Research Article

An Enhanced Grasshopper Optimization Algorithm to the Bin Packing Problem

Hangwei Feng ^{1,2} Hong Ni,^{1,2} Ran Zhao,^{1,2} and Xiaoyong Zhu ^{1,2}

¹National Network New Media Engineering Research Center, Institute of Acoustics, Chinese Academy of Sciences, Beijing 100190, China

²University of Chinese Academy of Sciences, Beijing 100049, China

Correspondence should be addressed to Xiaoyong Zhu; zhuxy@dsp.ac.cn

Received 27 August 2019; Revised 22 January 2020; Accepted 10 February 2020; Published 10 March 2020

Academic Editor: Daniel Morinigo-Sotelo

Copyright © 2020 Hangwei Feng et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

The grasshopper optimization algorithm (GOA) is a novel metaheuristic algorithm. Because of its easy deployment and high accuracy, it is widely used in a variety of industrial scenarios and obtains good solution. But, at the same time, the GOA algorithm has some shortcomings: (1) original linear convergence parameter causes the processes of exploration and exploitation unbalanced; (2) unstable convergence speed; and (3) easy to fall into the local optimum. In this paper, we propose an enhanced grasshopper optimization algorithm (EGOA) using a nonlinear convergence parameter, niche mechanism, and the β -hill climbing technique to overcome the abovementioned shortcomings. In order to evaluate EGOA, we first select the benchmark set of GOA authors to test the performance improvement of EGOA compared to the basic GOA. The analysis includes exploration ability, exploitation ability, and convergence speed. Second, we select the novel CEC2019 benchmark set to test the optimization ability of EGOA in complex problems. According to the analysis of the results of the algorithms in two benchmark sets, it can be found that EGOA performs better than the other five metaheuristic algorithms. In order to further evaluate EGOA, we also apply EGOA to the engineering problem, such as the bin packing problem. We test EGOA and five other metaheuristic algorithms in SchWae2 instance. After analyzing the test results by the Friedman test, we can find that the performance of EGOA is better than other algorithms in bin packing problems.

1. Introduction

Bin packing problem (BPP) is one of the most important combinatorial optimization problems. It is widely used in many fields of science and engineering and is the basis of many practical engineering optimization problems, including processors scheduling [1] and cloud computing resource allocation [2] in the field of computer science. BPP belongs to NP-hard problems [3], so that basic heuristic algorithms has been proposed for solving BPP. The basic heuristic algorithms can ensure fast and accurate solutions for small-scale bin packing problems. But, the performance of this way starts to deteriorate with the increase of scale. Thus, metaheuristic algorithms which can find a high-quality solution though adjusting the exploitation and exploration of search space become a new trend to solve BPP.

In [4], the authors combine a genetic algorithm with a grouping mechanism. In [5], the authors apply first fit and ranked order values (ROVs) in CS to solve the BPP. In [6], the authors solve the BPP using an improved whale optimization algorithm.

Metaheuristic algorithms have developed rapidly and have been widely applied in many different fields, such as signal detection [7], resource allocation [8], load balancing [9], feature selection [10], task scheduling [11], and engineering applications [12]. The representative algorithms contain Dragonfly Algorithm (DA) [13], Ant Lion Optimization (ALO) [14], Bee Algorithm (BA) [15], and Particle Swarm Optimization (PSO) [16, 17]. These algorithms seek inspiration from physical or biological phenomena to solve optimization problems, including ant colony foraging, bird migrating, bee colony behavior, grey wolves hunting, and

fish schooling [18]. It has been proved in practice that they are superior to traditional optimization methods in many application scenarios.

The metaheuristic algorithms form a randomly initialized population for a given problem, evaluate the solutions using the objective function (s), and improve the random solutions during the iteration until satisfying the terminating condition in order to find the global optimum. This unique way to find solution give them the advantages as follows: (i) simplicity, mainly inspired by fairly simple concepts and are easy to apply; (ii) flexibility, no special requirements on the objective function; and (iii) independent of derivation, no need to calculate the derivative in the process of finding the optimal solution. These advantages make metaheuristic algorithms not only limited to specific problems but also have a wide range of applications.

Although there are differences between different metaheuristic algorithms, they all have one feature in common, which is the two phases during the search process: exploration and exploitation. In the exploration phase, the algorithm explores the search space as widely as possible for promising regions. While in the exploitation phase, it pays more attention to local search ability for the optimal solution around promising regions. How to balance the relationship between exploration and exploitation has become the key to design and improve a new metaheuristic algorithm.

The GOA is a state-of-the-art population-based metaheuristic algorithm that was inspired by the long-range and abrupt movements of adult grasshoppers in groups [19]. Also, GOA is widely used in a variety of industrial scenarios. Aljarah proposes a hybrid approach based on the grasshopper to optimize the parameters of the SVM model [20]. Hekimoğlu Baran and Ekinci Serder employ GOA to solve many optimization problems in an automatic voltage regular system [21]. Wu proposes a dynamic GOA for optimizing the distributed trajectories of unmanned aerial vehicles in urban environments [22]. In [23], the authors apply the basic multiobjective GOA to solve several benchmark problems with superior performance.

While the GOA can obtain good solutions in a reasonable timeframe, it presents some shortcomings: (1) original linear convergence parameter makes the processes of exploration and exploitation unbalanced, (2) unstable convergence speed, and (3) easy to fall into the local optimum. Some studies have proposed improved GOA algorithms. Luo proposes an improved grasshopper using levy flight in [24]. Tharwat applies GOA for constrained and unconstrained multiobjective optimization problems in [25]. Ewees introduces an opposition-based learning strategy to the GOA in [26]. This strategy improves the ability of jumping out of local optimum and movement issues. But, the improvement of algorithm in other shortcomings is not significant because they consider neither the balance between exploration and exploitation nor the relationship between the diversity of the population and the convergence speed during the group optimization process.

To overcome these disadvantages, this paper proposed an enhanced grasshopper optimization algorithm. The main contributions of this paper could be summarized as follows:

- (i) We introduced a nonlinear convergence parameter into the basic GOA to balance the exploration and exploitation phases to improve the overall performance.
- (ii) We applied a niche repulsing mechanism to the basic GOA. This mechanism directs group optimization and ensures the diversity of the search space to increase the speed of convergence.
- (iii) We adapted the β -hill climbing (BHC) technique to GOA to avoid falling into the local optimums.

The remainder of this paper is organized as follows. The basic GOA and EGOA are introduced in Sections 2 and 3. Several experiments of two benchmark sets are implemented and the analysis results are shown in Section 4. The BPP and the applicability of the EGOA to the BPP are discussed in Section 5. Finally, conclusions and future works are discussed in Section 6.

2. Basic Grasshopper Optimization Algorithm

2.1. Biological Inspiration. Grasshoppers are considered to be pests based on the damage they inflict on crops and vegetation. Instead of acting individually, grasshoppers form some of the largest swarms among all living creatures. Millions of grasshoppers jump and move like large rolling cylinders. The influences of the individuals in a swarm, wind, gravity, and food sources all affect swarm movement.

2.2. Main Procedures for the GOA. The GOA is a novel swarm-intelligence-based metaheuristic algorithm that was inspired by the long-range and abrupt movements of adult grasshoppers in groups. Food source seeking is an important behavior of grasshopper swarms. Metaheuristic algorithms logically divide the search process into two phases: exploration and exploitation. The long-range and abrupt movements of grasshoppers represent the exploration phase, and local movements to search for better food sources represent the exploitation phase.

A mathematical model for this behavior was presented by Mirjalili in [19]. This model is defined as follows:

$$x_i = S_i + G + A, \quad (1)$$

where x_i is the position of the i grasshopper, S_i represents the social interaction in a group, G is the force of gravity acting on the i grasshopper, and A is the wind direction. By expanding S_i , G and A in (1), the equation can be rewritten as follows:

$$x_i = \sum_{j=1, j \neq i}^N s(|x_j - x_i|) \frac{x_j - x_i}{d_{ij}} - g\hat{e}_g + u\hat{e}_w, \quad (2)$$

where $s(r) = fe^{-r/l} - e^{-r}$ is a function simulating the impact of social interactions and N is the number of grasshoppers. $g\hat{e}_g$ is the expanded G component, where g is the gravitational force and $\hat{e}_g$ is a unit vector pointing toward the center of the earth. $u\hat{e}_w$ is the expanded A component, where u is a constant drift and $\hat{e}_w$ is a unit vector pointing in the

direction of the wind. d_{ij} is the distance between the i and j grasshoppers and is calculated as $d_{ij} = |x_j - x_i|$.

