]. On the basis of DNA research, because of the inspiration of the protective role played by the cell membrane when the substance in the cell reacted, the calculation model of membrane computing was proposed and called it membrane system or P system [2]. Membrane calculation models are grouped into three main types: (1) cell-like P system [2], (2) tissue-like P system [3], (3) spiking neural P system [4]. Zhang et al. [5] have summarized the research on the efficiency and computing power of the three computing models. The tissue-like P system is applied as the basis, and a hierarchical tissue-like P system with starting ablation rules is proposed to assist the variants of intelligent optimization algorithm BBO to seek out the optimal solution. Since the organizational P system was proposed, countless extended systems have been introduced, and the universality of calculation has been proved, and some are combined with practical problems. There are multiple channel states with tissue P systems [6], and new evolutional symport/antiport rules were applied in tissue-like P systems [7], tissue P systems which carried cooperating rules [8], a unidirectional organization P system with promoters [9], timed steady-state tissue-like P systems which introduced evolutional symport/antiport rules [10], and image segmentation with complex chain P system based on evolutionary mechanism [11]. In the system, objects realize calculations through evolutionary rules and communication rules, where the implementation of the rules is uncertain,

and the number of rules can be maximized simultaneously, and thereby, the computational efficiency is enhanced stronger.

Biogeography-based optimization (BBO) [12] is one of the intelligent optimization algorithms, such as other optimization algorithms (particle swarm optimization (PSO) [13], artificial bee colony (ABC) [14], ant colony optimization (ACO) [15], differential evolution (DE) [16,17], Grey Wolf Optimizer (GWO) [18], monarch butterfly optimization (MBO) [19], earthworm optimization algorithm (EWA) [20], elephant herding optimization (EHO) [21], moth search (MS) algorithm [22], slime mould algorithm (SMA) [23], and Harris hawks optimization (HHO) [24]); a fixed calculation mode is constructed to solve diverse optimization problems. Gaining-sharing knowledge optimization algorithm (GSK) [25] was proposed, which is based on the concept of acquiring and sharing knowledge in the human life cycle. And there have been many research studies on the combination of metaheuristic algorithms and local search. The mutation operator in differential evolution is introduced into PFA (HPFA) [26]. A new classification for the source of inspiration for nature-inspired algorithms was designed too, and it can be classified into four groups: evolutionary techniques, swarm intelligence techniques, physics-based techniques, and human-related techniques. The BBO algorithm is one of the swarm intelligence techniques, which is essentially due to changes in external environmental factors, through the evolution of species within the habitat and the migration and communication between habitats, to achieve the enrichment of species diversity and the exploration and exploitation of species, so as to realize the evolution of habitat. Because the calculation mode of standard BBO is simple and single, its optimization efficiency is limited. Thence, since BBO was proposed, many scholars have made improvements to the shortcomings of the algorithm. For example, a new migration operator called blended migration was come up with which the feature of immigration is obtained by another solution [27]. An ecogeography-based optimization (EBO) was proposed [28], in which topological structure is introduced and habitat population was combined with it to form ecosystems. BBO-M integrates differential evolution (DE) algorithm and chaos theory to increase the search ability of mutation operators [29]. Worst opposition learning and random-scaled differential mutation BBO (WRBBO) was proposed to obtain global and local search ability [30]. The replacement of mutation operators greatly reduces the complexity of the algorithm. The fireworks algorithm (FA) and the BBO algorithm were crossed; that is, the advantages of the two algorithms are integrated to obtain a higher quality solution [31]. A two-stage biogeography-based optimization which carried differential (TDBBO) was constructed to solve the problem of obtaining a local optimal solution due to premature convergence and reduce the rotation difference [32]. DE/BBO was produced which combined the exploration of DE with the development of BBO effectively, so as to solve numerical optimization problems from a global perspective [33]. For the purpose of strengthening the optimization performance and reducing the complexity of the calculation process, a fused and efficient biogeography-based optimization (EMBBO) algorithm was programmed which utilizes a new example earning approach [34]. In this paper, a novel DCGBBO algorithm is designed, the convergence speed of the algorithm and optimization effect are improved on account of dynamic crossover migration operator and dynamic Gaussian mutation operator, and computational complexity is reduced through the introduction of tissue-like P system with ablation rules. The contributions are expressed as follows:

(1) Dynamic crossover migration operator is come up with to strengthen the global search ability and increase the species diversity. A dynamic Gaussian algorithm is produced, and the algorithm is planned into two steps, not only speed up the convergence speed but also increase the local search ability greatly meanwhile. By way of resolving situations that fall into local optimum, the opposition-based learning mechanism is brought in. The above operation improves the exploitation ability and the exploration ability of the algorithm.

(2) At the same time, a hierarchical tissue-like P system with triggering ablation rules designed for DCGBBO is introduced to reduce the complexity of the calculation process by the execution of rules in the system.

(3) DCGBBO is tested on 8 classic benchmark functions and CEC 2017 benchmark functions to terrify the optimization efficiency, and the Wilcoxon signed-rank test is employed to verify the optimization performance of DCGBBO; the results prove that DCGBBO is more efficient than many state-of-the-art BBO variants and other algorithms.

(4) At the meanwhile, DCGBBO is applied to segment colour image; the segmentation results reveal that DCGBBO is much better than other competitive algorithms.

The frame structure of the paper is as follows. Section 2 provides the related work and background knowledge which contained basic BBO algorithm and tissue-like P system. The proposed algorithm DCGBBO and hierarchical tissue-like P system with triggering ablation rules designed for DCGBBO are introduced in Section 3. And the experiment and the analysis of the algorithm are described in Section 4. The last Section 5 provides the summary and prospect of this paper.

2. Related Work: BBO Algorithm and Tissue-Like P System

2.1. BBO Algorithm. Biogeography-based optimization (BBO) was proposed in 2008 by Simon [12]. According to the definition of the BBO algorithm, the habitat is used to represent each individual in the population, that is, the candidate solution in the optimization problem. The combination of many habitats (N candidate solutions) constitutes a population. The fitness index variable (SIV) that measures the habitability of each habitat corresponds to the

feature vector (D-dimension) of each candidate solution in the optimization problem. Habitat fitness value (HSI) is used to measure the living environment of each habitat and the fitness function value of each candidate solution in the optimization problem ($f(x)$). SIVs can be regarded as controllable independent variables and HSI as dependent variables in algorithms.

In the BBO algorithm, if a habitat is suitable for survival, then more species would be contained in there. To renew each habitat, the information between habitats and other habitats would be exchanged through the migration of species. And the number of species in each habitat changes accordingly by calculating the immigration rate, the emigration rate, and mutation rate which are calculated by equations (1)–(3), respectively, as follows:

$$\lambda_i = I\left(1 - \frac{S_i}{S_{\max}}\right), \tag{1}$$

$$\mu_i = E\left(\frac{S_i}{S_{\max}}\right), \tag{2}$$

$$m_i = m_{\max}\left(1 - \frac{P_i}{P_{\max}}\right). \tag{3}$$

where I and E are on behalf of the maximum immigration rate and maximum emigration rate, respectively, S_i denotes the number of species of habitat H_i, $S_{\max}$ represents the maximum species numbers of every habitat, when the immigration rate and emigration rate have the same value, S_0 represents the number of species at this time. $m_{\max}$ is the maximum mutation probability and its value is ensured by users, P_i is the probability of species [12], and $P_{\max}$ is the maximum probability of species. The migration operator and mutation operator are expressed as the following equations (4) and (5), respectively:

$$H_i\left(\mathrm{SIV}_j\right) \longleftarrow H_e\left(\mathrm{SIV}_j\right), \tag{4}$$

$$H_i\left(\mathrm{SIV}_j\right) \longleftarrow lb_j + \mathrm{rand} * \left(ub_j - lb_j\right). \tag{5}$$

In equation (4), H_i is the immigration habitat, H_e is the emigration habitat, and the value is obtained via roulette wheel selection, and $H_i(\mathrm{SIV}_j)$ denotes the jth SIV of the habitat H_i. In equation (5), H_i is the mutation habitat, and ub_j and lb_j are the maximum and minimum limit values of the jth SIV of the habitat H_i, respectively.

The main calculation process of the BBO algorithm is as follows:

Initialization: randomly initialize the solution population containing N individuals, the maximum number of iterations (MI), the maximum migration rate (I), the maximum migration rate (E), the maximum mutation probability ($m_{\max}$), and the number of elite solutions K. Calculate the fitness value of each solution according to the designed parameter values, and sort them from the best to the worst.

Iteration process: for each iteration, the migration rate and mutation rate of each solution are calculated, K elite solutions are retained, and then migration and mutation operations are performed according to equations (4) and (5). Then fitness values are calculated and solutions are sorted again, the K previously retained elite solutions are replaced with the worst K solutions, and finally, these solutions are sorted and the iteration ends.

Iteration stop condition: before satisfying the iteration stop condition (usually reaching the designed MI), the algorithm will execute step "Iteration process" in a loop. After satisfying the stopping condition, the first-ranked solution is the optimal solution and the algorithm ends. This is just the result of the algorithm running once.

2.2. Tissue-Like P System. The tissue-like P system [3] is a calculation model abstracted based on the structure and function of the cell population in the organization and is an extension of the cell-like P system. It does not have nonbasic membranes like the cell-like P system; it only has the basic membrane and the environment. Information is communicated by symport/antiport rules and evolutionary rules through the cell and the environment [7]. Similarly, certain rules and cell state requirements are satisfied, and objects can also communicate between cells.

A formal expression of tissue-like P system is as follows:

$$\prod = \left(\sum, \mu, \mathrm{syn}, \omega_1, \ldots, \omega_q, R, R', \sigma_0\right), \tag{6}$$

where $\sum$ denotes a nonempty finite alphabet of the system; μ represents the membrane structure of the system; $\mathrm{syn} \subseteq \{1, 2, \ldots, q\} \times \{1, 2, \ldots, q\}$ means the attach relationship between cells; $\omega_1, \ldots, \omega_q$ is a number of multisets initially existed in the cell; R is the collection of limited evolution rules; R' is a series of limited communication rules; and $\sigma_0 = \sigma_{\mathrm{out}}$, σ_0 is the output area that can store the results.

3. Methods

BBO is a simple optimization algorithm with a single mode. It can find the best local and global solutions to balance exploration and exploitation capabilities. However, there are still shortcomings such as limited exploration capabilities and slow convergence speed. Therefore, in connection with the shortcomings of the BBO algorithm, some improvements were made to improve the effect which was DCGBBO. Later, a hierarchical tissue-like P system with triggering ablation rules was proposed, and it was combined with DCGBBO, and the extremely parallel principle of the P system was used in order to improve the efficiency of the algorithm, for the reason that the optimal solution could obtain.

3.1. Improved BBO (DCGBBO)

3.1.1. Dynamic Crossover Migration. In order to improve the local search ability preferably, increase the species diversity, and enhance the exploitation ability, dynamic crossover migration operator is designed to replace equation (4) and it is described as following equations:

$$H_i\left(\mathrm{SIV}_j\right) \longleftarrow H_R\left(\mathrm{SIV}_j\right) + \alpha * (1 - 2 * \mathrm{rand}) * \left(H_i\left(\mathrm{SIV}_j\right) - H_R\left(\mathrm{SIV}_j\right)\right), \tag{7}$$

$$\alpha = 2 * \left(1 - \frac{t}{\mathrm{MI}}\right), \tag{8}$$

where α is a cross-scaling factor and H_R is a habitat that is worked out by the roulette wheel selection.

The dynamic crossover migration operator is similar to [30]. In [30], H_R is chosen by the example learning selection [35] rather than the roulette wheel selection in this paper. In our opinion, the example learning selection will only choose migration habitat from good habitats, and the lack of randomness of selection will increase the possibility of the algorithm falling into local optimization. Consequently, the roulette wheel selection is preserved. α is inspired by the factor in GWO [16], which can adjust dynamically as the number of iterations changes. The range of α is 0 to 2, which is linearly decreased, and the range of $(1 - 2 * \mathrm{rand})$ values is from -1 to 1. In the early step of the iterations, the value of $\alpha * (1 - 2 * \mathrm{rand})$ is relatively large, so a relatively large disturbed value can be received, the diversity of characteristics can be increased, and the global search ability can be enhanced simultaneously. Later in the iteration, $\alpha * (1 - 2 * \mathrm{rand})$ is relatively small, the algorithm will continue to optimize in the direction of convergence, and the local ability is strengthened at the same time.

3.1.2. Dynamic Gaussian Mutation. For mutation strategies, the authors in [36] introduced a novel mutation rule that the population was divided into the best, better, and worst groups, one of each partition to implement the mutation process. In [37], two mutation strategies are introduced and a hybridization framework is proposed to improve the algorithm performance. Gong et al. [38] introduced Gaussian, Cauchy, and Levy mutation operators into BBO for real space. In this paper, we proposed an improved dynamic Gaussian mutation operator to speed up convergence, reduce algorithm complexity, and improve exploration capability.

The probability density function of the Gaussian distribution is described as follows:

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right], \tag{9}$$

where μ is the mean and σ is the standard deviation. Then, the dynamic Gaussian mutation operator with $\mu = 0$ and $\sigma = 1$ is presented in equations (10) and (11), in which a is a user-delimited parameter and $a = 0.02$ and ub_j and lb_j is the maximum and minimum limit values of the jth SIV of the habitat H_i, respectively.

$$\begin{cases} H_i\left(\mathrm{SIV}_j\right) \longleftarrow H_i\left(\mathrm{SIV}_j\right) + \delta * \mathrm{randn}(0,1), & t \le \dfrac{\mathrm{MI}}{2}, \\ H_i\left(\mathrm{SIV}_j\right) \longleftarrow H_i\left(\mathrm{SIV}_j\right) + \left(1 - \dfrac{t}{\mathrm{MI}}\right) * \mathrm{randn}(0,1), & \text{else}, \end{cases} \tag{10}$$

$$\delta = a * \left(ub_j - lb_j\right). \tag{11}$$

In the first half of the iteration, δ is a constant value and it depends on the maximum and minimum limit values of the feature, so the convergence is accelerated and the global search capability is enhanced too. As the value of t increases, the iteration enters the second half, and then $(1 - (t/\mathrm{MI}))$ goes down, disturbed value also goes down, the solution continues to search for the optimal value along the direction of convergence in the first stage, and the local search ability and exploration capability are improved.

The occurrence of random events is uncertain, so the HSI of the habitat will abruptly change, so there is a mutation operator in BBO. The mutation probability of the original standard BBO algorithm is obtained from species count probabilities; there is a relatively low probability for a small number of species or a large number of species and a high probability when the number of species approaches the equilibrium point. However, as the occurrence of unexpected events is uncertain, obtaining the mutation probability based on species count probabilities cannot guarantee the correctness and feasibility of the mutation, so in this paper, each habitat has the same probability of mutation, that is, maximum mutation rate $m_{\max}$.

3.1.3. Opposition-Based Learning Approach. Because the setting of the initial solution is relatively random, the diversity of the population is constrained, and the direction and degree of migration and mutation of each solution are different, which could lead the optimization process to get the local optimum value without jumping out of the limit. As a result, the optimal solution cannot be found and the running time of the algorithm is consumed. Some methods [35,39,40] have been produced to avoid the algorithm falling into local optima in some respects. And a new operator is designed in equation (12) where $H_l(\mathrm{SIV}_j)$ is the last that is the worst habitat.

$$H_l\left(\mathrm{SIV}_j\right) \longleftarrow lb_j + \left(ub_j - H_l\left(\mathrm{SIV}_j\right)\right). \tag{12}$$

If the algorithm finds a better value at a certain moment, the value of the solution remains unchanged after subsequent iterations, but it is not the optimal value. At this time, equation (12) can be used to obtain a new value. And then according to the new value, the solution space might be changed or keep the original order.