Because grasshoppers quickly find comfortable zones and exhibit poor convergence, the influences of wind and gravity are far weaker than the relationships between grasshoppers, meaning this mathematical model should be modified as follows:

$$x_i = c \left(\sum_{j=1, j \neq i}^N c \frac{\text{ub} - \text{lb}}{2} s(|x_j - x_i|) \frac{x_j - x_i}{d_{ij}} \right) + \widehat{T}_d, \quad (3)$$

where ub and lb represent the upper and lower boundaries of the search space, $\widehat{T}_d$ is the value relative to the target (best solution found so far), and c is a decreasing coefficient that balances the processes of exploitation and exploration, which is defined as follows:

$$c = c_{\max} - \text{iter} \frac{c_{\max} - c_{\min}}{\text{Max}_{\text{iter}}}, \quad (4)$$

where $c_{\max}$ is the maximum value (equal to 1), $c_{\min}$ is the minimum value (equal to 0.00001), iter represents the current iteration, and Max_{iter} is the maximum number of iterations.

3. Enhanced Grasshopper Optimization Algorithm

First, the EGOA utilizes a nonlinear convergence parameter to balance the exploration phase exploitation phase. Second, the EGOA is hybridized with a niche mechanism to balance diversity and convergence speed. Finally, the EGOA is combined with the β -hill climbing technique to avoid local optimum.

3.1. Nonlinear Convergence Parameter. In metaheuristic algorithm, the convergence parameter c directly affects the step size. The larger the convergence parameter, the larger the step size. In the exploration phase of metaheuristic algorithm, the step size should be as large as possible to ensure that the algorithm can search the global optimum in a wider range to prevent falling into the local optimum. In the exploitation phase, the step size should be as small as possible to ensure that the algorithm gradually converges to the global optimum to prevent skipping the optimal value. In the basic GOA, the convergence parameter c decreases linearly in the two phases of exploration and exploitation. Also, the size of the step cannot be balanced according to the characteristics of the two phases, which affects the performance of the algorithm. In order to make up for the lack of linear convergence parameters, we introduce the tanh function, a nonlinear adjustment parameter widely used in the field of deep learning with good performance. Tanh function can compensate for the lack of GOA nonlinear convergence parameters and the characteristics of exploration and exploitation phases based on the nonlinear characteristics to improve performance at different phases of GOA. The formula for the tanh function is defined as follows:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (5)$$

A nonlinear convergence parameter based on a variant of the tanh function is introduced as follows:

$$f = \frac{e^{-l_c+5} - e^{-l_c-5}}{e^{-l_c+5} + e^{-l_c-5}}, \quad (6)$$

where l_c is defined as follows:

$$l_c = w \frac{\text{iter}}{50}, \quad (7)$$

where iter is the current iteration and w is the adjustment factor.

3.2. Niche Mechanism. The idea of the GOA algorithm to solve the problem is that grasshoppers interact with each other. When solving problems, the agent in GOA mimics the grasshoppers' interaction and searches the global optimum in the solution space. In the process of the agent searching for the global optimum, when a local optimum is found, all agents will move closer to this local optimum. If the moving is too fast, the agent swarm loses diversity. This leads to a local optimum and slows down convergence speed. In order to balance the convergence speed and diversity of GOA, we introduce niche technology.

Niche technology is inspired by the natural phenomenon where creatures tend to live with similar creatures. It can be used to balance GOA's convergence speed and the diversity of a swarm. Therefore, to maintain the diversity of the swarm, we design a niche repulsing mechanism for the EGOA. Through this mechanism, the algorithm maintains the diversity of the swarm while preserving local optimum. The simulation of this mechanism is defined as follows:

$$d_{km} = \sqrt{\sum_{j=1}^{\text{dim}} (x_{k,j} - x_{m,j})^2}, \quad k = 1, 2, \dots, N; \\ m = 1, 2, \dots, N, k \neq m,$$

$$\overline{D} = \frac{(\sum_{i=1}^n d_i)}{N}. \\ \begin{cases} \text{fitness}(V_k) = p, & \text{fitness}(V_k) > \text{fitness}(V_m), \\ \text{fitness}(V_m) = p, & \text{otherwise.} \end{cases} \quad (8)$$

First, calculate d_{km} , where d_{km} is the Euclidean distance between a search agent x_k and search agent x_m (there are N search agents). Then, get d_i and calculate $\overline{D}$, where d_i is the minimum distance between search agent x_i and any other search agent in the population, dim is the dimensionality of the search agents. Afterwards, when d_{km} is less than $\overline{D}$, compare the fitnesses of x_k and x_m , and penalty is assigned to the worse one. If $\text{fitness}(x_k) > \text{fitness}(x_m)$, then $\text{fitness}(x_k)$ is p ; otherwise, $\text{fitness}(x_m)$ is p . Finally, after all individuals have been processed, they are ranked according to their fitness. $m\% N$

individuals with poor fitness will be eliminated. To maintain the population and increase diversity, we select the best and worse $m\%N$ individuals to generate new individuals.

$$\text{new}x_i = \text{rand} * x_{\text{best}} + \text{rand} * x_{\text{worst}}. \quad (9)$$

3.3. The β -Hill Climbing Technique. As the iterative process continues, the GOA is similar to many other metaheuristic algorithms. It only focuses on the process of converging to the global optimum and ignores the mechanism of avoiding the local optimum. When falling into a local optimum, GOA search cannot continue. This is very fatal when solving practical engineering problems.

To allow the algorithm to escape from local optimum, the β -hill climbing (BHC) technique is introduced [27] in the EGOA. The BHC technique can iteratively enrich a series of randomly approximated solutions. In the BHC technique, a stochastic strategy called the β -operator is employed to establish a fine balance between exploration and exploitation throughout a global search.

The BHC technique begins exploitation with a search agent $x_i = (x_{i,1}, x_{i,2}, x_{i,3}, \dots, x_{i,\text{dim}})$ that represents a poor solution. It uses the β -operator throughout its exploitative steps to generate a new solution $x'_i = (x'_{i,1}, x'_{i,2}, x'_{i,3}, \dots, x'_{i,\text{dim}})$. The elements of the new solution are updated according to their current values or filled randomly with a probability of β as follows:

$$x'_i = \begin{cases} U(0, 1) * (\text{ub} - \text{lb}) + \text{lb}, & z < \beta, \\ (\text{ub} + \text{lb}) - x_i, & \text{otherwise}, \end{cases} \quad (10)$$

where $U(0, 1)$ and z represent a random value in the range $(0, 1)$. Next, the BHC technique compares x'_i to x_i and records the best (downhill) value as follows:

$$x_i = \begin{cases} x'_i, & \text{fitness}(x'_i) \leq \text{fitness}(x_i), \\ x_i, & \text{otherwise}. \end{cases} \quad (11)$$

In terms of exploration, the BHC technique's ability to escape from local optima solutions using the β -operator can be considered as the key concept for avoiding local optimum (Algorithm 1).

4. Experimental Results on Benchmarks

In this section, the efficiency of the proposed EGOA is compared to that of the basic GOA and several other metaheuristic algorithms, namely Dragonfly Algorithm (DA), Ant Colony Optimizer (ALO), Particle Swarm Optimization (PSO), and opposition-based learning GOA (OBLGOA) based on various benchmark problems. All codes utilized in this section were implemented consistently using MATLAB 9.3 (R2017b) and executed on a PC with the Windows 10 64 bit professional operating system and 16 GB of RAM. In order to fully verify the performance of EGOA, this paper cites two benchmark sets: The benchmark set 1 is the GOA original algorithm test environment. This part of benchmark set is used to verify the performance improvement of the original algorithm and the comparison with other algorithms

[28, 29]. The benchmark set 2 is the most novel metaheuristic algorithm performance benchmark set CEC2019 [30]. It can be further verified that EGOA also has good performance in the recent benchmark set. To investigate the differences between the results obtained by the EGOA and those obtained by the other algorithms, a nonparametric Wilcoxon rank sum test [31] with a significance level of 5% was adopted in this study. The Wilcoxon rank sum test generates a p value to determine if two datasets came from the same distributed set. The higher the p value is, the more similar the two datasets are. If the p value is less than 0.05, then the two datasets are considered to be independent.

4.1. Benchmark Set 1. The benchmark set 1 used in this paper contains 29 test functions. Different test functions have different characteristics, which can comprehensively measure the performance of the algorithm. Test functions include unimodal, multimodal, fixed-dimension multimodal functions, and composite functions which are listed in Tables 1 and 2. In the tables, the column labelled Function represents the objective fitness function. The column labelled Dim represents the dimensionality of each function, Range represents the boundaries of the search space, $f_{\min}$ represents the optimal value, and Type represents the function type.

For the unimodal, multimodal, and fixed-dimension multimodal functions, each of the benchmark functions uses 30 search agents over 500 iterations. The presented results were recorded based on 30 independent trials with random initial conditions to calculate statistical results, including the average fitness (avg) which represents the average performance and reliability of the algorithm, standard deviation of fitness (std) which represents the stability of the algorithm, best fitness (best) which represents the best optimization ability of the algorithm, and worst fitness (worst) which represents the capability boundary.

4.1.1. Evaluation of Exploitation Capability. Unimodal functions, which have only one global optimum, can test the exploitation capabilities of algorithms. Table 3 shows the results for the unimodal functions. It can be clearly seen that EGOA achieves best performance in F_1 , F_2 , and F_4 and second best performance in F_7 . Moreover, EGOA outperforms the other algorithms in F_3 in terms of the avg fitness and the best avg fitness. In addition, EGOA achieves the best avg fitness among all compared algorithms in F_5 . Besides, most of the p values in Table 4 are much less than 0.05 which means that EGOA and other solutions can be considered irrelevant. From the abovementioned results, we can conclude that EGOA has a good exploitation capability. The main reasons for the superior performance of EGOA are the embedded BHC local search and the exploitative patterns inherited from the basic GOA.

4.1.2. Evaluation of Exploration Capability. Multimodal functions, which have several local optimums, are used for evaluating exploration ability. Tables 5 and 6 list the results of the compared algorithms on multimodal test functions and fixed-dimension multimodal functions, respectively. The


```

Initialize the swarm position  $X(x_i, i = 1, \dots, N)$  and the parameters
Initialize  $c_{\max}$ ,  $c_{\min}$ , and  $\text{Max}_{\text{iter}}$  as the maximum number of iterations
Calculate the fitness of each search agent and let  $T$  represent the best fitness
while ( $\text{iter} < \text{Max}_{\text{iter}}$ ) do
    Calculate  $f$  using equation (6)
    for  $i = 1 : N$  do
        Update  $x_i$  using equation (3)
        Calculate the fitness
        if current fitness is worse than target fitness
        then
             $x_i$  conducts BHC Using equations (10) and (11)
        end if
        Bring the current search agent back if it travels outside the boundaries
    end for
     $X$  conducts niche mechanism using equation (9)
    Update  $T$  and position
     $\text{iter} = \text{iter} + 1$ 
end while
Return the target fitness and target position

```

ALGORITHM 1: Enhanced grasshopper optimization algorithm.

EGOA works best in 10 of the 16 benchmark tests. Although, among the remaining six benchmark tests that do not achieve the best results, the results of F_8 , F_{13} , F_{15} , and F_{19} are all very close to the optimal values. These results show that the EGOA has competitive exploration ability. Also, the p values presented in Table 4 demonstrate that it has statistical significance.

4.1.3. Ability to Avoid Local Minima. Composite functions, which have composite values, are very challenging test problems for metaheuristic algorithms. Furthermore, they are suitable for benchmarking both exploration and exploitation simultaneously for a massive number of local optimums. The results for composite benchmark functions listed in Table 7 indicate that the EGOA outperforms the other algorithms in F_{24} and F_{29} , gets the best avg fitness in F_{25} , and achieves the best std fitness and avg fitness in F_{27} . The results can demonstrate that the EGOA is able to escape from local optimums.

4.1.4. Analysis of EGOA Convergence Curves. In this subsection, the convergence speed of the EGOA is analyzed. Several benchmark functions were chosen and executed with 30 search agents over 500 iterations. The results for the EGOA, GOA, DA, ALO, PSO, and OBLGOA are presented in Figure 1. The abscissa uses the number of current iterations, and the ordinate represents the best fitness value in each iteration. The representative convergence curves in each part of benchmark functions are selected, which can show that EGOA has an outstanding performance in solving different kinds of benchmark functions compared to well-known metaheuristic algorithm. During the iterations of F_1 , F_4 , and F_{10} , the convergence speed of EGOA is approximately in the middle of the other five comparison algorithms before about 250 iterations. But, after 250 iterations, an inflection point appears, and EGOA accelerates the convergence process. This is because EGOA is hybridized with a niche mechanism to provide better individual direction and faster convergence

speed. Besides, during the iterations of F_8 , F_{15} , and F_{21} , the slope of the convergence curve is larger in the initial iteration, and after several jumps, it jumps out of the local optimum and finally obtains the optimal fitness. This can be explained as EGOA utilizes a nonlinear convergence parameter f to balance the two search phases, and the combination of EGOA and β -hill climbing technique can improve the ability to jump out of local optimization. Overall, the convergence curves demonstrate that the proposed EGOA provides relatively fast convergence speed in most of the iterations.

4.2. Benchmark Set 2. The CEC2019 benchmark set contains 10 test functions. Unlike the previous CEC benchmark set, the complexity of the test function is significantly increased, and more attention is paid to the ability of evaluation algorithms to find an accurate solution. In Table 8, the column labelled Function represents the objective fitness function. The column labelled Dim represents the dimensionality of each function, Range represents the boundaries of the search space, and $f_{\min}$ represents the optimal value.

Each test function uses 30 search agents for 500 iterations. The presented results were recorded based on 30 independent trials with random initial conditions to calculate statistical results, including the average fitness (avg), standard deviation of fitness (std), best fitness (best), and worst fitness (worst) which represent the same meaning as mentioned above in the benchmark set 1. To assess the accuracy of the algorithm, in this part of the solution space, we need to focus more on average fitness and best fitness.

4.2.1. Evaluation of EGOA's Performance in CEC2019. The CEC2019 test functions, which have complicated solution space, are more challenging test problems for metaheuristic algorithms. Furthermore, they are suitable for evaluating the algorithm's ability to find accurate solutions in the complicated search process. The results

TABLE 1: Benchmark functions.

Function	Dim	Range	$f_{\min}$	Type
$F_1(x) = \sum_{i=1}^n x_i^2$	30	$[-100, 100]$	0	Unimodal
$F_2(x) = \sum_{i=1}^n x_i + \prod_{i=1}^n x_i $	30	$[-10, 10]$	0	Unimodal
$F_3(x) = \sum_{i=1}^n (\sum_{j=1}^i x_j)^2$	30	$[-100, 100]$	0	Unimodal
$F_4(x) = \max_i \{ x_i , 1 \leq i \leq n\}$	30	$[-100, 100]$	0	Unimodal
$F_5(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-30, 30]$	0	Unimodal
$F_6(x) = \sum_{i=1}^n (x_i + 0.5)^2$	30	$[-100, 100]$	0	Unimodal
$F_7(x) = \sum_{i=1}^n ix_i^4 + \text{random}[0, 1]$	30	$[-1.28, 1.28]$	0	Unimodal
$F_8(x) = \sum_{i=1}^n -x_i \sin(\sqrt{ x_i })$	30	$[-500, 500]$	-418 * Dim	Multimodal
$F_9(x) = \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i) + 10]$	30	$[-5.12, 5.12]$	0	Multimodal
$F_{10}(x) = -20 \exp(-0.2 \sqrt{(1/n) \sum_{i=1}^n x_i^2}) - \exp((1/n) \sum_{i=1}^n \cos(2\pi x_i)) + 20 + e$	30	$[-32, 32]$	0	Multimodal
$F_{11}(x) = (1/4000) \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos(x_i / \sqrt{i}) + 1$	30	$[-600, 600]$	0	Multimodal
$F_{12}(x) = (\pi/n) \{10 \sin(\pi y_1) + \sum_{i=1}^{n-1} (y_i - 1)^2 [1 + 10 \sin^2(\pi y_{i+1})] + (y_n - 1)^2\} (y_n - 1)^2 + \sum_{i=1}^n u(x_i, 10, 100, 4)$ $y_i = 1 + (x_i + 1/4)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m x_i & x_i > a \\ 0 - a & x_i < a \\ k(-x_i - a)^m x_i & x_i < -a \end{cases}$	30	$[-50, 50]$	0	Multimodal
$F_{13}(x) = 0.1 \{ \sin^2(3\pi x_1) + \sum_{i=1}^n (x_i - 1)^2 [1 + \sin^2(3\pi x_i + 1)] + (x_n - 1)^2 [1 + \sin^2(2\pi x_n)] \} + \sum_{i=1}^n u(x_i, 5, 100, 4)$	30	$[-50, 50]$	0	Multimodal
$F_{14}(x) = ((1/500) + \sum_{j=1}^{25} (1/(j + \sum_{i=1}^2 (x_i - a_{ij})^6)))^{-1}$	2	$[-65, 65]$	0	Fixed-dimension multimodal
$F_{15}(x) = \sum_{i=1}^{11} [a_i - ((x_i(b_i^2 + b_i x_2)) / (b_i^2 + b_i x_3 + x_4))]^2$	4	$[-5, 5]$	0.00030	Fixed-dimension multimodal
$F_{16}(x) = 4x_1^2 - 2.1x_1^4 + (1/3)x_1^6 + x_1 x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]$	-1.0316	Fixed-dimension multimodal
$F_{17}(x) = (x_2 - (5.1/4\pi^2)x_1^2 + (5/\pi)x_1 - 6)^2 + 10(1 - (1/8\pi)\cos x_1) + 10$	2	$[-5, 5]$	0.3979	Fixed-dimension multimodal
$F_{18}(x) = [1 + (x_1 + x_2 + 1)^2 (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1 x_2 + 3x_2^2)] \times [30 + 2x_1 - 3x_2] \times (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1 x_2 + 27x_2^2)$	2	$[-2, 2]$	3	Fixed-dimension multimodal
$F_{19}(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^3 a_{ij}(x_j - p_{ij})^2)$	3	$[1, 3]$	-3.86	Fixed-dimension multimodal