The pseudocode of DCGBBO is described in Algorithm 1.

Algorithm start
Initialization:
Randomly initialize N solutions, and set up I, E, MI, and $m_{\max}$.
Calculate the value of HSI of every solution and then sort every solution from the best to the worst according to their HSIs.
Compute the immigration and emigration and preserve K elite habitats.
Iteration process:
for $t = 1$ to MI//iteration stop condition
  for $i = 1$ to N
    if $i == N$ then
      for $j = 1$ to D
        Execute the opposition-based learning approach that is equation (12).
      end for
    else
      for $j = 1$ to D//migration operator
        if rand $< \lambda_i$
          Carry out the roulette wheel selection to screen out emigration habitat H_e, and perform equations (7) and (8) to renew $H_i(\mathrm{SIV}_j)$.
        end if
        if rand $< m_i$ //mutation operator
          Perform equations (10) and (11) to renew $H_i(\mathrm{SIV}_j)$.
        end if
      end for
    end
  end
Compute the fitness (HSI) of every habitat and sort every solution from the best to the worst on the basis of their HSIs again.
Comply the elite operator and perform an exchange of poor solution with elite solution
Sort every solution from the best to the worst via their HSIs finally.
end for
Algorithm finished

Algorithm 1: The main process of DCGBBO.

3.2. Hierarchical Tissue-Like P System with Triggering Ablation Rules (HTPTA) for DCGBBO. In this part, the DCGBBO algorithm based on HTPTA is proposed and named as DCGBBO-HTPTA. We introduced a new trigger mechanism, defined ablation rules and replication rules, and applied the evolutionary rules and communication rules between membranes. In the algorithm, an individual (potential solution) is an object in the membrane of the P system. All objects in a membrane can be regarded as a population, and a cell membrane can be regarded as a subpopulation. Therefore, the optimization of the algorithm can be completed by the evolution rules between individuals (potential solutions) in a membrane and the communication rules between the membrane and the membrane. Hierarchical tissue-like P system with triggering ablation rules with $2q + 1$ cell membranes, which is designed for the DCGBBO algorithm, is shown in Figure 1. The P system is constructed just as the following form:

$$\prod = \left(\sum, \mu, \sigma_0, \sigma_1, \ldots, \sigma_{2q+1}, \omega_1, \ldots, \omega_q, R_{2/3,q}, R_1', R_{2/3,q}', R_{2/3,q}^c, \beta, R_a, \sigma_{\text{out}}\right), \tag{13}$$

where

$\sum$ is an alphabet of the system, and letters correspond to objects in the DCGBBO-HTPTA.

μ represents a 3-layer membrane structure, which contains $2q + 1$ cell membranes.

σ_0 represents environment. $\sigma_1, \sigma_2, \ldots, \sigma_{2q+1}$ are $2q + 1$ cell membranes. σ_{2q+1} is the global cell membrane, which store the optimal solution for each subblock after the algorithm ends.

$\omega_1, \ldots, \omega_q$ express the collection of initial objects in q cell membrane, and ω_i denotes the collection of initial objects in $\sigma_{2,i}$.

$R_{2/3,q}$ represents object-based evolution rules within the membranes. $R_{2/3,i}$ shows the evolutional rule which is executed in membrane $\sigma_{2/3,i}$, and the specific form is $[u]_{2,i} \longrightarrow [v]_{2,i}$, which means object u evolved to v in membrane $\sigma_{2,i}$.

$R_1', R_{2/3,q}'$ indicate the communication symport rules from membrane $\sigma_{2/3,i}$ to membrane $\sigma_{1/2/3,i}$, and the form is $[u]_{2,i}[]_{3,i} \longrightarrow []_{2,i}[u]_{3,i}$, which means the object u in the membrane $\sigma_{2,i}$ entered the membrane $\sigma_{3,i}$.

$R_{2/3,q}^c$ represents copy-rules of the object in membrane $\sigma_{2/3,i}$, and the form is $[u]_{2,i} \longrightarrow [uu]_{2,i}$, which means there are two u in the membrane $\sigma_{2,i}$ after applying the copy-rules.

β is a trigger which exists in the second and third layer of cell membranes. When a specific condition is reached, there are only K objects left in the membrane $\sigma_{2/3,q}$, and the sequence numbers of these K objects are

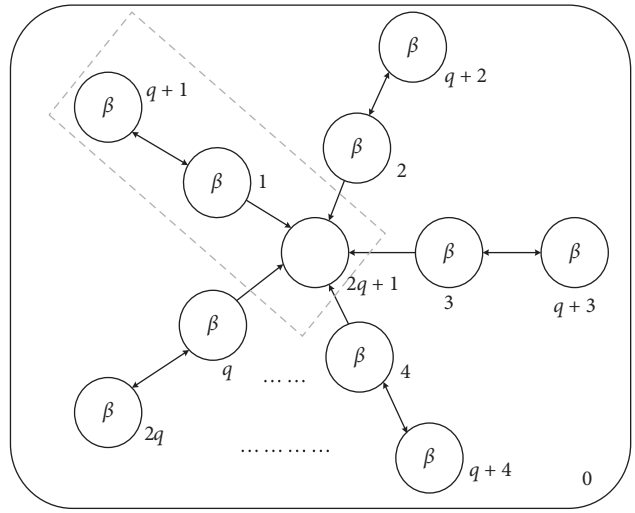

Figure 1: Structure of hierarchical tissue-like P system with triggering ablation rules for DCGBBO algorithm.

$(N-k+1, N)$, the trigger β is activated, and the ablation rules R_a is executed, and the form is $[u]_{2,i} \longrightarrow []_{2,i}$, which indicates that the object u in membrane $\sigma_{2,i}$ is ablated when certain conditions are met.

σ_{out} is the output cell membrane, the optimal solution in membrane σ_1 will be output to the output membrane σ_{out}, and the environment is the output area of the algorithm.

For μ, there are one cell membrane σ_1 in the first layer and q cell membranes in the second layer and third layer denoted by $\sigma_{2,1}, \ldots, \sigma_{2,q}$ and $\sigma_{3,1}, \ldots, \sigma_{3,q}$, respectively. The cell membrane corresponding to every 3 layers is a subblock, so there are q subblocks.

Calculation process.

(1) Initialize the number of cell membranes $(2q+1)$, the number of initial solutions (objects) (N), the value of initial objects in each cell in the second layer, and the number of elite objects K. Calculate the fitness value $(f(x))$ of each object in the initial cell and sort them, calculate the emigration rate and the immigration rate, and start the iteration.

(2) Execute the copy-rule, copy the objects with the serial number $1-K$, and execute the communication rules to send these K objects to the corresponding third layer of cells.

(3) Using the extremely parallel rule, update each object according to formulas (7), (8), and (10)–(12) and sort again, and then send the object with sequence number $1-(N-K)$ to the corresponding third layer of cells.

(4) At this time, there are only K worst objects left in the second layer of cells, and certain conditions are met, so that the trigger is activated, the ablation rule is executed, K objects are ablated, and there are no more objects in the second layer of cells. There are N objects in the third layer of cells. At this point, the first iteration is over.

(5) The second iteration in the third layer of cells is started until the MI is met; otherwise, repeat steps 2–4. When the calculation stops, the optimal object that is the optimal solution from the membrane will be sent to the environment.

4. Experiments and Analysis

In order to test the optimal performance of DCGBBO, some experiments are implemented on a series of classic benchmark functions. The operating system of the experiment was Microsoft Windows 7 on PC, with 1.70 GHz CPU and 4 GB memory. The experimental environment is MATLAB R2017b.

4.1. Experiment Setting. The classic benchmark functions used to testify the optimization efficiency include the traditional continuous unimodal functions (f_3-f_5), which are implemented to evaluate the exploitation ability of the algorithm, the multimodal functions (f_6-f_8), which are applied to evaluate the exploration ability of the algorithm, a step function, which just obtain one minimum and is discrete (f_1), and a noisy quartic function (f_2). The particular information of these classic benchmark functions is demonstrated in Table 1. CEC 2017 contains novel basic problems, composing test problems, rotated trap problems, graded level linkages, and many other problems. In order to further verify the optimization ability of DCGBBO to deal with complex problems, a large number of experiments were carried out on CEC 2017 [41] benchmark functions.

4.2. Experiment Results

4.2.1. Experiment Results and Analysis on Classic Benchmark Functions. The DCGBBO-HTPTA is compared with standard BBO [12] algorithm and three BBO variants which include B-BBO [27], BBO-M [29], and TDBBO [32] on 8 benchmark functions and some CEC 2017 benchmark

Table 1: Benchmark functions.

Name	Function	Search range	$f_{\min}$
Step	$f_1(x) = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	$[-100, 100]^D$	0
Quartic	$f_2(x) = \sum_{i=1}^{D} i x_i^4 + \text{random}[0, 1)$	$[-1.28, 1.28]^D$	0
Sphere	$f_3(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
Schwefel 2.22	$f_4(x) = \sum_{i=1}^{D} \lvert x_i \rvert + \prod_{i=1}^{D} \lvert x_i \rvert$	$[-10, 10]^D$	0
Zakharov	$f_5(x) = \sum_{i=1}^{D} x_i^2 + (\sum_{i=1}^{D} 0.5 i x_i)^2 + (\sum_{i=1}^{D} 0.5 i x_i)^4$	$[-5, 10]^D$	0
Griewank	$f_6(x) = (1/4000) \sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos(x_i/\sqrt{i}) + 1$	$[-600, 600]^D$	0
Ackley	$f_7(x) = 20 + e - 20 \exp(-0.2 \sqrt{(1/D) \sum_{i=1}^{D} x_i^2}) - \exp((1/D) \sum_{i=1}^{D} \cos 2\pi x_i)$	$[-30, 30]^D$	0
Alpine	$f_8(x) = \sum_{i=1}^{D} \lvert x_i \sin(x_i + 0.1 x_i) \rvert$	$[-10, 10]^D$	0

functions to testify the performance of themselves. The parameters of the comparison algorithm are shown in Table 2. To be fair, the algorithm's public parameters are set to the same. For classic benchmarks functions, the number of independent runs (Num) of all the algorithms is 30, and the population size is 50 ($N = 50$). And on the 30D functions, MI is 2500, maximum number of function evaluation (MNFE) is MI $*$ N. On the 50D functions, MI is 3500, and MNFE also is MI $*$ N.

In this experiment part, the mean value (Mean) and the standard deviation value (Std) are used as evaluate indicators to testify the optimization performance of the algorithms. For the algorithm, the Mean value represents the optimization ability and the Std value represents the stability. So the Mean value is the key point of the comparison.

From Table 3, we can reach that DCGBBO is the first of the 7 functions and is the second of 1 function on 30D optimization problem. TDBBO is the first of 2 functions, and B-BBO and BBO-M are the first of 1 function, respectively, as evaluate indicators are equal. In the traditional continuous unimodal function (f_3-f_4), DCGBBO is better than other algorithms except f_5 and is slightly inferior to TDBBO, but in the multimodal functions (f_6-f_8), a step function (f_1), and a noisy quartic function (f_2), DCGBBO is distinctly better than other algorithms which shows that dynamic Gaussian mutation enhances the global search ability and exploration ability of the algorithm indeed. On function f_1, DCGBBO gets the optima value (0) which proves that DCGBBO has great convergence performance and optimization performance. From Table 4, we can conclude that DCGBBO is the first of the 7 functions and is the second of 1 function on 50D benchmark functions. B-BBO is the first of 2 functions, and BBO-M and TDBBO are the first of 1 function, respectively, as evaluate indicators are equal. In the traditional continuous unimodal function (f_3-f_5), DCGBBO is better than other algorithms, which shows that the dynamic crossover migration enhances the exploitation ability and local search ability. In the multimodal functions (f_6-f_8), DCGBBO ranks first and the evaluation index is obviously better than that of other algorithms; this shows that dynamic Gaussian mutation enhances the global search ability and exploration ability. On f_1, all BBO variants can get the best value (0) except standard BBO. On f_2, DCGBBO is slightly inferior to B-BBO, but the difference in the evaluation index is very small.

We get the convergence curves of 5 comparison algorithms on 8 classic functions shown in Figures 2 and 3. The abscissa "Iteration" gives the number of iterations, and the ordinate "f value" gives the optimization functions value of 8 benchmark functions tested by 5 comparison algorithms. From Figure 2, no matter which function, DCGBBO's convergence performance is better than that of other algorithms. Although on functions f_2 and f_8, DCGBBO's convergence speed is inferior to B-BBO, DCGBBO's convergence speed is obviously faster than that of other comparison algorithms. From Figure 3, on 50-dimension benchmark functions, the proposed algorithm has the faster convergence speed than other algorithms except f_1, f_2, and f_8, which has a slightly faster speed of convergence than DCGBBO. But the convergence efficiency of DCGBBO is significantly superior to other comparison algorithms. To sum up, DCGBBO has the best convergence performance than other improved algorithms.

4.2.2. Experiment Results and Analysis on CEC 2017 Benchmark Functions. In this paper, for CEC 2017 benchmark functions, the number of independent runs (Num) of all the algorithms is 51, and the population size is 100 ($N = 100$). Maximum number of function evaluation (MNFE) are 100000, 300000, and 500000 for $D = 10$, 30, and 50. The mean value (Mean) and the standard deviation value (Std) are used as evaluate indicators to testify the optimization performance of the algorithms. For the algorithm, the Mean value represents the optimization ability and the Std value represents the stability. So the Mean value is the key point of the comparison.

Table 5 depicts the comparison of the DCGBBO with considered algorithms on CEC 2017 benchmark functions in terms of mean and best fitness values, and the dimension is 10. From the table, it can be stated that the mean values (Mean) and the standard deviation value (Std) returned by DCGBBO are better than those of the considered algorithms for twenty-four and twenty-one problems out of thirty,

Table 2: Parameter setting of the experimental algorithm.

Algorithm	Parameter setting
BBO	$N = 50$ or 100, $I = 1$, $E = 1$, $m_{\max} = 0.01$, $K = 10$
B-BBO	$N = 50$ or 100, $I = 1$, $E = 1$, $m_{\max} = 0.01$, $K = 10$, α = random
BBO-M	$N = 50$ or 100, $I = 1$, $E = 1$, $m_{\max} = 0.01$, $K = 10$, $x_{n+1} = 4 * x_n(1 - x_n)$
TDBBO	$N = 50$ or 100, $I = 1$, $E = 1$, $m_{\max} = 0.01$, $K = 10$, $c = 0.3$
DCGBBO	$N = 50$ or 100, $I = 1$, $E = 1$, $m_{\max} = 0.01$, $K = 10$, $a = 0.02$

Table 3: Comparison results between DCGBBO and BBO variants on classic benchmark functions ($D = 30$).