TABLE 1: Continued.

Function	Dim	Range	$f_{\min}$	Type
$F_{20}(x) = -\sum_{i=1}^4 c_i \exp(-\sum_{j=1}^6 a_{ij}(x_j - p_{ij})^2)$	6	[0, 1]	-3.32	Fixed-dimension multimodal
$F_{21}(x) = -\sum_{i=1}^5 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	-10.1532	Fixed-dimension multimodal
$F_{22}(x) = -\sum_{i=1}^7 [(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	-10.4028	Fixed-dimension multimodal
$F_{23}(x) = -\sum_{i=1}^1 0[(X - a_i)(X - a_i)^T + c_i]^{-1}$	4	[0, 10]	-10.5363	Fixed-dimension multimodal

TABLE 2: Composite functions.

Function	Dim	Range	$f_{\min}$	Type
F_{24} (CF1) $f_1, f_2, f_3, \dots, f_{10}$ = Sphere Function, $[\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10}] = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [5/100, 5/100, 5/100, \dots, 5/100]$	30	$[-5, 5]$	0	Composite
F_{25} (CF2) $f_1, f_2, f_3, \dots, f_{10}$ = Griewank's Function, $[\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10}] = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [5/100, 5/100, 5/100, \dots, 5/100]$	30	$[-5, 5]$	0	Composite
F_{26} (CF3) $f_1, f_2, f_3, \dots, f_{10}$ = Griewank's Function, $[\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10}] = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [1, 1, 1, \dots, 1]$	30	$[-5, 5]$	0	Composite
F_{27} (CF4) f_1, f_2 = Ackley's Function, f_3, f_4 = Rastrigin's Function, f_5, f_6 = Weierstrass Function, f_7, f_8 = Griewank's Function, f_9, f_{10} = Sphere Function, $[\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10}] = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [5/32, 5/32, 5/32, \dots, 5/32]$	30	$[-5, 5]$	0	Composite
F_{28} (CF5) f_1, f_2 = Rastrigin's Function, f_3, f_4 = Weierstrass Function, f_5, f_6 = Griewank's Function, f_7, f_8 = Ackley's Function, f_9, f_{10} = Sphere Function, $[\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10}] = [1, 1, 1, \dots, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [1/5, 1/5, 5/0.5, 5/0.5, 5/100, 5/100, 5/32, 5/32, 5/100, 5/100]$	30	$[-5, 5]$	0	Composite
F_{29} (CF6) f_1, f_2 = Rastrigin's Function, f_3, f_4 = Weierstrass Function, f_5, f_6 = Griewank's Function, f_7, f_8 = Ackley's Function, f_9, f_{10} = Sphere Function $[\sigma_1, \sigma_2, \sigma_3, \dots, \sigma_{10}] = [0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1]$ $[\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_{10}] = [0.1 * 1/5, 0.2 * 1/5, 0.3 * 5/0.5, 0.4 * 5/0.5, 0.5 * 5/100, 0.6 * 5/100, 0.7 * 5/32, 0.8 * 5/32, 0.9 * 5/100, 1 * 5/100]$	30	$[-5, 5]$	0	Composite

TABLE 3: Results for unimodal functions.

Function	Type	EGOA	GOA	DA	ALO	PSO	OBLGOA
F_1	avg	9.0534E-14	0.9730	37.2776	0.0014	3.1013E-05	3.2306E-05
	std	3.7011E-14	0.8698	81.6186	0.0011	7.9301E-05	2.3431E-05
	best	1.9846E-14	0.0172	3.5526	0.0002	2.9928E-07	5.2292E-06
	worst	1.8459E-13	3.8065	450.1749	0.0048	0.0004	1.0108E-04
F_2	avg	1.2287E-07	18.0367	4.4112	30.2392	0.0776	0.0146
	std	2.7105E-08	36.4320	2.3929	37.3048	0.1459	0.0390
	best	7.2962E-08	0.0306	1.3295	1.1430	0.0003	0.0013
	worst	2.0402E-07	118.1239	10.9685	114.7681	0.6804	0.1775
F_3	avg	0.0028	1413.8190	1128.2543	4431.5768	228.0791	0.0054
	std	0.0036	628.5017	1139.0449	1821.2736	173.7777	0.0018
	best	8.5499E-13	475.2364	214.3597	2059.2765	63.3951	0.0019
	worst	0.0121	2784.6786	6174.2426	9027.3843	732.1625	0.0087
F_4	avg	1.4732E-07	8.6396	30.4830	19.6348	3.7143	0.0122
	std	5.2777E-08	2.9120	7.9770	4.8814	1.6919	0.0084
	best	6.4635E-08	5.0805	16.6437	11.1498	1.6923	0.0024
	worst	2.6566E-07	16.9399	52.9255	30.7645	7.5366	0.0304
F_5	avg	27.9972	932.4950	4273.4106	347.3018	168.3237	28.4168
	std	0.6484	1473.4525	16369.5087	439.0214	543.5966	0.2120
	best	26.2374	28.0057	91.6157	28.5465	12.5981	27.9057
	worst	29.0806	6233.7871	90522.6022	1608.0766	3038.7872	28.7632
F_6	avg	0.0476	0.8813	45.0163	0.0019	8.9790E-05	1.5503
	std	0.0389	1.0279	124.4759	0.0018	0.0003	0.4663
	best	0.0061	0.0048	2.5149	0.0002	2.2448E-07	0.7925
	worst	0.1730	4.4407	670.9717	0.0070	0.0018	2.7873
F_7	avg	0.0131	0.0277	0.1641	0.2639	0.0330	0.0014
	std	0.0061	0.0106	0.0870	0.0707	0.0146	0.0006
	best	0.0047	0.0101	0.0441	0.1493	0.0135	0.0004
	worst	0.0264	0.0605	0.3609	0.4763	0.0819	0.0028

TABLE 4: The Wilcoxon rank sum test for multimodal functions.