Function	Value	BBO	B-BBO	BBO-M	TDBBO	DCGBBO
	Mean	$2.6667E-01$	$0.0000E+00$	$0.0000E+00$	$0.0000E+00$	$0.0000E+00$
f_1	Std	$4.4222E-01$	$0.0000E+00$	$0.0000E+00$	$0.0000E+00$	$0.0000E+00$
	Rank	2	1	1	1	1
	Mean	$1.9511E-02$	$6.1618E-03$	$3.6860E-02$	$2.8988E-02$	$4.1232E-03$
f_2	Std	$6.4387E-03$	$2.5473E-03$	$1.1431E-02$	$1.0614E-02$	$1.3524E-03$
	Rank	3	2	5	4	1
	Mean	$7.0000E-01$	$4.3405E-08$	$5.0172E-05$	$4.2146E-07$	$1.8894E-16$
f_3	Std	$2.8581E-01$	$1.2372E-07$	$3.3052E-05$	$3.1749E-07$	$6.3309E-16$
	Rank	5	2	4	3	1
	Mean	$3.1585E-01$	$8.9749E-06$	$5.0074E-03$	$1.0957E-03$	$7.1868E-10$
f_4	Std	$4.4929E-02$	$1.3574E-05$	$1.9967E-03$	$3.0729E-04$	$1.1770E-09$
	Rank	5	2	4	3	1
	Mean	$9.1743E+01$	$2.0162E+01$	$8.6091E-02$	$2.4515E-02$	$4.4026E-02$
f_5	Std	$2.1962E+01$	$7.8399E+00$	$3.1246E-02$	$1.4351E-02$	$2.6713E-02$
	Rank	5	4	3	1	2
	Mean	$7.1557E-01$	$2.2409E-02$	$2.1025E-02$	$7.2223E-02$	$9.6829E-03$
f_6	Std	$1.7063E-01$	$2.3567E-02$	$2.1008E-02$	$1.0019E-01$	$9.4377E-03$
	Rank	5	3	2	4	1
	Mean	$2.9384E-01$	$1.4242E-05$	$2.9104E-03$	$4.0678E-04$	$5.2792E-09$
f_7	Std	$9.0291E-02$	$1.1358E-05$	$2.6930E-03$	$1.2802E-04$	$5.7223E-09$
	Rank	5	2	4	3	1
	Mean	$7.3721E-03$	$4.2457E-06$	$6.3157E-03$	$1.4100E-04$	$2.8565E-06$
f_8	Std	$2.8770E-03$	$1.4019E-05$	$3.6322E-03$	$1.3572E-04$	$1.1144E-05$
	Rank	5	2	4	3	1
Count		0	1	1	2	7
Average rank		4.375	2.25	3.375	2.75	1.125
Total rank		5	2	4	3	1

Table 4: Comparison results between DCGBBO and BBO variants on classic benchmark functions ($D = 50$).

Function	Value	BBO	B-BBO	BBO-M	TDBBO	DCGBBO
	Mean	$2.6667E-01$	$0.0000E+00$	$0.0000E+00$	$0.0000E+00$	$0.0000E+00$
f_1	Std	$5.7349E-01$	$0.0000E+00$	$0.0000E+00$	$0.0000E+00$	$0.0000E+00$
	Rank	2	1	1	1	1
	Mean	$3.5416E-02$	$5.0596E-03$	$1.3334E-01$	$1.0006E-01$	$8.8093E-03$
f_2	Std	$1.2171E-02$	$1.6088E-03$	$5.1771E-02$	$2.7291E-02$	$2.3044E-03$
	Rank	3	1	5	4	2
	Mean	$9.4072E-01$	$7.6629E-07$	$1.1455E-03$	$3.6507E-05$	$2.8116E-11$
f_3	Std	$2.6482E-01$	$6.9881E-07$	$5.0193E-04$	$1.2786E-05$	$1.2494E-11$
	Rank	5	2	4	3	1
	Mean	$4.6046E-01$	$1.1890E-04$	$6.0901E-02$	$1.8880E-02$	$1.7604E-06$
f_4	Std	$7.0598E-02$	$5.7422E-05$	$1.4203E-02$	$4.0873E-03$	$7.1498E-07$
	Rank	5	2	4	3	1
	Mean	$1.5938E+02$	$1.1296E+01$	$8.8797E-01$	$4.7962E-01$	$3.4126E-01$
f_5	Std	$3.9232E+01$	$5.6901E+00$	$1.9818E-01$	$1.4686E-01$	$1.6971E-01$
	Rank	5	4	3	2	1
	Mean	$6.9633E-01$	$1.3972E-02$	$1.6892E-02$	$4.7606E-02$	$5.2553E-03$
f_6	Std	$1.1879E-01$	$2.0940E-02$	$1.3186E-02$	$4.7559E-02$	$6.7809E-03$
	Rank	5	2	3	4	1

Table 4: Continued.

Function	Value	BBO	B-BBO	BBO-M	TDBBO	DCGBBO
	Mean	$2.0238E-01$	$1.4413E-04$	$5.5962E-01$	$3.2706E-03$	$2.4462E-06$
f_7	Std	$5.5586E-02$	$4.1983E-05$	$2.0181E+00$	$7.1758E-04$	$1.1495E-06$
	Rank	4	2	5	3	1
	Mean	$9.4789E-03$	$1.3125E-05$	$8.7008E-02$	$3.1986E-03$	$2.2021E-06$
f_8	Std	$2.8390E-03$	$5.4439E-05$	$3.0606E-02$	$1.8003E-03$	$2.6616E-06$
	Rank	4	2	5	3	1
Count		0	2	1	1	7
Average rank		4.125	2	3.75	2.875	1.125
Total rank		5	2	4	3	1

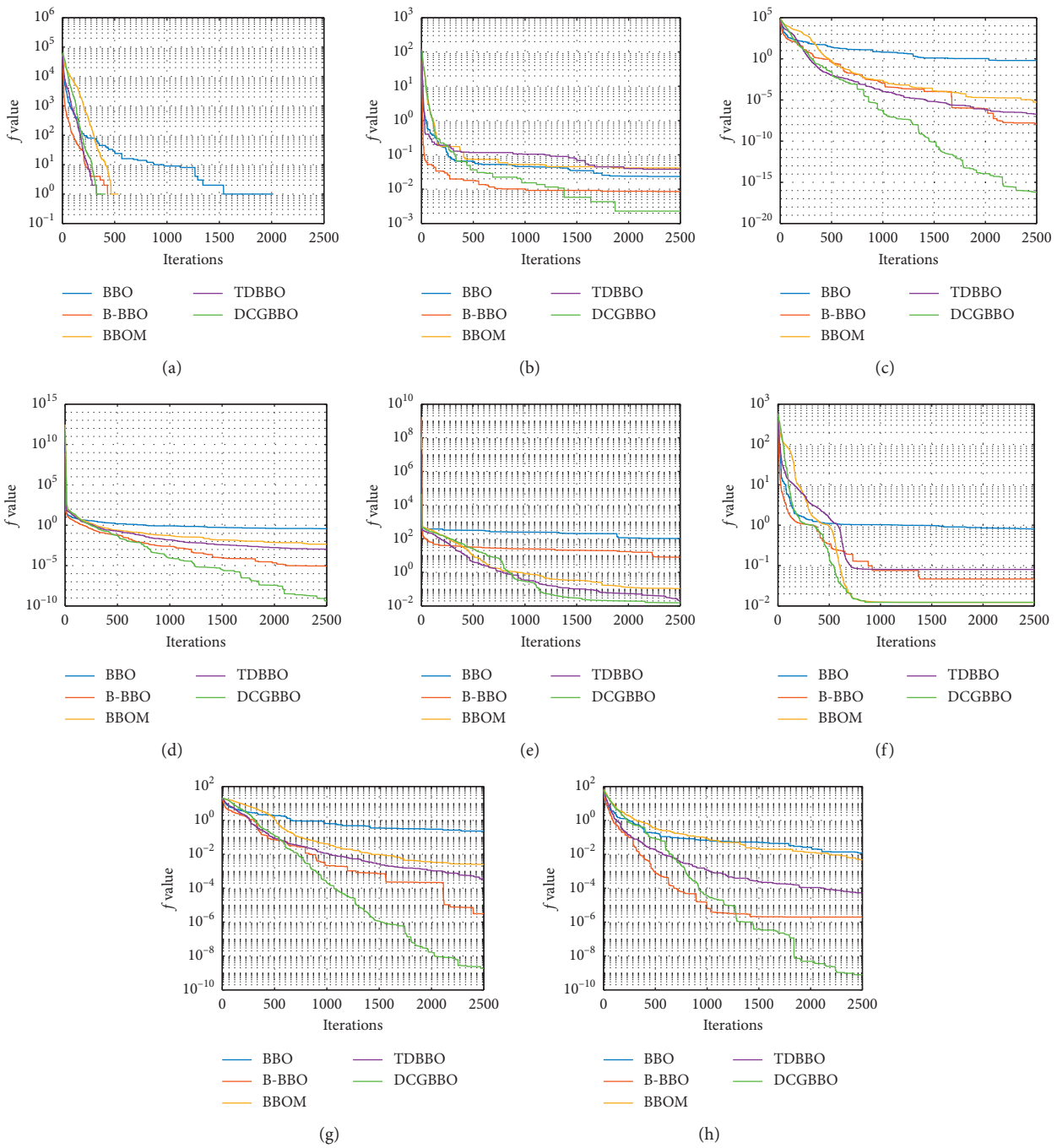

Figure 2: Convergence curves of 5 algorithms on 30D classic benchmark functions. (a) f_1. (b) f_2. (c) f_3. (d) f_4. (e) f_5. (f) f_6. (g) f_7. (h) f_8.

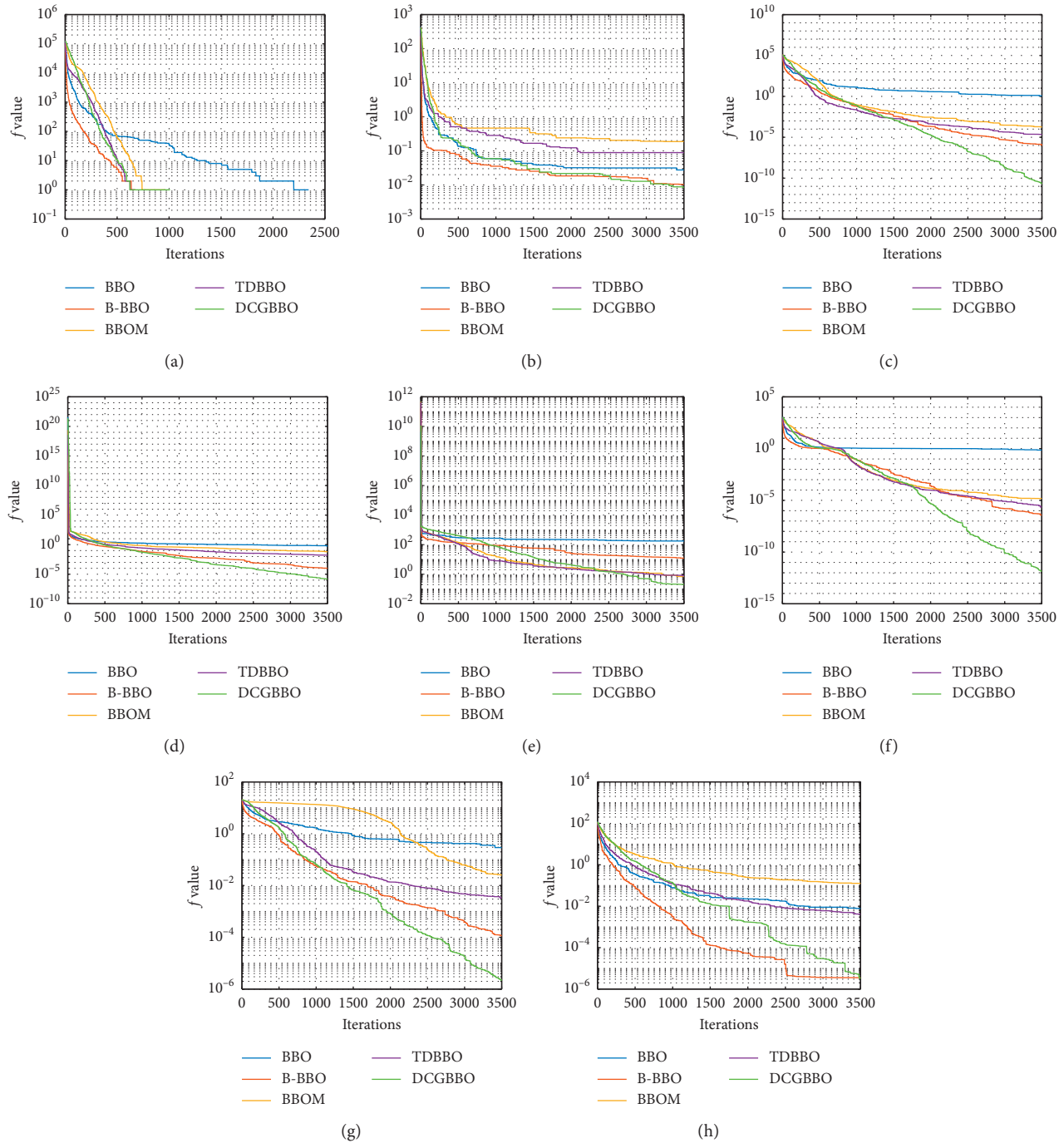

Figure 3: Convergence curves of 5 algorithms on 50D classic benchmark functions. (a) f_1. (b) f_2. (c) f_3. (d) f_4. (e) f_5. (f) f_6. (g) f_7. (h) f_8.

respectively. For $D = 30$ dimensions, the comparative results are shown in Table 6. From the table, we can see that DCGBBO outperforms the considered algorithms on twenty-two problems for the mean values (Mean) and on twenty-one problems for the standard deviation values (Std). And from Table 7 for $D = 50$ dimensions, DCGBBO outperforms the considered algorithms on twenty-three problems for the mean values (Mean) and on twenty-two problems for the standard deviation values (Std). In short, DCGBBO has better optimization efficiency than other modified BBO variants and standard BBO. In particular, as the dimension of the solution space increases, the performance of the algorithm proposed in this paper gradually stabilizes, except for a few more sensitive functions.

Furthermore, the convergence behavior of DCGBBO is analyzed for each class of CEC 2017 benchmark problems. Figure 4 shows the convergence trend of four 10-dimensional benchmark problems, respectively, from F1, F4, F11, and F24. In the same way, Figures 5 and 6 show the convergence trend of four 30-dimensional and 50-dimensional

Table 5: Comparison results on CEC 2017 benchmark functions ($D = 10$).

Function	Value	BBO	B-BBO	BBO-M	TDBBO	DCGBBO
F1	Mean	$5.9836E+05$	$3.3868E+05$	$4.4536E+03$	$3.3972E+04$	$3.9464E+03$
	Std	$3.9932E+05$	$2.3529E+05$	$3.6243E+03$	$3.9979E+03$	$3.6035E+03$
F2	Mean	$2.3599E+05$	$4.4616E+04$	$5.3770E+02$	$2.0010E+02$	$2.0003E+02$
	Std	$6.9658E+05$	$1.1562E+05$	$9.0270E+02$	$6.5937E+01$	$1.7951E-01$
F3	Mean	$2.0279E+04$	$1.6642E+04$	$3.0001E+02$	$3.0000E+02$	$3.0000E+02$
	Std	$1.0393E+04$	$1.8086E+04$	$2.5682E-02$	$3.4721E-06$	$2.5348E-06$
F4	Mean	$4.2173E+02$	$4.0596E+02$	$4.0892E+02$	$4.0856E+02$	$4.0442E+02$
	Std	$3.3960E+01$	$1.9164E+01$	$2.0448E+01$	$2.0446E+01$	$1.2810E+01$
F5	Mean	$5.1087E+02$	$5.1183E+02$	$5.2892E+02$	$5.2317E+02$	$5.2023E+02$
	Std	$4.6457E+00$	$5.5670E+00$	$1.0769E+01$	$1.2706E+01$	$7.6739E+00$
F6	Mean	$6.0946E+02$	$6.0528E+02$	$6.3145E+02$	$6.1692E+02$	$6.0366E+02$
	Std	$1.5806E-01$	$2.6333E-01$	$1.6474E+01$	$1.0657E+01$	$4.9938E+00$
F7	Mean	$7.3566E+02$	$7.2397E+02$	$7.5311E+02$	$7.3523E+02$	$7.3133E+02$
	Std	$7.6800E+00$	$5.1542E+00$	$1.7394E+01$	$1.0736E+01$	$6.0922E+00$
F8	Mean	$8.1574E+02$	$8.1549E+02$	$8.4917E+02$	$8.3194E+02$	$8.3115E+02$
	Std	$5.1018E+00$	$3.3043E+00$	$2.2043E+01$	$1.1538E+01$	$8.8128E+00$
F9	Mean	$9.1496E+02$	$9.0605E+02$	$1.5136E+03$	$1.1042E+03$	$1.0599E+03$
	Std	$1.2445E+01$	$6.7970E+00$	$5.2150E+02$	$2.0207E+02$	$1.0683E+02$
F10	Mean	$1.6931E+03$	$1.6626E+03$	$1.8659E+03$	$1.9533E+03$	$1.6221E+03$
	Std	$2.3580E+02$	$2.5230E+02$	$2.4510E+02$	$2.1323E+02$	$1.8947E+02$
F11	Mean	$1.2814E+03$	$1.3902E+03$	$1.2090E+03$	$1.1792E+03$	$1.1555E+03$
	Std	$2.5309E+02$	$4.5194E+02$	$6.0116E+01$	$5.8805E+01$	$5.1934E+01$
F12	Mean	$5.3143E+06$	$1.2458E+06$	$2.0959E+05$	$5.5021E+04$	$3.2888E+04$
	Std	$4.6787E+06$	$2.5726E+06$	$3.2162E+05$	$8.9702E+04$	$2.1839E+04$
F13	Mean	$2.6624E+04$	$1.4218E+04$	$1.2732E+04$	$1.3376E+04$	$1.1020E+04$
	Std	$2.9560E+04$	$9.2783E+03$	$1.2678E+04$	$7.7535E+03$	$1.1982E+03$
F14	Mean	$1.0534E+04$	$8.3992E+03$	$7.2156E+03$	$7.8535E+03$	$7.0313E+03$
	Std	$9.1737E+03$	$7.0081E+03$	$7.4258E+03$	$7.3835E+03$	$5.7340E+03$
F15	Mean	$9.4708E+03$	$3.8642E+03$	$7.3408E+03$	$4.7368E+03$	$1.0771E+04$
	Std	$7.1208E+03$	$2.9039E+03$	$5.3490E+03$	$4.5840E+03$	$8.7666E+03$
F16	Mean	$1.8861E+03$	$1.8211E+03$	$1.8469E+03$	$1.8164E+03$	$1.8056E+03$
	Std	$1.3616E+02$	$1.2774E+02$	$2.7480E+02$	$1.6961E+02$	$1.1233E+02$
F17	Mean	$1.8244E+03$	$1.7542E+03$	$1.8489E+03$	$1.8062E+03$	$1.7259E+03$
	Std	$4.0171E+01$	$6.7869E+01$	$8.7174E+01$	$3.8909E+01$	$1.0002E+02$
F18	Mean	$2.8159E+04$	$1.5797E+04$	$1.3360E+04$	$1.8035E+04$	$1.3628E+04$
	Std	$1.4967E+04$	$1.1278E+04$	$1.0592E+04$	$1.1003E+04$	$1.0383E+04$
F19	Mean	$1.7815E+04$	$9.1510E+03$	$1.6276E+04$	$8.7426E+03$	$1.3668E+03$
	Std	$1.2344E+04$	$8.4528E+03$	$1.1888E+04$	$5.8878E+03$	$1.1598E+03$
F20	Mean	$2.0168E+03$	$2.0189E+03$	$2.0888E+03$	$2.1190E+03$	$2.0454E+03$
	Std	$8.4723E+00$	$6.1012E+01$	$4.1748E+01$	$7.1002E+01$	$4.0318E+01$
F21	Mean	$2.3095E+03$	$2.3032E+03$	$2.3187E+03$	$2.3128E+03$	$2.3030E+03$
	Std	$3.2624E+01$	$4.5667E+01$	$6.0807E+01$	$6.0903E+01$	$3.2062E+01$
F22	Mean	$2.3067E+03$	$2.3988E+03$	$2.4428E+03$	$2.4588E+03$	$2.3053E+03$
	Std	$4.4024E+00$	$2.7549E+02$	$3.4655E+02$	$4.2801E+02$	$4.2451E+00$
F23	Mean	$2.6249E+03$	$2.6278E+03$	$2.6317E+03$	$2.6305E+03$	$2.6248E+03$
	Std	$4.9188E+00$	$6.0687E+00$	$1.0194E+01$	$8.0294E+00$	$4.7604E+00$
F24	Mean	$2.7623E+03$	$2.7727E+03$	$2.7858E+03$	$2.8045E+03$	$2.7606E+03$
	Std	$8.3267E+01$	$9.1873E+01$	$5.1557E+01$	$8.7950E+01$	$4.0653E+01$
F25	Mean	$2.9424E+03$	$2.9428E+03$	$2.9429E+03$	$2.9500E+03$	$2.9410E+03$
	Std	$1.3550E+01$	$2.0553E+01$	$3.5638E+01$	$3.3467E+01$	$2.8726E+01$
F26	Mean	$3.0793E+03$	$3.0256E+03$	$3.1166E+03$	$3.1222E+03$	$2.9502E+03$
	Std	$3.1888E+02$	$1.2137E+02$	$3.0811E+02$	$2.6391E+02$	$7.4867E+01$
F27	Mean	$3.1173E+03$	$3.1223E+03$	$3.1318E+03$	$3.1396E+03$	$3.1014E+03$
	Std	$2.8662E+01$	$2.7372E+01$	$2.3107E+01$	$3.9406E+01$	$1.0045E+01$
F28	Mean	$3.2998E+03$	$3.3459E+03$	$3.3178E+03$	$3.3258E+03$	$3.2963E+03$
	Std	$1.4596E+02$	$1.4968E+02$	$1.4401E+02$	$1.9021E+02$	$1.4367E+02$
F29	Mean	$3.2447E+03$	$3.2584E+03$	$3.2414E+03$	$3.2478E+03$	$3.2397E+03$
	Std	$8.3164E+01$	$2.8049E+01$	$6.8896E+01$	$5.6400E+01$	$3.5545E+01$
F30	Mean	$9.4055E+05$	$7.9785E+05$	$6.4566E+05$	$5.3678E+05$	$2.0345E+05$
	Std	$9.1204E+05$	$7.3106E+05$	$6.8573E+05$	$7.9743E+05$	$4.3233E+05$

Table 6: Comparison results on CEC 2017 benchmark functions ($D = 30$).

Function	Value	BBO	B-BBO	BBO-M	TDBBO	DCGBBO
F1	Mean	2.0905$E+06$	3.5688$E+03$	7.7199$E+03$	9.4396$E+03$	3.4986$E+03$
	Std	9.0756$E+05$	4.3978$E+03$	7.9901$E+03$	8.2207$E+03$	4.1129$E+03$
F2	Mean	3.1966$E+13$	2.8591$E+12$	1.9300$E+13$	1.8739$E+10$	1.4756$E+10$
	Std	4.8093$E+13$	2.7083$E+12$	4.6269$E+13$	2.6501$E+11$	1.7861$E+11$
F3	Mean	1.5299$E+05$	1.2101$E+05$	3.9023$E+04$	6.9579$E+03$	6.2097$E+03$
	Std	6.1148$E+04$	4.1128$E+04$	1.3549$E+04$	4.7113$E+03$	2.6195$E+03$
F4	Mean	5.1936$E+02$	4.9314$E+02$	4.9515$E+02$	4.9914$E+02$	4.7975$E+02$
	Std	3.6142$E+01$	1.7435$E+01$	1.3408$E+01$	1.2598$E+01$	8.6073$E+00$
F5	Mean	5.8870$E+02$	6.0481$E+02$	7.8190$E+02$	7.2287$E+02$	6.7660$E+02$
	Std	1.1801$E+01$	1.1554$E+01$	5.0211$E+01$	9.8490$E+00$	1.6995$E+01$
F6	Mean	6.1057$E+02$	6.0030$E+02$	6.7059$E+02$	6.4922$E+02$	6.0079$E+02$
	Std	5.9632$E+00$	2.0303$E-03$	2.1465$E+01$	9.0771$E+00$	4.9564$E+00$
F7	Mean	8.1992$E+02$	8.5653$E+02$	1.9826$E+03$	1.2808$E+03$	9.3643$E+02$
	Std	6.9548$E+00$	1.9823$E+01$	5.7593$E+02$	1.2134$E+02$	4.6188$E+01$
F8	Mean	9.0031$E+02$	8.8258$E+02$	1.0723$E+03$	9.8745$E+02$	9.9107$E+02$
	Std	1.4242$E+01$	3.3716$E+01$	1.0359$E+02$	4.0662$E+01$	6.1861$E+01$
F9	Mean	4.3912$E+03$	1.7531$E+03$	1.2094$E+04$	5.8881$E+03$	5.8829$E+03$
	Std	5.5701$E+02$	2.1804$E+02$	2.0380$E+03$	5.2205$E+02$	1.2671$E+03$
F10	Mean	4.5699$E+03$	4.5175$E+03$	5.0778$E+03$	5.2580$E+03$	4.5080$E+03$
	Std	6.1731$E+02$	5.5524$E+02$	7.8534$E+02$	7.1589$E+02$	5.3022$E+02$
F11	Mean	7.7552$E+03$	6.6105$E+03$	1.3689$E+03$	1.4891$E+03$	1.3598$E+03$
	Std	5.4998$E+03$	3.1270$E+03$	2.4036$E+02$	1.2940$E+02$	1.1504$E+02$
F12	Mean	7.2515$E+06$	2.1051$E+06$	1.2461$E+06$	1.4148$E+06$	9.1516$E+05$
	Std	4.4098$E+06$	1.1815$E+06$	4.9808$E+05$	1.0915$E+06$	4.0362$E+05$
F13	Mean	6.5786$E+05$	3.3389$E+05$	1.4912$E+04$	2.3485$E+04$	1.3990$E+04$
	Std	2.7134$E+05$	1.9016$E+05$	3.4013$E+03$	2.2300$E+04$	1.0098$E+03$
F14	Mean	4.1154$E+06$	1.6399$E+06$	4.8582$E+05$	2.7522$E+05$	2.2566$E+05$
	Std	1.6526$E+06$	1.5944$E+06$	2.9789$E+05$	9.3017$E+04$	1.9777$E+05$
F15	Mean	4.2048$E+05$	3.8589$E+05$	6.6368$E+03$	1.3934$E+04$	1.0980$E+04$
	Std	1.9513$E+05$	1.7599$E+05$	1.3747$E+03$	1.2331$E+04$	1.0807$E+04$
F16	Mean	2.9285$E+03$	2.8481$E+03$	3.1832$E+03$	3.3054$E+03$	2.6314$E+03$
	Std	2.5813$E+02$	2.6539$E+02$	2.8659$E+02$	2.9366$E+02$	1.9852$E+02$
F17	Mean	2.4803$E+03$	2.6151$E+03$	2.8180$E+03$	2.6563$E+03$	2.4369$E+03$
	Std	2.4101$E+02$	1.5216$E+02$	2.9776$E+02$	3.3716$E+02$	8.0657$E+01$
F18	Mean	2.9469$E+06$	1.4557$E+06$	1.8334$E+06$	1.9069$E+06$	1.3240$E+06$
	Std	2.2382$E+06$	6.2132$E+05$	2.7873$E+05$	8.6939$E+05$	1.6639$E+05$
F19	Mean	1.2565$E+05$	1.5159$E+05$	2.1308$E+04$	1.8654$E+04$	1.0586$E+04$
	Std	4.5501$E+04$	1.4916$E+05$	1.6375$E+04$	2.0967$E+04$	8.7381$E+03$
F20	Mean	2.5248$E+03$	2.6783$E+03$	2.9207$E+03$	2.7799$E+03$	2.4961$E+03$
	Std	1.4347 $E+02$	2.2008$E+02$	4.1438$E+02$	1.6015$E+02$	1.3643$E+02$
F21	Mean	2.4007$E+03$	2.4017$E+03$	2.5215$E+03$	2.5316$E+03$	2.4833$E+03$
	Std	8.6829$E+00$	2.7428$E+01$	5.0754$E+01$	1.2309$E+01$	2.3791$E+01$
F22	Mean	5.0347$E+03$	5.5745$E+03$	6.8846$E+03$	7.2661$E+03$	5.0281$E+03$
	Std	1.3926$E+03$	1.4422$E+03$	6.5978$E+02$	2.2049$E+03$	1.9036$E+03$
F23	Mean	2.7858$E+03$	2.8232$E+03$	3.0110$E+03$	4.3402$E+03$	2.7844$E+03$
	Std	1.9115$E+01$	3.4769$E+01$	8.6116$E+01$	1.6425$E+02$	1.5676$E+01$
F24	Mean	2.9992$E+03$	3.1099$E+03$	3.2308$E+03$	3.4116$E+03$	2.9982$E+03$
	Std	3.9877$E+01$	3.7526$E+01$	7.1558$E+01$	2.7861$E+02$	2.7076$E+01$
F25	Mean	2.9042$E+03$	2.9020$E+03$	2.8969$E+03$	2.9352$E+03$	2.8899$E+03$
	Std	1.8822$E+01$	1.6433$E+01$	1.1056$E+01$	3.5758$E+01$	7.6811$E+00$
F26	Mean	5.6751$E+03$	4.8496$E+03$	9.8539$E+03$	4.6828$E+03$	5.5333$E+03$
	Std	1.2513$E+02$	2.4925$E+02$	2.5407$E+02$	2.8345$E+02$	6.9512$E+01$
F27	Mean	3.2466$E+03$	3.2824$E+03$	3.3265$E+03$	3.4259$E+03$	3.2456$E+03$
	Std	1.5488$E+01$	2.0294$E+01$	5.8324$E+01$	1.3208$E+02$	1.2590$E+01$
F28	Mean	3.2650$E+03$	3.2181$E+03$	3.2160$E+03$	3.2258$E+03$	3.2129$E+03$
	Std	2.3417$E+01$	1.6696$E+01$	2.6681$E+01$	1.1103$E+03$	1.1220$E+01$
F29	Mean	3.9047$E+03$	3.9269$E+03$	4.9176$E+03$	4.4414$E+03$	3.9039$E+03$
	Std	1.2652$E+02$	1.5749$E+02$	1.8912$E+02$	2.6653$E+02$	1.1682$E+02$
F30	Mean	8.6592$E+04$	2.3951$E+04$	1.8992$E+04$	1.5963$E+04$	1.1207$E+04$
	Std	5.2602$E+04$	1.7045$E+04$	1.2445$E+04$	3.4998$E+03$	3.3715$E+03$

Table 7: Comparison results on CEC 2017 benchmark functions ($D = 50$).