Function	GOA	DA	ALO	PSO	OBLGOA
F_1	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$
F_2	$8.9934E-11$	$3.0199E-11$	$3.0199E-11$	$5.0723E-10$	$5.5727E-10$
F_3	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	0.0001
F_4	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$
F_5	$7.7725E-09$	$3.0199E-11$	0.0002	0.0064	0.3953
F_6	$1.5292E-05$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$
F_7	$3.0939E-06$	$3.0199E-11$	$3.0199E-11$	$1.0277E-06$	$3.0199E-11$
F_8	0.8187	0.0271	$1.1841E-09$	$2.7726E-05$	0.5895
F_9	$6.6914E-11$	$2.9566E-11$	$1.7616E-11$	$2.3050E-08$	$1.3989E-05$
F_{10}	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$
F_{11}	$3.0180E-11$	$3.0180E-11$	$3.0180E-11$	$3.0180E-11$	$3.0180E-11$
F_{12}	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	0.1907	$7.2208E-06$
F_{13}	$3.8307E-05$	$3.6897E-11$	$5.9673E-09$	$5.5999E-07$	$1.2493E-05$
F_{14}	$7.9359E-10$	$1.5600E-11$	$1.8519E-10$	0.0005	$1.6415E-11$
F_{15}	0.0003	0.0040	0.0004	0.0073	0.0001
F_{16}	$7.7686E-12$	$7.7686E-12$	$7.7686E-12$	$3.0802E-07$	$7.7686E-12$
F_{17}	$1.2118E-12$	$1.2118E-12$	$4.4703E-12$	NaN	$1.2118E-12$
F_{18}	$3.0047E-11$	$3.0047E-11$	$3.0047E-11$	$2.0056E-09$	$3.0047E-11$
F_{19}	$6.5238E-07$	$3.1565E-05$	0.125964933	$1.1279E-11$	$5.4604E-06$
F_{20}	0.9705	0.5688	0.7503	$5.5530E-05$	0.0404
F_{21}	0.0000	0.0061	0.0013	0.0624	0.0144
F_{22}	0.0002	0.0002	0.0271	0.1019	0.0406
F_{23}	0.0004	0.0163	0.0010	0.6038	0.0519
F_{24}	0.2581	0.1224	$3.0199E-11$	$4.4440E-07$	0.0421
F_{25}	0.0679	0.0224	$3.0199E-11$	0.3403	0.0615
F_{26}	0.0103	0.0018	$3.0199E-11$	0.0011	0.0002
F_{27}	$6.0459E-07$	$2.1327E-05$	$3.0199E-11$	$8.8829E-06$	$1.1567E-07$
F_{28}	0.0022	0.0701	$3.0199E-11$	0.0064	0.0271
F_{29}	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$3.0199E-11$	$4.1825E-09$

TABLE 5: Results for multimodal functions.

Function	Type	EGOA	GOA	DA	ALO	PSO	OBLGOA
F_8	avg	-7870.7306	-7579.2715	-7376.4956	-5681.4728	-6619.9581	-7671.5200
	std	649.2707	900.0826	882.9292	694.0512	784.2524	730.6476
	best	-9102.6546	-9467.2412	-9220.7846	-9004.5042	-8481.6966	-9154.0221
	worst	-6794.9884	-5681.3700	-5402.0041	-5417.6748	-5343.1670	-6321.3596
F_9	avg	0.5506	9.7174	7.0345	9.5848	5.3064	1.1282
	std	1.1216	7.7979	4.1399	5.0157	3.1063	1.5848
	best	0.0000	0.9950	0.9952	0.9950	0.9950	$6.0484E-09$
	worst	4.0747	32.8334	16.9164	19.8991	14.9244	5.9717
F_{10}	avg	6.8570E-08	3.1128	8.1242	6.0103	1.7712	0.0010
	std	1.8729E-08	1.0787	3.9469	3.6325	0.7301	0.0002
	best	4.2697E-08	1.6466	2.8930	2.3181	0.0003	0.0006
	worst	1.1399E-07	6.2071	16.6509	14.3756	2.9579	0.0016
F_{11}	avg	3.9540E-13	0.7293	1.1663	0.0628	0.0367	0.0003
	std	5.6432E-13	0.1782	0.1652	0.0302	0.0730	$8.1399E-05$
	best	5.0071E-14	0.2872	1.0366	0.0120	$3.4788E-07$	0.0001
	worst	3.1125E-12	1.0295	1.7957	0.1268	0.3924	0.0005
F_{12}	avg	0.0596	4.7713	20.6637	11.6964	0.4780	0.0443
	std	0.0465	2.2841	8.1149	4.4639	0.6003	0.0198
	best	$1.5136E-02$	0.8415	7.9487	4.8224	2.3850E-09	0.0124
	worst	0.2322	9.1428	39.2302	22.3226	2.3453	0.0888
F_{13}	avg	0.8174	7.6309	103.8394	22.4201	0.1950	0.4521
	std	0.3692	10.0853	336.0417	14.6435	0.6831	0.3147
	best	0.2073	0.0571	2.6543	0.0120	2.9372E-07	0.0783
	worst	1.4061	40.2498	1865.8767	52.4418	3.6765	1.6232

TABLE 6: Results for fixed-dimension multimodal functions.

Function	Type	EGOA	GOA	DA	ALO	PSO	OBLGOA
F_{14}	avg	0.9980	1.0641	2.0501	2.1827	4.0791	3.2005
	std	1.93E-16	0.3622	2.0262	1.9746	3.2178	2.4733
	best	0.9980	0.9980	0.9980	0.9980	0.9980	0.9980
	worst	0.9980	2.9821	10.7632	7.8740	12.6705	10.7632
F_{15}	avg	0.0005	0.0063	0.0023	0.0035	0.0004	0.0076
	std	0.0003	0.0087	0.0049	0.0067	0.0004	0.0098
	best	0.0003	0.0006	0.0004	0.0005	0.0003	0.0003
	worst	0.0013	0.0204	0.0204	0.0204	0.0016	0.0224
F_{16}	avg	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	std	4.7012E-16	1.0871E-12	2.2380E-06	9.7720E-14	6.6486E-16	3.5117E-06
	best	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
	worst	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316	-1.0316
F_{17}	avg	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
	std	0.0000	6.0985E-13	7.5449E-07	3.5239E-14	0.0000	1.0888E-06
	best	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
	worst	0.3979	0.3979	0.3979	0.3979	0.3979	0.3979
F_{18}	avg	3.0000	13.8000	3.0000	3.0000	3.9000	8.4000
	std	4.6795E-15	28.0054	1.5781E-05	9.0080E-13	4.929503018	20.5504
	best	3.0000	3.0000	3.0000	3.0000	3.0000	3.0000
	worst	3.0000	84.0000	3.0001	3.0000	30.0000	84.0006
F_{19}	avg	-3.8628	-3.8091	-3.8628	-3.8628	-3.8628	-3.8357
	std	5.0160E-07	0.155858005	1.2444E-06	5.5947E-13	2.6962E-15	0.1481
	best	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628	-3.8628
	worst	-3.8628	-3.0333	-3.8628	-3.8628	-3.8628	-3.0513
F_{20}	avg	-3.2341	-3.2665	-3.2767	-3.2620	-3.2665	-3.2388
	std	0.0584	0.0603	0.0606	0.0611	0.0603	0.1028
	best	-3.3220	-3.3220	-3.3220	-3.3220	-3.3220	-3.3220
	worst	-3.1803	-3.2030	-3.1831	-3.2000	-3.2031	-3.0253
F_{21}	avg	-8.2885	-4.7947	-6.3718	-6.7001	-5.0570	-7.8820
	std	2.4926	2.6812	2.8591	2.9905	2.8139	2.9107
	best	-10.1532	-10.1532	-10.1532	-10.1532	-10.1532	-10.1532
	worst	-5.0552	-2.6305	-2.6305	-2.6305	-2.6305	-2.6304
F_{22}	avg	-8.5972	-7.3266	-6.1121	-7.4811	-5.8788	-7.7390
	std	2.8452	3.6392	3.1788	3.2560	3.5547	3.4005
	best	-10.4029	-10.4029	-10.4029	-10.4029	-10.4029	-10.4029
	worst	-2.7659	-2.7519	-2.7518	-2.7519	-2.7519	-2.7658
F_{23}	avg	-9.4383	-5.5672	-4.7293	-6.0268	-5.2543	-7.0597
	std	2.5121	3.6810	2.8552	3.5713	3.3498	3.8262
	best	-10.5364	-10.5364	-10.5364	-10.5364	-10.5364	-10.5364
	worst	-3.1054	-1.8595	-2.4217	-2.4217	-2.4217	-2.4217

for CEC2019 test functions listed in Table 9 indicate that EGOA works best in f_4 and f_{10} and second in f_5 and f_6 . Meanwhile, it gets the best avg fitness in 7 of the 10 benchmark tests and achieves the best fitness in 5 of 10. The results demonstrate that compared with other algorithms, EGOA has a good average performance and higher reliability. At the same time, it shows the best optimization ability in most search processes.

4.2.2. Analysis of EGOA's Convergence Curves in CEC2019.

In this subsection, the convergence behavior of the EGOA in CEC2019 test functions is investigated. Several benchmark functions were chosen and executed with 30 search agents over 500 iterations. The results over part of the benchmark functions are presented in Figure 2. The parameter space in Figure 1 indicates that the solution space of the CEC2019 test function is

more complex than test functions solution space in benchmark set 1, and it is more challenging to find the optimal fitness. During the iterations of f_4 , f_7 , and f_{10} , EGOA keeps a large slope relatively in the initial iteration, and after several sudden drops of the curve, it converges to the optimal fitness in the search space, which is far superior to other algorithms. Although during the iterations of f_2 OBLGOA achieves the best result, EGOA ranks second. Considering comprehensively, it can still prove that the proposed EGOA has also superior search ability in more complex solution spaces, compared with well-known metaheuristics.

5. Application of the EGOA to the BPP

5.1. BPP Formulation. The BPP consists of packing a set of items with different weights into a minimum number of bins, which may have different capacities. Mathematically,

TABLE 7: Results for composite functions.

Function	Type	EGOA	GOA	DA	ALO	PSO	OBLGOA
F_{24}	avg	84.4598	122.8630	188.7522	1123.4550	266.8987	188.0859
	std	59.7037	127.0180	109.6276	106.0951	100.0898	88.8280
	best	13.7184	20.9414	86.6431	901.6364	144.8265	40.9847
	worst	254.6628	515.3122	576.9355	1272.6311	586.9204	359.9907
F_{25}	avg	299.2990	326.4544	337.6506	1178.6612	443.8959	399.0075
	std	134.7602	160.2337	166.3636	79.6722	132.5885	156.4962
	best	48.4789	33.0915	87.4429	997.4426	181.6229	69.3652
	worst	563.9395	528.9073	575.9922	1303.5229	647.2719	625.4685
F_{26}	avg	735.5623	681.8806	866.8562	1532.2853	618.5467	887.9904
	std	139.1450	277.1884	181.9240	128.6867	142.0694	151.7927
	best	418.2166	285.5568	636.9803	1705.2732	386.0311	536.5325
	worst	919.4273	1249.5640	1357.3310	1174.8147	881.3347	1209.6244
F_{27}	avg	892.2331	942.9682	989.6282	1408.3321	755.0872	891.5666
	std	42.5450	122.9602	76.1249	48.0027	127.5822	46.2035
	best	666.9724	648.5125	695.3967	1309.5712	497.6061	646.9350
	worst	900.0019	1040.3032	1082.9054	1479.5117	1025.0401	900.0087
F_{28}	avg	264.4913	163.9287	233.8788	1343.2227	308.1830	557.7770
	std	204.3920	141.5067	211.9575	142.2203	127.8672	352.8388
	best	105.9604	55.0450	93.3352	1017.2573	128.6536	138.0444
	worst	900.0010	545.2874	1038.8526	1534.4240	722.3016	900.0052
F_{29}	avg	900.0003	907.2424	924.6529	1357.1334	923.0487	900.0005
	std	0.0001	4.8004	6.1472	50.5599	8.8742	0.0005
	best	900.0001	900.2385	914.0881	1187.8816	911.5907	900.0001
	worst	900.0007	918.4818	936.4468	1460.9597	944.8864	900.0022

the BPP can be defined as a programming problem as follows [32]:

$$\begin{aligned}
 & \min \sum_{i=1}^n y_i, \\
 & \sum_{j=1}^n w_j z_{ij} \leq C y_i, \quad i \in N = \{1, \dots, n\}, \\
 & \sum_{i=1}^n z_{ij} = 1, \quad j \in N, \\
 & y_i \in \{0, 1\}, \quad i \in N \\
 & z_{ij} \in \{0, 1\}, \quad i \in N, j \in N,
 \end{aligned} \tag{12}$$

where y_i is a binary variable that represents whether or not bin i contains any items, z_{ij} is a binary variable that indicates whether or not item j is assigned to bin i , w_j is the weight of item j , C is the bin capacity, and n is the number of available bins. The goal is to minimize the number of bins. The sum of the sizes of the items assigned to any bin cannot exceed its capacity.

Use of the bin number as a fitness function may lead to algorithm stagnation because there are many solutions, which have the same number of bins. However, this information can be helpful if integrated with other information, such as the fullness of the bins. The following formulation was introduced in [33]:

$$\min F = 1 - \left(\frac{\sum_{i=1}^{nbin} (\text{ocup}_i / C)^s}{nbin} \right), \tag{13}$$

where $nbin$ is the number of used bins, ocup_i is the occupancy of each bin i , which is the sum of the weights of all items packed, C is the capacity, and s is a constant that defines an equilibrium point for filling bins.

5.2. Discretization Search Space. The basic version of the GOA was proposed for optimization problems in continuous search spaces. To apply such algorithm to combinatorial search spaces, different techniques have been proposed, such as largest order values [34], ROV [35], smallest position values [36], and largest ranked values [37]. These techniques transform continuous solutions into permutations with different orders. In this study, the ROV [35] rule was adopted for experimentation. The ROV rule is a simple method based on random key representations for forming permutations, as shown in Table 10. We assign items to available bins according to the first-fit policy. In the first-fit policy, each item is placed in the first bin that has the capacity to hold the item.

In the EGOA, the current best number of bins is compared to the theoretical optimal solution. The optimal solution can be computed as follows:

$$\text{optimal number} = \frac{\sum \text{itemsizes}}{\text{bin capacity}}, \tag{14}$$

where itemsizes represents the number of items that fit into a bin and bin capacity is equal to C . If the current best number of bins is equal to the optimal number, the search stops.

5.3. Experimental Results. In order to test the validity of the proposed algorithm, some experiments are conducted on selected benchmarks from the "SchWae2" instances [38] in which EGOA is compared to other metaheuristic algorithms. For the parameter settings, small numbers of search agents and iterations are selected for the EGOA and other metaheuristic algorithm. Specifically, 30 agents

(a)

FIGURE 1: Continued.

(b)

FIGURE 1: Some convergence curves for benchmark set 1.

TABLE 8: The 100-digit challenge test function.

Function	Dim	Range	$f_{\min}$
f_1 Storns Chebyshev polynomial fitting problem	9	$[-8192, 8192]$	1
f_2 Inverse Hilbert matrix problem	16	$[-16384, 16384]$	1
f_3 Lennard-Jones Minimum energy cluster	18	$[-4, 4]$	1
f_4 Shifted and rotated rastrigins function	10	$[-100, 100]$	1
f_5 Shifted and rotated Griewanks function	10	$[-100, 100]$	1
f_6 Shifted and rotated Weierstrass function	10	$[-100, 100]$	1
f_7 Shifted and rotated Schwefels function	10	$[-100, 100]$	1
f_8 Shifted and rotated expanded Schaffers F_6 function	10	$[-100, 100]$	1
f_9 Shifted and rotated Happy Cat function	10	$[-100, 100]$	1
f_{10} Shifted and rotated Ackley function	10	$[-100, 100]$	1

TABLE 9: Results for CEC2019 functions.