Function	Value	BBO	B-BBO	BBO-M	TDBBO	DCGBBO
F1	Mean	4.5534$E+06$	8.5285$E+03$	1.2450$E+04$	7.1327$E+03$	6.6860$E+03$
	Std	8.2775$E+05$	4.6143$E+03$	8.6411$E+03$	5.6290$E+03$	3.5733$E+03$
F2	Mean	4.9564$E+31$	3.6130$E+20$	8.3500$E+36$	6.8599$E+22$	1.3255$E+17$
	Std	7.0092$E+31$	4.0180$E+18$	1.1792$E+37$	9.6981$E+22$	1.2605$E+17$
F3	Mean	2.4582$E+05$	7.7333$E+04$	1.4055$E+05$	6.9552$E+04$	1.3164$E+05$
	Std	3.3805$E+04$	3.6960$E+04$	4.5305$E+04$	3.8235$E+04$	2.9520$E+04$
F4	Mean	5.6291$E+02$	5.5274$E+02$	5.6517$E+02$	5.3337$E+02$	5.0774$E+02$
	Std	4.9271$E+01$	6.0366$E+01$	4.7069$E+01$	7.1149$E+01$	4.5645$E+01$
F5	Mean	7.1753$E+02$	7.0801$E+02$	9.5435$E+02$	8.7642$E+02$	8.4335$E+02$
	Std	1.9408$E+01$	3.4002$E+01$	1.1771$E+02$	8.1464$E+01$	5.4256$E+01$
F6	Mean	6.2677$E+02$	6.0007$E+02$	6.7606$E+02$	6.5970$E+02$	6.2564$E+02$
	Std	7.8823$E-02$	4.6893$E-02$	3.9206$E+00$	8.2295$E+00$	3.8631$E+00$
F7	Mean	1.0088$E+03$	9.8735$E+02$	3.5893$E+03$	2.0899$E+03$	1.0749$E+03$
	Std	3.0508$E+01$	2.6606$E+01$	2.7245$E+02$	2.0947$E+02$	6.1975$E+01$
F8	Mean	9.6919$E+02$	9.8924$E+02$	1.3265$E+03$	1.1628$E+03$	1.1804$E+03$
	Std	5.1574$E+01$	3.1845$E+01$	1.5945$E+02$	1.3265$E+03$	3.0023$E+01$
F9	Mean	4.1603$E+03$	4.4364$E+03$	2.3369$E+04$	1.2206$E+04$	1.7392$E+04$
	Std	2.6953$E+03$	1.9429$E+03$	3.5611$E+03$	3.8837$E+03$	1.9011$E+03$
F10	Mean	6.7747$E+03$	7.0697$E+03$	8.7494$E+03$	8.9914$E+03$	6.6757$E+03$
	Std	7.4257$E+02$	8.5307$E+02$	6.5265$E+02$	6.7515$E+02$	6.1082$E+02$
F11	Mean	1.1240$E+04$	6.3925$E+03$	1.4738$E+03$	1.4670$E+03$	1.4369$E+03$
	Std	6.3868$E+03$	3.4914$E+03$	2.7387$E+01$	1.3309$E+02$	2.6880$E+01$
F12	Mean	3.4829$E+07$	5.5871$E+06$	5.4082$E+06$	5.4123$E+06$	3.7717$E+06$
	Std	1.1321$E+07$	1.9812$E+06$	2.0075$E+06$	3.3981$E+06$	1.9140$E+06$
F13	Mean	1.5819$E+06$	3.6046$E+05$	1.6165$E+04$	1.5964$E+04$	1.0071$E+04$
	Std	7.2597$E+05$	3.4239$E+05$	5.8611$E+03$	6.1982$E+03$	5.2622$E+03$
F14	Mean	7.2961$E+06$	1.0222$E+07$	3.6465$E+05$	4.1988$E+05$	3.1091$E+05$
	Std	2.7307$E+05$	2.6795$E+06$	1.5113$E+05$	2.3230$E+05$	1.0629$E+05$
F15	Mean	1.7722$E+06$	1.5313$E+06$	1.2537$E+04$	1.6464$E+04$	9.9267$E+03$
	Std	9.3174$E+05$	3.8290$E+05$	7.4936$E+03$	7.9128$E+03$	7.2030$E+03$
F16	Mean	3.5805$E+03$	3.6935$E+03$	4.0112$E+03$	3.5772$E+03$	3.4679$E+03$
	Std	2.8490$E+02$	5.3918$E+02$	5.2281$E+02$	2.3058$E+02$	1.8773$E+02$
F17	Mean	3.6204$E+03$	3.4643$E+03$	4.3956$E+03$	3.8489$E+03$	3.4381$E+03$
	Std	5.3340$E+02$	3.5515$E+02$	7.0775$E+02$	4.8888$E+02$	3.2285$E+02$
F18	Mean	1.7178$E+07$	6.7179$E+06$	2.0530$E+06$	1.7286$E+06$	7.8350$E+05$
	Std	1.0903$E+07$	1.2929$E+06$	1.0905$E+06$	1.2087$E+06$	4.6721$E+05$
F19	Mean	2.6680$E+05$	3.7502$E+04$	1.7302$E+04$	2.2410$E+04$	1.1665$E+04$
	Std	1.3560$E+05$	8.7685$E+03$	1.2369$E+04$	1.3954$E+04$	5.5641$E+03$
F20	Mean	3.3597$E+03$	3.3762$E+03$	3.7006$E+03$	3.6179$E+03$	3.3449$E+03$
	Std	3.1046$E+02$	4.0777$E+02$	1.1085$E+02$	3.1626$E+02$	1.8588$E+02$
F21	Mean	2.7789$E+03$	2.3100$E+03$	2.8415$E+03$	2.7329$E+03$	2.3595$E+03$
	Std	6.4204$E+01$	1.5609$E+01$	5.4735$E+01$	5.1492$E+01$	4.7061$E+01$
F22	Mean	8.8223$E+03$	8.4081$E+03$	1.0196$E+04$	1.0986$E+04$	7.3650$E+03$
	Std	1.4081$E+03$	9.9405$E+02$	3.6858$E+02$	5.7951$E+02$	6.9539$E+02$
F23	Mean	3.2448$E+03$	3.0430$E+03$	3.7024$E+03$	3.4871$E+03$	3.0420$E+03$
	Std	6.1704$E+01$	8.2445$E+01$	2.1648$E+02$	1.1224$E+02$	5.1718$E+01$
F24	Mean	3.3788$E+03$	3.4315$E+03$	3.6068$E+03$	3.6100$E+03$	3.3103$E+03$
	Std	2.8741$E+01$	5.3317$E+01$	1.2678$E+02$	1.9000$E+02$	9.1731$E+00$
F25	Mean	3.0812$E+03$	3.0954$E+03$	3.0824$E+03$	3.0764$E+03$	3.0112$E+03$
	Std	3.2545$E+01$	4.1084$E+00$	1.0947$E+01$	9.5134$E+00$	3.6953$E+00$
F26	Mean	7.3849$E+03$	8.1513$E+03$	1.4529$E+04$	1.1593$E+04$	7.3758$E+03$
	Std	1.7275$E+02$	7.6898$E+02$	7.3687$E+02$	2.3545$E+03$	1.6739$E+02$
F27	Mean	3.6131$E+03$	3.7026$E+03$	4.4800$E+03$	4.2449$E+03$	3.5982$E+03$
	Std	9.1037$E+02$	1.2647$E+02$	5.9388$E+02$	1.3665$E+02$	1.1960$E+02$
F28	Mean	3.3357$E+03$	3.3334$E+03$	3.3443$E+03$	3.3186$E+03$	3.3083$E+03$
	Std	1.6452$E+01$	8.1412$E+00$	1.4603$E+01$	9.9218$E+00$	7.8384$E-01$
F29	Mean	4.3156$E+03$	4.2839$E+03$	6.1147$E+03$	6.0862$E+03$	4.1902$E+03$
	Std	4.9479$E+02$	3.7419$E+02$	4.4033$E+02$	5.4248$E+02$	3.3537$E+02$
F30	Mean	1.9725$E+06$	1.3692$E+06$	4.6563$E+06$	2.1541$E+06$	1.2843$E+06$
	Std	3.6799$E+05$	7.9265$E+05$	9.4001$E+05$	9.1187$E+05$	2.5702$E+05$

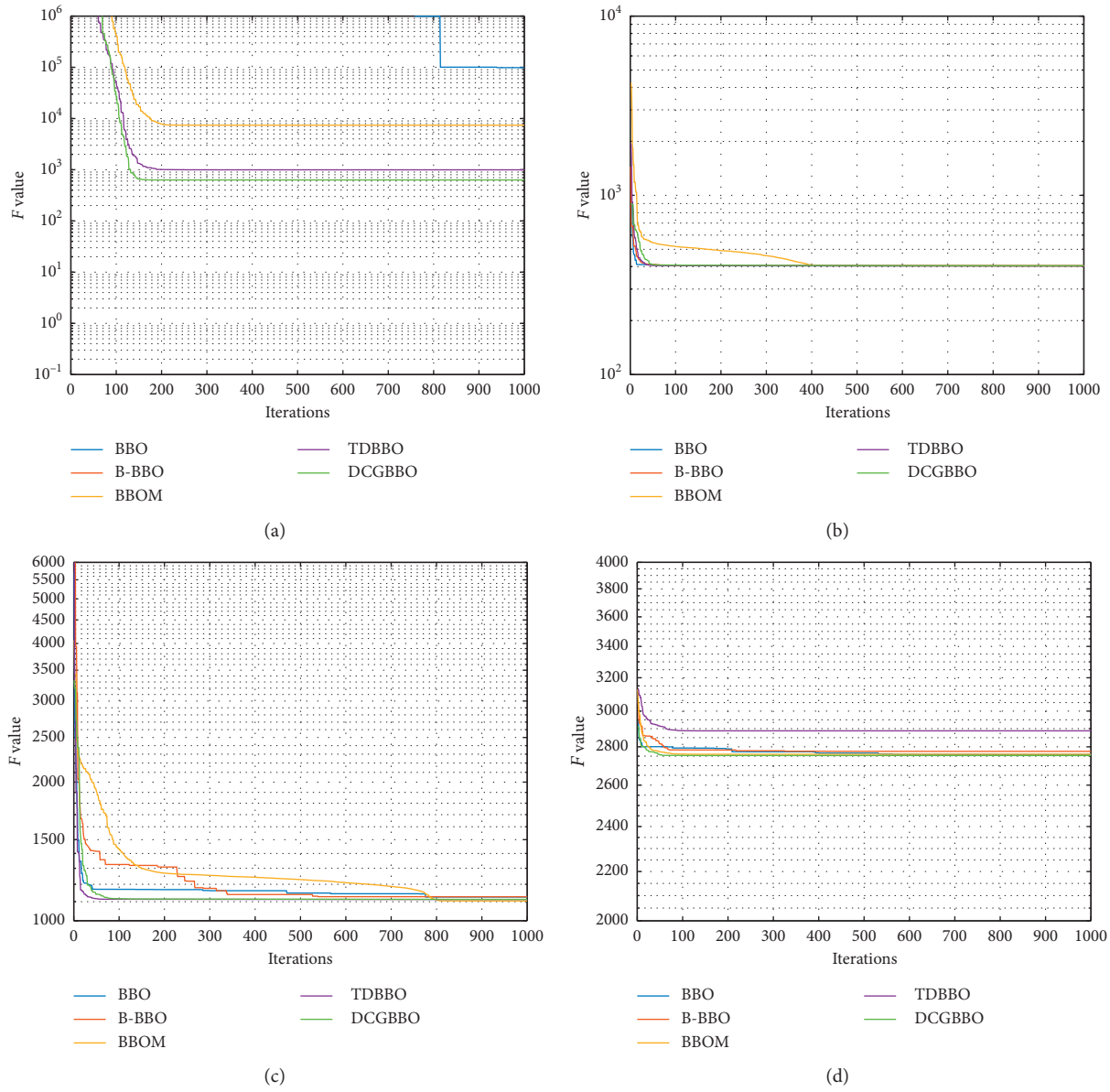

Figure 4: Convergence curves of 5 algorithms on 10D CEC 2017 benchmark functions. (a) F1. (b) F4. (c) F11. (d) F24.

benchmark problems, respectively. It can be seen from Figures 4–6 that, among the most advanced methods compared in the experiment, DCGBBO's exploration ability is very good at the beginning. The algorithm converges fast and in a consistent direction and finally reaches the optimal solution. The results show that the proposed DCGBBO can balance the local search and global search.

4.2.3. Experiment Results and Analysis on CEC 2017 Benchmark Functions with respect to State-of-the-Art Algorithms. In order to have a fair comparison of the proposed SACS with respect to state-of-the-art algorithms, basic CS [42], basic BBO, and CV1.0 [43] algorithms are used. These algorithms are highly competitive and have proven their value in various CEC competitions and solving other real-world optimization problems. The results for CS and CV1.0 are taken from [44]. The population is set to 50 ($N = 50$). The stopping criterion was taken as $10{,}000 \times D$ total number of function evaluations with 51 runs performed for each test problem. The error values are calculated by finding the difference between the expected and the desired solution, and if the difference becomes less than 10^{-8}, the error is considered as zero. The comparison results are shown in Table 8. The last row of the table gives the values of the Wilcoxon rank-sum test [45]. Here, "+ (win/w)" represents the algorithm under consideration is better than the proposed algorithm, "- (lost/l)" corresponds to the situation that algorithm under test is worse as compared to DCGBBO, and "= (tie/t)" represents that they either have no correlation, or they have the same statistical results and are equal to each other.

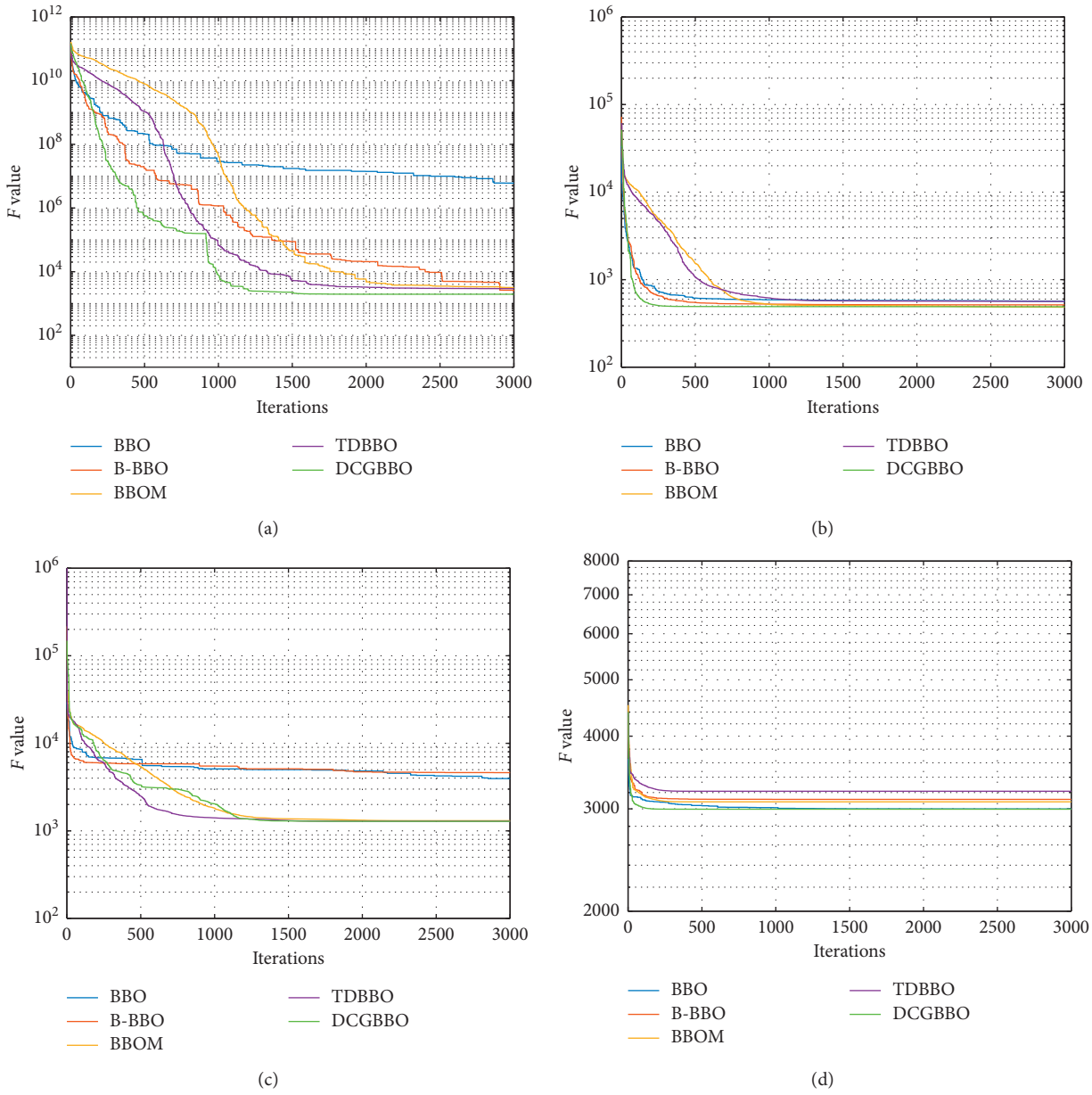

Figure 5: Convergence curves of 5 algorithms on 30D CEC 2017 benchmark functions. (a) F1. (b) F4. (c) F11. (d) F24.