Function	Type	EGOA	GOA	DA	ALO	PSO	OBLGOA
f_1	avg	1.5212E+05	1.9571E+06	4.6646E+04	1.1544E+07	1.8749E+06	1.2357E+05
	std	1.8946E+05	1.8015E+06	1.4480E+05	1.4062E+07	1.4164E+06	1.4289E+05
	best	1.0000	3.2527E+05	1.0000	8.0545E+05	2.1036E+03	8.3500E+02
	worst	5.2933E+05	4.9750E+06	4.5874E+05	3.9307E+07	4.4657E+06	4.4661E+05
f_2	avg	271.2334	719.7666	116.6510	2664.5301	3490.0571	217.4122
	std	361.7063	645.6242	148.6381	2243.3673	1450.5327	92.5807
	best	4.3805	311.4380	5.0277	432.7258	1432.2960	103.6213
	worst	996.9688	2384.2235	372.9532	8234.1964	5455.4163	438.8886
f_3	avg	2.8669	9.2929	3.1398	8.5125	3.4387	2.1356
	std	1.0038	2.2391	1.1173	1.4137	1.9444	1.7195
	best	1.7707	4.6082	1.4266	6.4314	1.4091	1.4091
	worst	4.3937	11.7111	5.2141	11.7116	7.1563	6.7118
f_4	avg	27.3663	27.6417	33.7491	38.0307	27.3972	31.8436
	std	9.6834	11.4310	14.8956	14.0494	13.4154	11.2370
	best	8.9597	13.9345	8.9738	22.7559	9.9546	17.9143
	worst	40.7982	47.0947	59.7039	67.5145	52.0411	58.7070
f_5	avg	1.2161	1.2897	1.3012	1.5997	1.2200	1.1464
	std	0.0770	0.1119	0.1121	0.2468	0.0906	0.0386
	best	1.0664	1.0813	1.1022	1.1276	1.0688	1.0787
	worst	1.3175	1.4304	1.4504	1.7923	1.3469	1.1968
f_6	avg	4.3797	4.0630	6.2424	7.2472	6.1541	2.6168
	std	1.6749	2.1278	1.7631	1.8683	1.6317	1.1671
	best	1.6809	1.7761	3.0929	4.0192	3.6529	1.0000
	worst	6.5611	9.3312	8.5585	9.5316	8.6161	4.7035
f_7	avg	920.2188	1183.3734	1254.7191	1195.1984	1318.8619	1058.0542
	std	325.6570	419.8797	254.7967	350.7395	555.2455	295.0898
	best	272.5803	479.3381	884.4280	815.0716	423.4350	641.5707
	worst	1307.0650	1834.6642	1561.1209	1877.9551	2318.1617	1532.6045
f_8	avg	4.0581	4.1528	4.4705	4.6317	4.5015	4.0682
	std	0.4209	0.4581	0.3054	0.4536	0.3332	0.4385
	best	3.1266	3.5196	3.9365	3.4530	3.8255	3.1707
	worst	4.5145	5.0039	4.9994	5.0139	5.0444	4.5417
f_9	avg	1.1803	1.2349	1.3449	1.3285	1.2919	1.2322
	std	0.0732	0.1509	0.1349	0.1052	0.0957	0.1046
	best	1.1162	1.1229	1.1485	1.2140	1.1649	1.0843
	worst	1.3133	1.5899	1.5767	1.5636	1.4580	1.4399
f_{10}	avg	21.0092	21.0149	21.0839	21.3776	21.0221	21.1974
	std	0.0293	0.0421	0.0894	0.1305	0.0595	0.1520
	best	20.9998	20.9999	21.0170	21.1255	20.9999	21.0000
	worst	21.0927	21.1339	21.3149	21.6091	21.1900	21.4238

are selected and the maximum number of iterations is set to 100. The values of the bin capacities, items weights, and numbers of items affect the solution quality of the

algorithm. Therefore, a selection of benchmark should be taken into account the previously cited values. Our experiments are conducted using the SchWae2 instances,

(a)
FIGURE 2: Continued.

FIGURE 2: Some convergence curves for benchmark set 2.

TABLE 10: Mapping between continuous solutions and discrete permutations using ROVs.

Continuous solution	1.23	0.39	5.36	3.64	4.59	2.00
Discrete permutation	2	1	6	4	5	3

where the weights of items are generated using the BPPGEN method discussed in [38]. BPPGEN assumes that components are uniformly distributed in the interval $[v1 * C, v2 * C]$ and determines weight values based on the following two steps:

- (i) Generation of a realization of a weight w_i , which is distributed in $(v1 * C, v2 * C + 1)$, $w_i = (v1 + (v2 - v1) * \text{rand}_i(0, 1)) * C + \text{rand}_i(0, 1)$
- (ii) Rounding down to the nearest integer, where the weight of the item is the resulting integer ($w_i = \lfloor w_i \rfloor$)

In addition, the Friedman test is one of the nonparametric tests, which is applied to test the average ranking values of these algorithms. The Friedman test [31] is next used to analyze the experimental results to show the performance difference between EGOA and the comparison algorithm.

The obtained results for the SchWae2 instances are summarized in Table 11. The first column labelled Number represents the numbers of items to be placed into bins. The column labelled C represents the capacity of the bin. The column labelled $v1v2$ represents the upper and lower bounds of the weight distributions. Also, the column labelled Optimal represents the theoretical optimal numbers of bins. The columns labelled GOA, EGOA, ALO, DA, PSO, and OBLGOA represent the results of the corresponding algorithms.

Different pairs of $v1, v2$ determine the difficulty levels of the problems. A greater value of $v2$ results in more complicated item weights. When $v1$ is equal to 0.1 and $v2$ is equal

TABLE 11: Experimental results for the BPP.

Number	C	$v_1 v_2$	Optimal	GOA	EGOA	ALO	DA	PSO	OBLGOA
100	1000	0.1 0.2	16	16	16	17	16	17	16
100	1000	0.1 0.2	16	16	16	17	16	16	16
100	1000	0.1 0.2	16	17	17	18	17	18	17
100	1000	0.1 0.2	16	16	16	17	17	17	16
100	1000	0.1 0.2	15	16	16	16	16	16	16
100	1000	0.1 0.2	16	16	16	16	17	17	17
100	1000	0.1 0.2	16	16	16	16	17	16	17
100	1000	0.1 0.2	16	16	16	16	17	16	17
100	1000	0.1 0.2	15	16	16	16	17	16	16
100	1000	0.1 0.2	16	16	16	16	17	17	17
100	1000	0.1 0.3	21	22	21	23	23	23	22
100	1000	0.1 0.3	21	22	22	23	23	22	22
100	1000	0.1 0.3	21	22	21	23	23	24	22
100	1000	0.1 0.3	20	21	21	22	22	23	21
100	1000	0.1 0.3	20	21	21	22	21	22	21
100	1000	0.1 0.3	20	20	20	20	21	22	21
100	1000	0.1 0.3	20	21	21	21	22	22	22
100	1000	0.1 0.3	21	22	22	22	23	23	24
100	1000	0.1 0.3	20	21	21	21	23	22	22
100	1000	0.1 0.3	21	23	22	22	24	23	24
100	1000	0.1 0.4	25	26	26	28	28	29	27
100	1000	0.1 0.4	24	26	25	27	26	27	26
100	1000	0.1 0.4	25	27	26	29	28	27	26
100	1000	0.1 0.4	27	28	28	31	29	31	29
100	1000	0.1 0.4	26	27	27	30	29	29	28
100	1000	0.1 0.4	25	27	26	26	28	27	28
100	1000	0.1 0.4	26	29	28	29	31	31	29
100	1000	0.1 0.4	26	28	27	28	30	30	29
100	1000	0.1 0.4	25	27	26	26	29	28	29
100	1000	0.1 0.4	24	25	25	25	26	26	27
100	1000	0.1 0.5	32	35	34	38	35	39	36
100	1000	0.1 0.5	31	35	34	37	38	36	35
100	1000	0.1 0.5	31	34	33	37	35	37	34
100	1000	0.1 0.5	31	34	33	37	36	37	34
100	1000	0.1 0.5	30	34	32	36	33	36	34
100	1000	0.1 0.5	29	32	31	31	33	33	34
100	1000	0.1 0.5	30	33	32	33	35	33	37
100	1000	0.1 0.5	30	32	31	33	36	32	36
100	1000	0.1 0.5	30	34	33	33	37	36	35
100	1000	0.1 0.5	29	33	31	33	35	33	35

FIGURE 3: Friedman test for different pair $v_1 v_2$.

to 0.2, the weights of items are ranging between 100 and 200. Similarly, when v_1 is equal to 0.1 and v_2 is equal to 0.5, the weights of items are ranging between 100 and 500. Also, the bin capacity is equal to 1000. The results of our experiments demonstrate that EGOA is the same as or closer to the optimal value compared to other algorithms in all cases which indicates the good searching performance of EGOA. In addition, Figure 3 is the conclusion drawn by Friedman test. It can be intuitively seen that EGOA ranks first, indicating that EGOA has excellent and stable performance in BPPs of different difficulty levels.

6. Conclusions

In this paper, a new variant of the GOA called the EGOA has been presented. First, a nonlinear convergence parameter, niche mechanism, and the BHC technique are added to improve the performance of the GOA. Next, we evaluate EGOA based on the benchmark set proposed by the GOA authors and the CEC2019 benchmark set and compare with the original GOA, DA, ALO, PSO, and OBLGOA in performance. The results of experiments and the Wilcoxon rank sum test indicate that the EGOA has excellent exploitation capability and competitive exploration capability. Finally, the EGOA is applied to the BPP. The EGOA is tested on the SchWae2 instances and compared to the GOA, DA, ALO, PSO, and OBLGOA algorithms. The experimental results and Friedman test show the ability of the EGOA to find optimal solutions in different problem sizes efficiently.

In the future, we wish to apply the EGOA to other variants of the BPP and engineering problems. Further studies could reveal additional methods for enhancing the EGOA to solving additional optimization problems.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This work was partially supported by Strategic Leadership Project of Chinese Academy of Science: SEANET Technology Standardization Research System Development (Project no. XDC02070100).

References

- [1] M. Xu, L. T. X. Phan, H. Y. Choi et al., "Holistic resource allocation for multicore real-time systems," in *Proceedings of the 2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pp. 345–356, IEEE, Montreal, Canada, April 2019.
- [2] W. Song, Z. Xiao, Q. Chen et al., "Adaptive resource provisioning for the cloud using online bin packing," *IEEE Transactions on Computers*, vol. 63, no. 11, pp. 2647–2660, 2013.
- [3] S. Anily, J. Bramel, and D. Simchi-Levi, "Worst-case analysis of heuristics for the bin packing problem with general cost structures," *Operations Research*, vol. 42, no. 2, pp. 287–298, 1994.
- [4] E. Falkenauer, "A hybrid grouping genetic algorithm for bin packing," *Journal of Heuristics*, vol. 2, no. 1, pp. 5–30, 1996.
- [5] Z. Zendaoui and A. Layeb, "Adaptive cuckoo search algorithm for the bin packing problem," *Modelling and Implementation of Complex Systems*, pp. 107–120, Springer, Cham, Switzerland, 2016.
- [6] M. Abdel-Basset, G. Manogaran, L. Abdel-Fatah et al., "An improved nature inspired meta-heuristic algorithm for 1-D bin packing problems," *Personal and Ubiquitous Computing*, vol. 22, no. 5-6, pp. 1117–1132, 2018.
- [7] A. Hamad, E. H. Houssein, A. E. Hassanien, and A. A. Fahmy, "Hybrid grasshopper optimization algorithm and support vector machines for automatic seizure detection in EEG signals," in *Proceedings of the International Conference on Advanced Machine Learning Technologies and Applications*, pp. 82–91, Cairo, Egypt, February 2018.
- [8] P. Kaur and S. Mehta, "Resource provisioning and work flow scheduling in clouds using augmented Shuffled Frog Leaping Algorithm," *Journal of Parallel and Distributed Computing*, vol. 101, pp. 41–50, 2017.
- [9] S. Dam, G. Mandal, K. Dasgupta et al., "An ant-colony-based meta-heuristic approach for load balancing in cloud computing," *Applied Computational Intelligence and Soft Computing in Engineering*, pp. 204–232, IGI Global, Pennsylvania, PA, USA, 2018.
- [10] A. A. Ewees, M. A. El Aziz, and A. E. Hassanien, "Chaotic multi-verse optimizer-based feature selection," *Neural Computing and Applications*, vol. 31, no. 4, pp. 991–1006, 2019.
- [11] S. H. H. Madni, M. S. A. Latiff, M. Abdullahi et al., "Performance comparison of heuristic algorithms for task scheduling in IaaS cloud computing environment," *PLoS One*, vol. 12, no. 5, Article ID e0176321, 2017.
- [12] M. Mavrovouniotis, C. Li, and S. Yang, "A survey of swarm intelligence for dynamic optimization: algorithms and applications," *Swarm and Evolutionary Computation*, vol. 33, pp. 1–17, 2017.
- [13] S. Mirjalili, "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems," *Neural Computing and Applications*, vol. 27, no. 4, pp. 1053–1073, 2016.
- [14] E. S. Ali, S. M. Abd Elazim, and A. Y. Abdelaziz, "Ant lion optimization algorithm for renewable distributed generations," *Energy*, vol. 116, pp. 445–458, 2016.
- [15] W. A. Hussein, S. Sahran, and S. N. H. Sheikh Abdullah, "The variants of the Bees algorithm (BA): a survey," *Artificial Intelligence Review*, vol. 47, no. 1, pp. 67–121, 2017.
- [16] R. Eberhart and J. Kennedy, "Particle swarm optimization," *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995.
- [17] A. Ghodousian and M. R. Parvari, "A modified PSO algorithm for linear optimization problem subject to the generalized fuzzy relational inequalities with fuzzy constraints (FRI-FC)," *Information Sciences*, vol. 418–419, pp. 317–345, 2017.
- [18] A. E. Hassanien and E. Emary, *Swarm Intelligence: Principles, Advances, and Applications*, CRC Press, Boca Raton, FL, USA, 2016.
- [19] S. Saremi, S. Mirjalili, and A. Lewis, "Grasshopper optimisation algorithm: theory and application," *Advances in Engineering Software*, vol. 105, pp. 30–47, 2017.
- [20] I. Aljarah, A. M. Al-Zoubi, H. Faris, M. A. Hassonah, S. Mirjalili, and H. Saadeh, "Simultaneous feature selection

- and support vector machine optimization using the grasshopper optimization algorithm,” *Cognitive Computation*, vol. 10, no. 3, pp. 478–495, 2018.
- [21] H. Baran and S. Ekinici, “Grasshopper optimization algorithm for automatic voltage regulator system,” in *Proceedings of the 2018 5th International Conference on Electrical and Electronic Engineering (ICEEE)*, IEEE, Istanbul, Turkey, May 2018.
- [22] J. Wu, H. Wang, N. Li et al., “Distributed trajectory optimization for multiple solar-powered UAVs target tracking in urban environment by Adaptive Grasshopper Optimization Algorithm,” *Aerospace Science and Technology*, vol. 70, pp. 497–510, 2017.
- [23] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, and I. Aljarah, “Grasshopper optimization algorithm for multi-objective optimization problems,” *Applied Intelligence*, vol. 48, no. 4, pp. 805–820, 2018.
- [24] J. Luo, H. Chen, Q. Zhang, Y. Xu, H. Huang, and X. Zhao, “An improved grasshopper optimization algorithm with application to financial stress prediction,” *Applied Mathematical Modelling*, vol. 64, pp. 654–668, 2018.
- [25] A. Tharwat, E. H. Houssein, M. M. Ahmed, A. E. Hassanien, and T. Gabel, “MOGOA algorithm for constrained and unconstrained multi-objective optimization problems,” *Applied Intelligence*, vol. 48, no. 8, pp. 2268–2283, 2018.
- [26] A. A. Ewees, M. Abd Elaziz, and E. H. Houssein, “Improved grasshopper optimization algorithm using opposition-based learning,” *Expert Systems with Applications*, vol. 112, pp. 156–172, 2018.
- [27] A. A. Heidari, I. Aljarah, H. Faris et al., “An enhanced associative learning-based exploratory whale optimizer for global optimization,” *Neural Computing and Applications*, pp. 1–27, 2019.
- [28] M. Molga and C. Smutnicki, “Test functions for optimization needs,” 2005.
- [29] X. S. Yang, “Test problems in optimization,” 2010, <http://arxiv.org/abs/1008.0549>.
- [30] K. V. Price, N. H. Awad, and M. Z. Ali, *The 100-Digit Challenge: Problem Definitions and Evaluation Criteria for the 100-Digit Challenge Special Session and Competition on Single Objective Numerical Optimization*, Nanyang Technological University, Singapore, 2018.
- [31] 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.
- [32] S. Martello, *Knapsack Problems: Algorithms and Computer Implementations*, Wiley-Interscience Series in Discrete Mathematics and Optimization, Wiley, Hoboken, NJ, USA, 1990.
- [33] M. Hyde, G. Ochoa, T. Curtois et al., *A Hyflex Module for the One Dimensional Bin-Packing Problem*, School of Computer Science, University of Nottingham, Nottingham, UK, 2009.
- [34] B. Qian, L. Wang, R. Hu et al., “A hybrid differential evolution method for permutation flow-shop scheduling,” *The International Journal of Advanced Manufacturing Technology*, vol. 38, no. 7-8, pp. 757–777, 2008.
- [35] B. Liu, L. Wang, B. Qian, and Y. Jin, “Hybrid particle swarm optimization for stochastic flow shop scheduling with no-wait constraint,” *IFAC Proceedings Volumes*, vol. 41, no. 2, pp. 15855–15860, 2008.
- [36] M. Fatih Tasgetiren, Y.-C. Liang, M. Sevkli, and G. Gencyilmaz, “Particle swarm optimization and differential evolution for the single machine total weighted tardiness problem,” *International Journal of Production Research*, vol. 44, no. 22, pp. 4737–4754, 2006.
- [37] J. J. Liang, Q. K. Pan, C. Tiejun et al., “Solving the blocking flow shop scheduling problem by a dynamic multi-swarm particle swarm optimizer,” *The International Journal of Advanced Manufacturing Technology*, vol. 55, no. 5–8, pp. 755–762, 2011.
- [38] P. Schwerin and G. Wäscher, “The bin-packing problem: a problem generator and some numerical experiments with FFD packing and MTP,” *International Transactions in Operational Research*, vol. 4, no. 5-6, pp. 377–389, 1997.