From Table 8, we can obtain that, for unimodal functions (F1–F3), CV1.0 is highly competitive and attain good results as CS and DCGBBO were not able to converge. For most of the multimodal functions (F4–F10), DCGBBO performs better than basic BBO and basic CS. For the hybrid functions (F11–F20), the CV1.0 algorithm has strong competitiveness, but among them the newly proposed DCGBBO is the best performing algorithm. For the final set of composite functions (F21–F30), the DCGBBO algorithm was again the best performing algorithm among all the algorithms under test. From the results of the last row of Table 8, we can say that the proposed algorithm is highly competitive and future modification in the same approach may lead to much better results.

4.2.4. Parametric Analysis. In order to test the individual effect of the proposed dynamic crossover migration operator, dynamic Gaussian mutation operator, and a unified maximum mutation rate $m_{\max}$ on the performance of DCGBBO, the experiments of the same DCGBBO version without the above improvements were carried out separately. The same DCGBBO version without the proposed dynamic crossover migration operator is called DCGBBO-1, the same DCGBBO version without the proposed dynamic Gaussian mutation operator is called DCGBBO-2, and the same DCGBBO version without the proposed unified maximum mutation rate $m_{\max}$ is called DCGBBO-3. Table 8 shows the experimental results of each version on CEC 2017 benchmark functions with dimension 30, the number of independent runs (Num) of all the algorithms is 51, and the

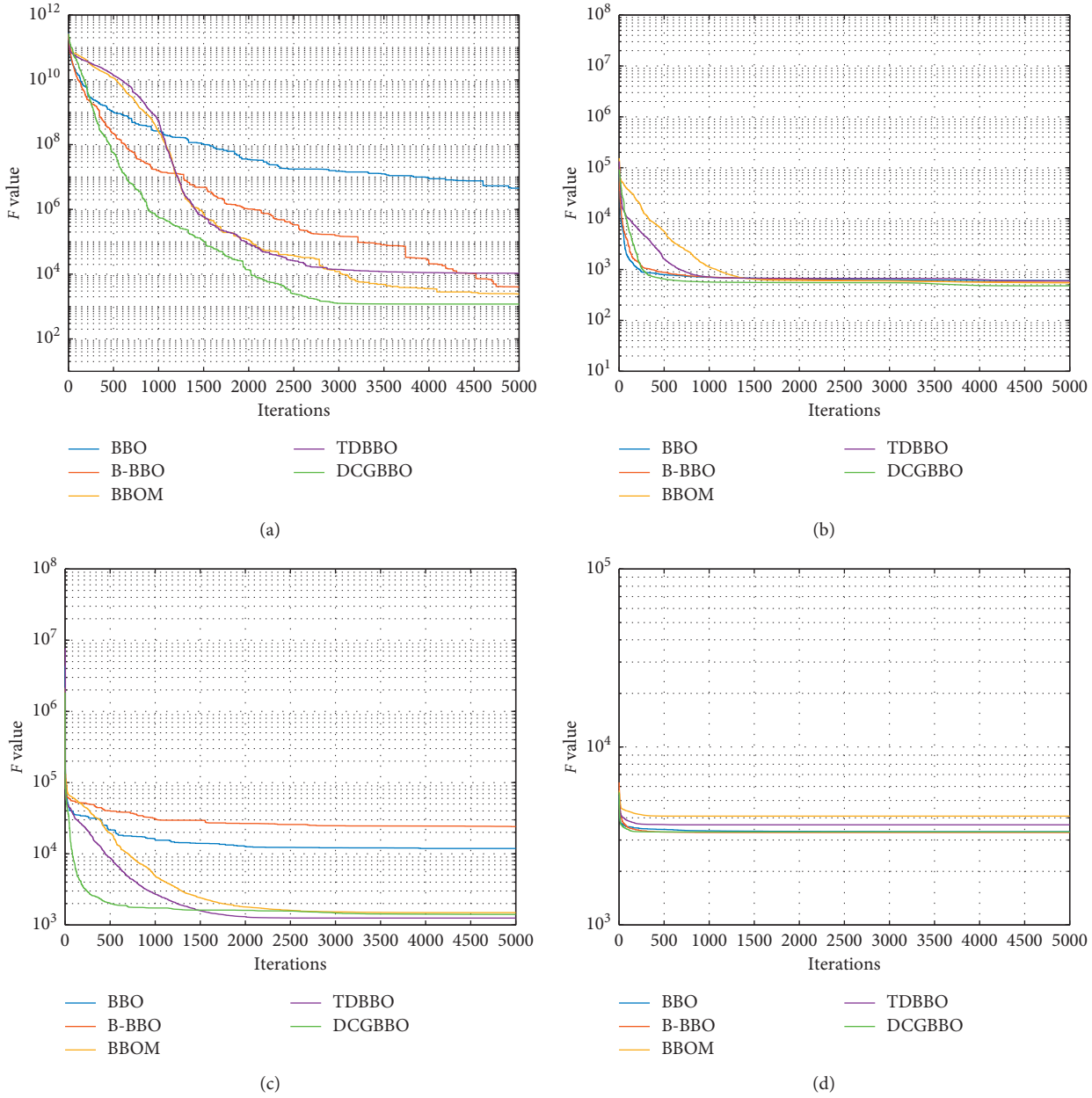

Figure 6: Convergence curves of 5 algorithms on 50D CEC 2017 benchmark functions. (a) F1. (b) F4. (c) F11. (d) F24.

population size is 100 ($N = 100$). Maximum number of function evaluation (MNFE) is 300000 for $D = 30$. The last row of the table gives the values of the Wilcoxon rank-sum test.

From Table 9, we can conclude that there are 25 functions in the CEC 2017 benchmark functions that can reflect the advantages of dynamic crossover migration, which can well increase the global search ability, and the population diversity will also increase. Similarly, there are 23 functions that can show the optimization performance of dynamic Gaussian mutation, which can greatly improve the local search ability of the algorithm, expand the search range of the solution space under the premise of finding the local optimal solution, and accelerate the convergence speed of the algorithm. In addition, there are 29 functions because of the fixed mutation probability $m_{\max}$, the overall optimization effect of the algorithm has been greatly improved. Therefore, each improvement can improve the optimization performance of the algorithm.

4.2.5. Friedman Test. To investigate the performance of DCGBBO, the Friedman test [45] was used for testing statistically the performance of DCGBBO compared with the comparison algorithms. The Friedman hypothesis test is a nonparametric test using rank to implement significant differences in multiple population distributions. Null hypothesis: all the comparison algorithms in the experiment

Table 8: Comparison results between DCGBBO and other algorithms ($D = 50$).

Functions	Value	CS	CV1.0	BBO	DCGBBO
	Mean	$1.00E+10$	$1.00E+10$	$3.57E+06$	$6.99E+03$
F1	Std	$0.00E+00$	$0.00E+00$	$6.50E+05$	$7.56E+03$
	Rank	—	—	—	
	Mean	—	$1.00E+10$	$5.64E+31$	$1.35E+17$
F2	Std	—	$0.00E+00$	$8.41E+30$	$4.06E+17$
	Rank	—	+	—	
	Mean	$2.52E+05$	$1.95E+04$	$2.43E+05$	$1.01E+05$
F3	Std	$3.02E+04$	$6.27E+03$	$3.55E+04$	$1.94E+04$
	Rank	-	+	—	
	Mean	$1.28E+02$	$1.16E+02$	$1.71E+02$	$1.11E+02$
F4	Std	$2.44E+01$	$6.27E+03$	$4.42E+01$	$2.36E+01$
	Rank	—	—	—	
	Mean	$4.86E+02$	$3.41E+02$	$2.68E+02$	$3.40\,E+02$
F5	Std	$4.66E+01$	$8.02E+01$	$5.04E+01$	$3.43E+01$
	Rank	—	—	+	
	Mean	$4.13E+01$	$4.85E+01$	$4.05E+01$	$3.56E+01$
F6	Std	$6.32E+00$	$4.85E+01$	$8.12E+00$	$4.66E+00$
	Rank	—	—	—	
	Mean	$5.51E+02$	$2.74E+02$	$3.09E+02$	$4.05E+02$
F7	Std	$4.08E+01$	$7.29E+01$	$4.05E+01$	$3.80E+01$
	Rank	—	+	+	
	Mean	$4.82E+02$	$3.89E+02$	$2.69E+02$	$3.80E+02$
F8	Std	$4.67E+01$	$7.29E+01$	$6.76E+01$	$4.89E+02$
	Rank	—	—	+	
	Mean	$3.53E+04$	$1.00E+04$	$4.42E+03$	$1.78E+04$
F9	Std	$4.82E+03$	$2.90E+03$	$2.54E+03$	$2.90E+03$
	Rank	—	+	+	
	Mean	$7.39E+03$	$7.10E+03$	$6.01E+03$	$5.88E+03$
F10	Std	$3.26E+02$	$5.34E+02$	$6.92E+02$	$3.11E+02$
	Rank	—	—	—	
	Mean	$3.45E+02$	$1.66E+02$	$1.00E+04$	$2.37E+02$
F11	Std	$4.16E+01$	$3.38E+01$	$6.19E+03$	$2.63E+01$
	Rank	—	+	—	
	Mean	$1.00E+10$	$1.00E+10$	$2.59E+07$	$3.84E+06$
F12	Std	$0.00E+00$	$0.00E+00$	$9.32E+06$	$2.93E+06$
	Rank	—	—	—	
	Mean	$1.00E+10$	$1.00E+10$	$4.02E+06$	$9.77E+03$
F13	Std	$0.00E+00$	$0.00E+00$	$3.14E+06$	$3.26E+03$
	Rank	—	—	—	
	Mean	$3.26E+05$	$2.05E+02$	$7.12E+06$	$3.00E+05$
F14	Std	$1.60E+05$	$2.13E+01$	$4.33E+05$	$2.50E+05$
	Rank	—	+	—	
	Mean	$7.85E+09$	$1.37E+09$	$2.47E+06$	$8.43E+03$
F15	Std	$4.12E+09$	$3.47E+09$	$4.39E+05$	$8.50E+03$
	Rank	—	—	—	
	Mean	$1.76E+03$	$1.53E+03$	$1.98E+03$	$1.44E+03$
F16	Std	$2.37E+02$	$2.74E+02$	$2.49E+02$	$1.90E+02$
	Rank	—	—	—	
	Mean	$1.18E+03$	$1.25E+03$	$2.02E+03$	$1.74E+03$
F17	Std	$1.78E+02$	$1.85E+02$	$4.43E+02$	$1.23E+02$
	Rank	+	+	—	
	Mean	$1.43E+06$	$5.21E+02$	$6.22E+07$	$7.81E+05$
F18	Std	$5.89E+05$	$1.19E+02$	$6.89E+06$	$3.57E+05$
	Rank	—	+	—	
	Mean	$1.99E+08$	$1.73E+02$	$2.01E+05$	$9.77E+03$
F19	Std	$1.39E+09$	$4.17E+02$	$3.66E+04$	$6.56E+03$
	Rank	—	+	—	
	Mean	$1.04E+03$	$1.05E+03$	$1.46E+03$	$1.24E+03$
F20	Std	$1.67E+02$	$2.14E+02$	$3.77E+02$	$1.66E+02$
	Rank	+	+	—	

Table 8: Continued.

Functions	Value	CS	CV1.0	BBO	DCGBBO
	Mean	$6.55E+02$	$5.41E+02$	$7.09E+02$	$2.63E+02$
F21	Std	$7.93E+01$	$6.27E+01$	$5.72E+01$	$4.91E+01$
	Rank	—	—	—	
	Mean	$8.19E+03$	$7.33E+03$	$6.62E+03$	$6.17E+03$
F22	Std	$4.08E+02$	$1.99E+03$	$1.01E+03$	$6.78E+02$
	Rank	—	—	—	
	Mean	$9.14E+02$	$7.74E+02$	$8.45E+02$	$7.42E+02$
F23	Std	$4.59E+01$	$8.06E+01$	$7.50E+01$	$4.17E+01$
	Rank	—	—	—	
	Mean	$1.01E+03$	$8.32E+02$	$9.79E+02$	$9.10E+02$
F24	Std	$6.38E+01$	$1.21E+01$	$1.87E+01$	$1.02E+01$
	Rank	—	+	—	
	Mean	$5.33E+02$	$5.43E+02$	$6.01E+02$	$5.21E+02$
F25	Std	$1.66E+01$	$1.51E+01$	$4.21E+01$	$3.70E+00$
	Rank	—	—	—	
	Mean	$4.57E+03$	$2.48E+03$	$4.78E+03$	$4.68E+03$
F26	Std	$1.82E+03$	$1.88E+03$	$2.33E+02$	$1.05E+02$
	Rank	+	+	—	
	Mean	$8.17E+02$	$7.38E+02$	$9.03E+02$	$8.90E+02$
F27	Std	$5.68E+01$	$8.21E+01$	$8.50E+02$	$1.96E+01$
	Rank	+	+	—	
	Mean	$5.12E+02$	$4.94E+02$	$5.16E+02$	$5.03E+02$
F28	Std	$1.88E+01$	$1.93E+01$	$2.45E+01$	$6.24E+00$
	Rank	—	+	—	
	Mean	$1.57E+03$	$1.69E+03$	$1.49E+03$	$1.32E+03$
F29	Std	$1.79E+02$	$2.29E+02$	$3.95E+02$	$1.35E+02$
	Rank	—	—	—	
	Mean	$2.95E+09$	$4.64E+06$	$2.25E+06$	$1.88E+06$
F30	Std	$4.59E+09$	$8.59E+06$	$5.72E+05$	$2.53E+05$
	Rank	—	—	—	
w/l/t		4/25/0	14/16/0	4/26/0	

Table 9: Comparison results ($D = 30$).

Function	Value	DCGBBO-1	DCGBBO-2	DCGBBO-3	DCGBBO
	Mean	$1.1356E+04$	$4.7084E+03$	$5.9709E+03$	$3.4986E+03$
F1	Std	$6.2167E+03$	$1.2510E+03$	$1.1553E+03$	$4.1129E+03$
	Rank	—	—	—	
	Mean	$8.3672E+12$	$2.8681E+13$	$2.4459E+12$	$1.4756E+10$
F2	Std	$1.1302E+13$	$2.1715E+13$	$3.4458E+12$	$1.7861E+11$
	Rank	—	—	—	
	Mean	$1.5852E+05$	$1.5282E+05$	$8.5642E+04$	$6.2097E+03$
F3	Std	$8.5953E+04$	$7.0895E+04$	$3.0558E+04$	$2.6195E+03$
	Rank	—	—	—	
	Mean	$5.4851E+02$	$4.8680E+02$	$4.8717E+02$	$4.7975E+02$
F4	Std	$3.4931E+01$	$1.8757E+01$	$1.7507E+00$	$8.6073E+00$
	Rank	—	—	—	
	Mean	$6.4970E+02$	$6.1790E+02$	$6.8210E+02$	$6.7660E+02$
F5	Std	$1.8458E+01$	$2.6254E+01$	$9.4053E+00$	$1.6995E+01$
	Rank	+	+	—	
	Mean	$6.2000E+02$	$6.0111E+02$	$6.2498E+02$	$6.0079E+02$
F6	Std	$1.4027E+01$	$4.4846E-03$	$1.5729E+01$	$4.9564E+00$
	Rank	—	—	—	
	Mean	$1.2476E+03$	$8.7160E+02$	$9.9638E+02$	$9.3643E+02$
F7	Std	$1.4698E+02$	$1.6442E+01$	$2.8770E+01$	$4.6188E+01$
	Rank	—	+	—	
	Mean	$9.6384E+02$	$9.2333E+02$	$1.0416E+03$	$9.9107E+02$
F8	Std	$3.4200E+01$	$3.4892E+01$	$4.9989E+01$	$6.1861E+01$
	Rank	+	+	—	

Table 9: Continued.

Function	Value	DCGBBO-1	DCGBBO-2	DCGBBO-3	DCGBBO
	Mean	4.5104$E+03$	3.0913$E+03$	6.5061$E+03$	5.8829$E+03$
F9	Std	1.7918$E+03$	2.3509$E+02$	2.8977$E+03$	1.2671$E+03$
	Rank	+	+	—	
	Mean	6.0419$E+03$	4.2296$E+03$	5.8954$E+03$	4.5080$E+03$
F10	Std	4.8381$E+02$	3.9345$E+02$	1.2187$E+03$	5.3022$E+02$
	Rank	—	+	—	
	Mean	1.3765$E+03$	6.9547$E+03$	1.3162$E+03$	1.3598$E+03$
F11	Std	1.4102$E+02$	2.2973$E+03$	6.3358$E+01$	1.1504$E+02$
	Rank	—	—	+	
	Mean	1.4078$E+06$	1.1135$E+06$	1.4175$E+06$	9.1516$E+05$
F12	Std	4.6185$E+05$	7.8204$E+05$	5.9977$E+05$	4.0362$E+05$
	Rank	—	—	—	
	Mean	3.4200$E+04$	2.1576$E+05$	3.3701$E+04$	1.3990$E+04$
F13	Std	2.4819$E+04$	7.8291$E+04$	2.6332$E+04$	1.0098$E+03$
	Rank	—	—	—	
	Mean	2.4390$E+06$	4.5935$E+06$	2.5577$E+05$	2.2566$E+05$
F14	Std	1.6254$E+06$	3.9305$E+06$	1.0537$E+05$	1.9777$E+05$
	Rank	—	—	—	
	Mean	2.2555$E+04$	8.2162$E+04$	1.3483$E+04$	1.0980$E+04$
F15	Std	1.3587$E+04$	5.7354$E+04$	9.5731$E+03$	1.0807$E+04$
	Rank	—	—	—	
	Mean	3.2154$E+03$	2.6706$E+03$	3.1079$E+03$	2.6314$E+03$
F16	Std	4.8213$E+02$	2.3915$E+02$	4.9006$E+02$	1.9852$E+02$
	Rank	—	—	—	
	Mean	2.4487$E+03$	2.4587$E+03$	2.7369$E+03$	2.4369$E+03$
F17	Std	1.5107$E+02$	1.8634$E+02$	4.9742$E+01$	8.0657$E+01$
	Rank	—	—	—	
	Mean	1.7959$E+06$	6.9817$E+06$	2.4653$E+06$	1.3240$E+06$
F18	Std	1.5266$E+06$	4.5528$E+06$	1.7262$E+06$	1.6639$E+05$
	Rank	—	—	—	
	Mean	8.1250$E+03$	3.0583$E+04$	3.3237$E+04$	1.0586$E+04$
F19	Std	2.0035$E+03$	2.8386$E+04$	1.5454$E+04$	8.7381$E+03$
	Rank	+	—	—	
	Mean	2.6911$E+03$	2.6922$E+03$	3.1552$E+03$	2.4961$E+03$
F20	Std	1.7123$E+02$	5.1005$E+01$	2.1504$E+02$	1.3643$E+02$
	Rank	—	—	—	
	Mean	2.4305$E+03$	2.4101$E+03$	2.4939$E+03$	2.4833$E+03$
F21	Std	2.6785$E+01$	2.8179$E+01$	5.4148$E+01$	2.3791$E+01$
	Rank	+	+	—	
	Mean	6.4604$E+03$	5.9365$E+03$	6.4819$E+03$	5.0281$E+03$
F22	Std	5.0618$E+02$	1.7148$E+02$	9.7005$E+02$	9.0360$E+02$
	Rank	—	—	—	
	Mean	2.8965$E+03$	2.7928$E+03$	2.8558$E+03$	2.7844$E+03$
F23	Std	3.6426$E+01$	1.2264$E+01$	1.6936$E+01$	1.5676$E+01$
	Rank	—	—	—	
	Mean	3.0105$E+03$	3.0307$E+03$	3.0209$E+03$	2.9982$E+03$
F24	Std	2.5921$E+01$	8.0940$E+01$	3.4606$E+01$	2.7076$E+01$
	Rank	—	—	—	
	Mean	2.8960$E+03$	2.9172$E+03$	2.9140$E+03$	2.8899$E+03$
F25	Std	1.1293$E+01$	1.4694$E+01$	4.0064$E+01$	7.6811$E+00$
	Rank	—	—	—	
	Mean	5.7904$E+03$	5.1905$E+03$	6.9282 $E+03$	5.5333$E+03$
F26	Std	7.9146$E+02$	3.6037$E+02$	7.1928$E+02$	6.9512$E+01$
	Rank	—	+		
	Mean	3.3503$E+03$	3.2459$E+03$	3.2850$E+03$	3.2456$E+03$
F27	Std	7.0498$E+01$	1.4053$E+01$	2.0881$E+01$	1.2590$E+01$
	Rank	—	—	—	
	Mean	3.2178$E+03$	3.2866$E+03$	3.2382$E+03$	3.2129$E+03$
F28	Std	1.3027$E+01$	6.9604$E+01$	3.5481$E+01$	1.1220$E+01$
	Rank	—	—	—	

Table 9: Continued.

Function	Value	DCGBBO-1	DCGBBO-2	DCGBBO-3	DCGBBO
	Mean	$4.2793E+03$	$3.9474E+03$	$4.2701E+03$	$3.9039E+03$
F29	Std	$2.2204E+02$	$1.7223E+02$	$3.1895E+02$	$1.1682E+02$
	Rank	—	—	—	
	Mean	$4.1858E+04$	$1.6967E+04$	$1.5831E+04$	$1.1207E+04$
F30	Std	$4.3777E+04$	$6.6238E+03$	$3.3964E+03$	$3.3715E+03$
	Rank	—	—	—	
w/l/t		5/25/0	7/23/0	1/29/0	

have the same performance for any benchmark function. In the Friedman test, the 8 benchmark functions are treated as a random sample and each optimization algorithm is considered as a treatment. The mean values (Mean) of the optimization approaches on each benchmark function are ranked from the largest to the smallest [46]. The ranking of the j comparison algorithm on the i test function $f(x)$ can be expressed as $r_{i,j}$, and the Friedman test statistic χ_r^2 is structured as follows:

$$\chi_r^2 = \frac{12}{np(p+1)} \sum_{p=1}^{p} \left(\sum_{i=1}^{n} r_{i,j} \right)^2 - 3n(p+1), \tag{14}$$

where n is the number of test benchmark functions and p is the number of comparison algorithms. The Friedman test statistic follows a chi-squared distribution with $p-1$ degrees of freedom.

The 5 optimization algorithms in this paper are tested on 8 benchmark functions and obtained the ranks of the mean values (Mean), as shown in Tables 10 and 11. As for Table 10, the Friedman test statistic χ_r^2 is computed, and the result is $\chi_r^2 = 20.95$. As for Table 11, the Friedman test statistic χ_r^2 is $\chi_r^2 = 21.25$. With $p-1=4$ degrees of freedom, the critical value is $\chi_r^2 = 9.488$ at the significance level $\alpha = 0.05$. The conclusion of the Friedman test, therefore, is to reject the null hypothesis that the 5 optimization algorithms levels on 8 benchmark functions in different functional dimensions are significantly different. In this case, compared algorithms in this paper get significantly different fitness function values.

4.2.6. Wilcoxon Signed-Rank Test. Wilcoxon signed-rank test is a nonparametric test method [45], and it is used to test whether there are significant differences in the comparison algorithm to test the performance of the algorithm. The p values can be computed by using IBM SPSS Statistics 25.0. Wilcoxon signed-rank test is performed on the 10-dimensional, 30-dimensional, and 50-dimensional CEC 2017 benchmark functions. The data are taken from Tables 6–8, and the results are shown in Tables 12–14. Among them, "R^+" is the sum of ranks for the problems in which DCGBBO outperformed the comparison algorithm, and "R^-" is the sum of ranks for the opposite. When the DCGBBO algorithm and the comparison algorithm achieve the same optimization performance, the corresponding ranks are split evenly to "R^+" and "R^-." "w" represents DCGBBO algorithm wins the comparison algorithm on w functions, and "l" and "t" represent DCGBBO loses the comparison algorithm on l functions and ties on t functions, respectively. From the results, we can concluded whether on the 10-dimensional, the 30-dimensional, or the 50-dimensional functions, the values of p are all less than 0.05, so the proposed DCGBBO algorithm' optimization performance outperforms the comparison algorithms significantly.

Table 10: Ranks of the average values (mean) for the five optimization algorithms ($D=30$).

Function	BBO	B-BBO	BBO-M	TDBBO	DCGBBO
f_1	5	2.5	2.5	2.5	2.5
f_2	3	2	5	4	1
f_3	5	2	4	3	1
f_4	5	2	4	3	1
f_5	5	4	3	1	2
f_6	5	3	2	4	1
f_7	5	2	4	3	1
f_8	5	2	4	3	1
Sum rank	38	19.5	28.5	23.5	10.5
Average rank	4.7500	2.4375	3.5625	2.9375	1.3125

Table 11: Ranks of the average values (mean) for the five optimization algorithms ($D=50$).

Function	BBO	B-BBO	BBO-M	TDBBO	DCGBBO
f_1	5	2.5	2.5	2.5	2.5
f_2	3	1	5	4	2
f_3	5	2	4	3	1
f_4	5	2	4	3	1
f_5	5	4	3	2	1
f_6	5	2	3	4	1
f_7	4	2	5	3	1
f_8	4	2	5	3	1
Sum rank	36	17.5	31.5	24.5	10.5
Average rank	4.5	2.1875	3.9375	3.0625	1.3125

4.2.7. Application of DCGBBO to Image Segmentation. Image segmentation is the technique and process of dividing an image into several specific areas and proposing objects of interest. It is a key step from image processing to image analysis. In this paper, we take the pixel as the basic unit of image segmentation and concentrate on pixel image segmentation. Furthermore, we use the improved DCGBBO algorithm to segment colour images, which is done based on the features and the distances between the pixels of the colour images. We use the clustering optimization algorithm and Euclidean distance to segment the colour images, and the objective function is constructed by the following equation:

Table 12: Wilcoxon signed-rank test results on CEC 2017 benchmark functions ($D = 10$).

Comparison	R^+	R^-	p value	$w/l/t$
DCGBBO versus BBO	389	76	0.001	25/5/0
DCGBBO versus B-BBO	386	79	0.002	24/6/0
DCGBBO versus BBO-M	416	49	0.000	28/2/0
DCGBBO versus TDBBO	438.5	26.5	0.000	29/1/0

Table 13: Wilcoxon signed-rank test results on CEC 2017 benchmark functions ($D = 30$).

Comparison	R^+	R^-	p value	$w/l/t$
DCGBBO versus BBO	388	77	0.001	25/5/0
DCGBBO versus B-BBO	379	86	0.003	23/7/0
DCGBBO versus BBO-M	465	0	0.000	30/0/0
DCGBBO versus TDBBO	445	20	0.000	28/2/0

Table 14: Wilcoxon signed-rank test results on CEC 2017 benchmark functions ($D = 50$).

Comparison	R^+	R^-	p value	$w/l/t$
DCGBBO versus BBO	410	55	0.000	26/4/0
DCGBBO versus B-BBO	375	90	0.003	23/7/0
DCGBBO versus BBO-M	465	0	0.000	30/0/0
DCGBBO versus TDBBO	417	48	0.000	27/3/0

Table 15: Comparison results between DCGBBO and BBO variants on 4 images.

Image	Value	BBO	B-BBO	BBO-M	TDBBO	DCGBBO
Church1	Mean	$6.5949E+00$	$6.5708E+00$	$6.4798E+00$	$6.4792E+00$	$6.4677E+00$
	Std	$9.8742E-02$	$1.8412E-01$	$1.1488E-01$	$1.5759E-01$	$4.1861E-02$
	Maxvalue	$6.8362E+00$	$6.9859E+00$	$6.8159E+00$	$6.9498E+00$	$6.5358E+00$
	Minvalue	$6.4548E+00$	$6.4289E+00$	$6.4162E+00$	$6.4132E+00$	$6.4193E+00$
Church2	Mean	$9.2471E+00$	$9.1824E+00$	$9.0874E+00$	$9.0293E+00$	$9.0345E+00$
	Std	$1.8636E-01$	$1.6661E-01$	$1.2215E-01$	$7.4869E-02$	$6.6043E-02$
	Maxvalue	$9.5303E+00$	$9.5776E+00$	$9.3554E+00$	$9.2172E+00$	$9.1880E+00$
	Minvalue	$9.0416E+00$	$9.0475E+00$	$8.9833E+00$	$8.9773E+00$	$8.9767E+00$
Sunflower	Mean	$9.8809E+00$	$9.8240E+00$	$9.7161E+00$	$9.7447E+00$	$9.7067E+00$
	Std	$2.1451E-01$	$2.3833E-01$	$1.9982E-01$	$3.1698E-01$	$9.2090E-02$
	Maxvalue	$1.0278E+01$	$1.0350E+01$	$1.0264E+01$	$1.0583E+01$	$9.9261E+00$
	Minvalue	$9.6478E+00$	$9.5869E+00$	$9.5674E+00$	$9.5653E+00$	$9.5787E+00$
Bird	Mean	$7.6379E+00$	$7.5532E+00$	$7.5633E+00$	$7.5322E+00$	$7.5317E+00$
	Std	$4.9898E-02$	$4.6186E-02$	$4.8408E-02$	$2.0981E-02$	$1.2023E-02$
	Maxvalue	$7.7201E+00$	$7.6655E+00$	$7.6526E+00$	$7.5727E+00$	$7.5576E+00$
	Minvalue	$7.5575E+00$	$7.5157E+00$	$7.5146E+00$	$7.5143E+00$	$7.5159E+00$

$$f = \frac{\sum_{i=1}^{K} \sum_{p \in C_i} \left\| p - v_i \right\|_2}{n}, \tag{15}$$

where K represents the cluster number, p is a pixel which belongs to C_i, C_i is the ith cluster, v_i is the ith clustering center, and n is the number of pixels of an image.

The experiments were done between DCGBBO, BBO [12], B-BBO [27], BBO-M [29], and TDBBO [32]. N is 50, MI is 100, and Num is 10. We used 4 colour images with a pixel size of $481 * 321$ as experimental data, include "Church1," "Church2," "sunflower," and "bird." We take the mean value, standard deviation value, the maximum, and the minimum as the final result to compare the performance of the algorithms. The smaller the value, the more efficient the algorithms. The comparison results are exhibited in Table 15. The image segmentation results are shown in Figure 7.

From Table 14, we can clearly obtain that the performance of the DCGBBO algorithm is better than that of other algorithms in terms of Mean, Std, Maxvalue, and Minvalue. Although the minimum value of the new algorithm is not as good as the TDBBO algorithm except image "Church2," the performance of DCGBBO is more stable than that of other algorithms. From Figure 7, we can conclude that on "Church1," the segmentation edge of figure f is relatively smooth than that of other figures and the segmentation area is complete. On "Church2" and "Bird," the image segmentation effect of each algorithm is not different, and the target area is relatively completely segmented, but the details

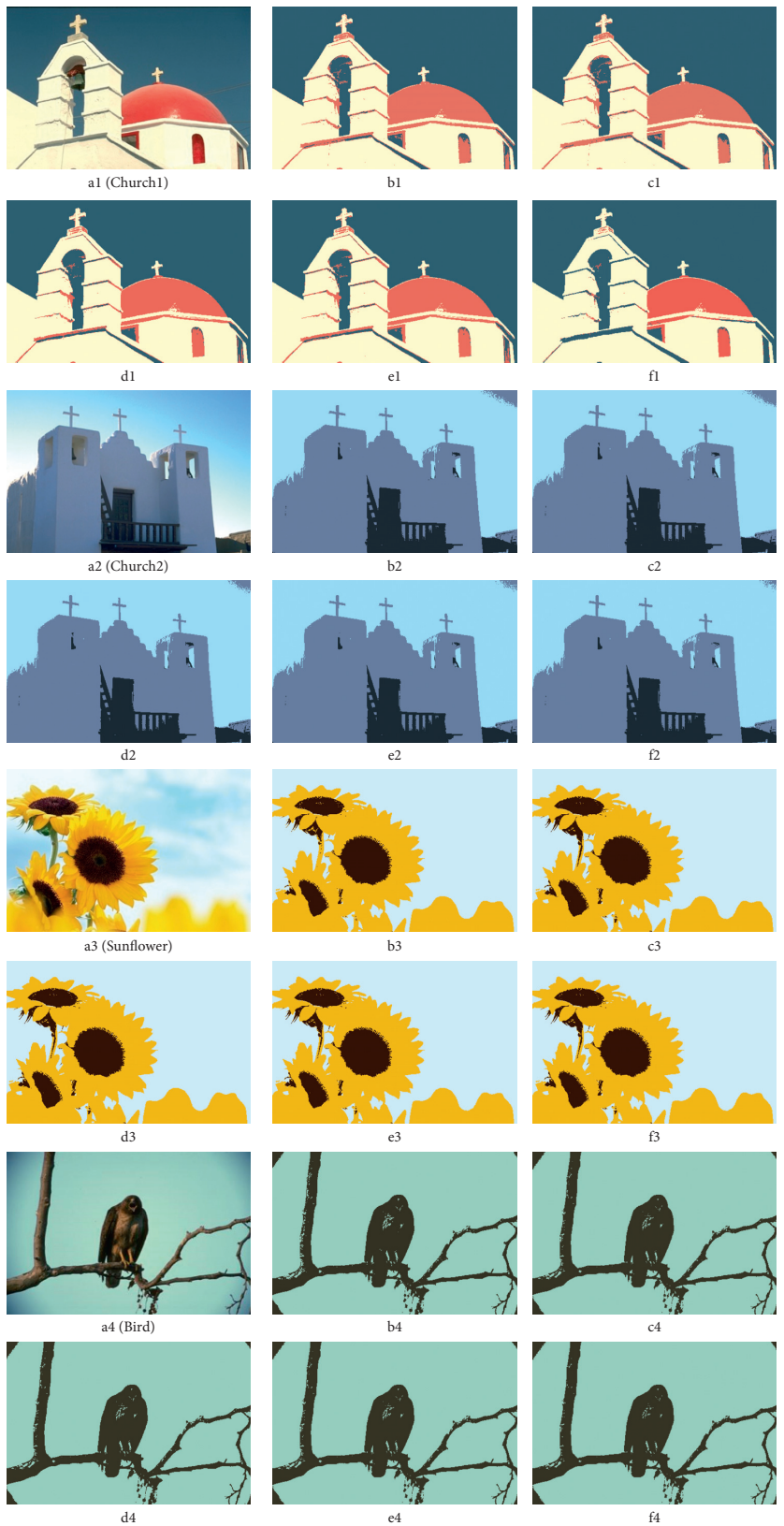

Figure 7: Image segmentation results between DCGBBO and BBO variants on 4 images (a is the original image, b is the segmentation image of BBO, c is the segmentation image of B-BBO, d is the segmentation image of BBO-M, e is the segmentation image of TDBBO, and f is the segmentation image of DCGBBO).

are not good. On "Sunflower," the segmentation effect of each algorithm is well. Through the above analysis, we can conclude that the proposed DCGBBO algorithm is more effective for colour image segmentation.

5. Conclusions

In this paper, we introduce an improved BBO algorithm (DCGBBO) based on hierarchical tissue-like P system with triggering ablation rules. By making full use of a series of rules defined in the algorithm, the iterative process of the optimization algorithm is completed and the effect of the algorithm is improved. For the DCGBBO algorithm, from the evolutionary principle of the algorithm itself, dynamic crossover migration, dynamic Gaussian mutation, opposition-based learning approach, and maximum mutation rate are designed. The above operations can balance the exploitation and exploration ability of the algorithm and improve the optimization efficiency. For the sake of testifying the optimization performance of DCGBBO, a number of experiments are implemented on eight classic benchmark functions. Through a large number of experimental analysis, the optimization effect of the proposed algorithm is better than that of the comparison algorithms. Finally, through segmenting four colour images, our algorithm is proved to be better than other compared algorithms.

Furthermore, except the BBO algorithm, some computational intelligence algorithms such as particle swarm optimization (PSO), artificial bee colony (ABC), ant colony optimization (ACO), differential evolution (DE), Grey Wolf optimizer (GWO), monarch butterfly optimization (MBO), earthworm optimization algorithm (EWA), elephant herding optimization (EHO), moth search (MS) algorithm, slime mould algorithm (SMA), and Harris hawks optimization (HHO) can also tested on some benchmark functions and might be used to segment images. For the other two P systems, there may be more combinations to find the optimal solution.

Data Availability

Data supporting the results of this study can be obtained by contacting the authors. The four images tested in this paper are from BSDS300 and BSDS500, which could be found at https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

Conflicts of Interest

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

Acknowledgments

This research project was supported by the National Natural Science Foundation of China (61876101, 61802234, and 61806114), Social Science Fund Project of Shandong Province, China (16BGLJ06 and 11CGLJ22), Postdoctoral Project, China (2017M612339 and 2018M642695), Natural Science Fund Project of Shandong Province, China (ZR2019QF007), Humanities and Social Sciences Youth Fund of the Ministry of Education, China (19YJCZH244), and Postdoctoral Special Funding Project, China (2019T120607).

References

[1] G. Păun, G. Rozenberg, and A. Salomaa, *The Oxford Handbook of Membrane Computing*, Oxford University Press, Oxford, UK, 2010.

[2] G. Păun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, pp. 108–143, 2000.

[3] C. Martín-Vide, J. Pazos, G. Păun, and A. Rodríguez-Patón, "Tissue P systems," *Theoretical Computer Science*, vol. 296, no. 2, pp. 295–326, 2003.

[4] M. Ionescu, G. Păun, and T. Yokomori, "Spiking neural P systems," *Fundamenta Informaticae*, vol. 71, no. 2, pp. 279–308, 2006.

[5] G.-X. Zhang and L.-Q. Pan, "A survey of membrane computing as a new branch of natural computing," *Chinese Journal of Computers*, vol. 33, no. 2, pp. 208–214, 2010.

[6] R. Freund, G. Păun, and M. J. Pérez-Jiménez, "Tissue P systems with channel states," *Theoretical Computer Science*, vol. 330, no. 1, pp. 101–116, 2005.

[7] B. Song, C. Zhang, and L. Pan, "Tissue-like P systems with evolutional symport/antiport rules," *Information Sciences*, vol. 378, pp. 177–193, 2017.

[8] X. Liu, Y. Zhao, and W. Sun, "Tissue P systems with cooperating rules," *Chinese Journal of Electronics*, vol. 27, no. 2, 2018.

[9] B. Song, X. Zeng, and A. Rodríguez-Patón, "Monodirectional tissue P systems with channel states," *Information Sciences*, vol. 546, pp. 206–219, 2021.

[10] Y. Luo, P. Guo, Y. Jiang et al., "Timed homeostasis tissue-like P systems with evolutional symport/antiport rules," *IEEE Access*, vol. 8, p. 1, 2020.

[11] X. Liu, L. Wang, J. Qu et al., "A complex chained P system based on evolutionary mechanism for image segmentation," *Computational Intelligence and Neuroscience*, vol. 2020, Article ID 6524919, 9 pages, 2020.

[12] D. Simon, "Biogeography-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 702–713, 2008.

[13] J. Kennedy, "Particle swarm optimization," *IEEE International Joint Conference on Neural Network*, vol. 4, pp. 1942–1948, 2011.

[14] Y. Gao, X. Li, M. Dong, and H.-p. Li, "An enhanced artificial bee colony optimizer and its application to multi-level threshold image segmentation," *Journal of Central South University*, vol. 25, no. 1, pp. 107–120, 2018.

[15] C. Blum, "Ant colony optimization: introduction and recent trends," *Physics of Life Reviews*, vol. 2, no. 4, pp. 353–373, 2005.

[16] R. Storn and K. Price, ""Differential evolution—a simple and efficient heuristic for global optimization over continuous space," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[17] L. Deng, S. Wang, L. Qiao et al., "DE-RCO: rotating crossover operator with multiangle searching strategy for adaptive differential evolution," *IEEE Access*, vol. 6, pp. 2970–2983, 2017.

[18] S. Mirjalili, S. Mirgalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, no. 3, pp. 46–61, 2104.

[19] Y. Feng, G. Wang, S. Deb et al., "Monarch butterfly optimization," *Neural Computing & Applications*, 2019.

[20] G. Wang, S. Deb, and L. D. S. Coelho, "Earthworm optimisation algorithm: a bio-inspired metaheuristic algorithm for global optimisation problems," *International Journal of Bio-Inspired Computation*, vol. 12, no. 1, 2015.

[21] G. Wang, S. Deb, and L. D. S. Coelho, "Elephant herding optimization," in *Proceedings of the 2015 3rd International Symposium on Computational and Business Intelligence (ISCBI 2015)*, pp. 1–5, IEEE, Bali, Indonesia, December 2015.

[22] G.-G. Wang, "Moth search algorithm: a bio-inspired metaheuristic algorithm for global optimization problems," *Memetic Computing*, vol. 10, no. 2, pp. 151–164, 2018.

[23] S. Li, H. Chen, M. Wang, A. A. Heidari, and S. Mirjalili, "Slime mould algorithm: a new method for stochastic optimization," *Future Generation Computer Systems*, vol. 111, pp. 300–323, 2020.

[24] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: algorithm and applications," *Future Generation Computer Systems*, vol. 97, pp. 849–872, 2019.

[25] A. W. Mohamed, A. A. Hadi, and A. K. Mohamed, "Gaining-sharing knowledge based algorithm for solving optimization problems: a novel nature-inspired algorithm," *International Journal of Machine Learning and Cybernetics*, vol. 11, no. 5, pp. 1501–1529, 2020.

[26] X. Qi, Z. Yuan, and Y. Song, "A hybrid pathfinder optimizer for unconstrained and constrained optimization problems," *Computational Intelligence and Neuroscience*, vol. 2020, Article ID 5787642, 25 pages, 2020.

[27] H. Ma and D. Simon, "Blended biogeography-based optimization for constrained optimization," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 3, pp. 517–525, 2011.

[28] Y.-J. Zheng, H.-F. Ling, and J.-Y. Xue, "Ecogeography-based optimization: enhancing biogeography-based optimization with ecogeographic barriers and differentiations," *Computers & Operations Research*, vol. 50, pp. 115–127, 2014.

[29] Q. Niu, L. Zhang, and K. Li, "A biogeography-based optimization algorithm with mutation strategies for model parameter estimation of solar and fuel cells," *Energy Conversion and Management*, vol. 86, pp. 1173–1185, 2014.

[30] X. Zhang, D. Wang, and H. Chen, "Improved biogeography-based optimization algorithm and its application to clustering optimization and medical image segmentation," *IEEE Access*, vol. 7, pp. 28810–28825, 2019.

[31] P. Farswan and J. C. Bansal, "Fireworks-inspired biogeography-based optimization," *Soft Computing*, vol. 23, no. 16, pp. 7091–7115, 2019.

[32] F. Zhao, S. Qin, Y. Zhang et al., "Two-stage differential biogeography-based optimization algorithm and its performance analysis," *Expert Systems with Applications*, vol. 115, pp. 329–345, 2018.

[33] W. Gong, Z. Cai, and C. Ling, "DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization," *Soft Computing*, vol. 15, no. 4, pp. 645–665, 2011.

[34] X. Zhang, Q. Kang, Q. Tu et al., "Efficient and merged biogeography-based optimization algorithm for global optimization problems," *Soft Computing*, vol. 23, no. 8, 2018.

[35] X. Zhang, Q. Kang, J. Cheng, and X. Wang, "A novel hybrid algorithm based on biogeography-based optimization and grey wolf optimizer," *Applied Soft Computing*, vol. 67, pp. 197–214, 2018.

[36] A. K. Mohamed and A. W. Mohamed, "Real-parameter unconstrained optimization based on enhanced AGDE algorithm," *Machine Learning Paradigms: Theory and Application*, vol. 801, 2019.

[37] A. W. Mohamed, A. A. Hadi, and K. M. Jambi, "Novel mutation strategy for enhancing SHADE and LSHADE algorithms for global numerical optimization," *Swarm and Evolutionary Computation*, vol. 50, Article ID 100455, 2019.

[38] W. Gong, Z. Cai, C. X. Ling, and H. Li, "A real-coded biogeography-based optimization with mutation," *Applied Mathematics and Computation*, vol. 216, no. 9, pp. 2749–2758, 2010.

[39] H. Tizhoosh, "Opposition-based learning: a new scheme for machine intelligence," *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce*, vol. 695, p. 701, 2005.

[40] R. A. Ibrahim, M. A. Elaziz, and S. Lu, "Chaotic opposition-based grey-wolf optimization algorithm based on differential evolution and disruption operator for global optimization," *Expert Systems with Applications*, vol. 108, 2018.

[41] N. H. Awad, M. Z. Ali, J. J. Liang et al., "Problem definitions and evaluation criteria for the CEC 2017 special session and competition on single objective real-parameter numerical optimization," Technical report, Nanyang Technological University, Jordan University of Science and Technology and Zhengzhou University, Singapore, 2016.

[42] X. Yang and S. Deb, "Cuckoo search via Lévy flights," in *Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, pp. 210–214, IEEE, Kitakyushu, Japan, December 2010.

[43] R. Salgotra, U. Singh, and S. Saha, "New cuckoo search algorithms with enhanced exploration and exploitation properties," *Expert Systems with Applications*, vol. 95, pp. 384–420, 2018.

[44] R. Salgotra, U. Singh, S. Saha et al., "Self adaptive cuckoo search: analysis and experimentation," *Swarm and Evolutionary Computation*, vol. 60, 2020.

[45] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, 2011.

[46] M. Friedman, "The use of ranks to avoid the assumption of normality implicit in the analysis of variance," *Journal of the American Statistical Association*, vol. 32, no. 200, pp. 1–27, 1937.