Hindawi
Complexity
Volume 2020, Article ID 8870783, 19 pages
https://doi.org/10.1155/2020/8870783

Research Article

A Dynamic Opposite Learning Assisted Grasshopper Optimization Algorithm for the Flexible Job Scheduling Problem

Yi Feng,[1] Mengru Liu,[1] Yuqian Zhang,[2] and Jinglin Wang[3]

[1] *Dalian University of Technology, Dalian, China*
[2] *Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China*
[3] *University of Nottingham Ningbo China, Ningbo, China*

Correspondence should be addressed to Yuqian Zhang; yq.zhang1@siat.ac.cn

Received 6 August 2020; Revised 21 September 2020; Accepted 26 November 2020; Published 30 December 2020

Academic Editor: Dan Selisteanu

Job shop scheduling problem (JSP) is one of the most difficult optimization problems in manufacturing industry, and flexible job shop scheduling problem (FJSP) is an extension of the classical JSP, which further challenges the algorithm performance. In FJSP, a machine should be selected for each process from a given set, which introduces another decision element within the job path, making FJSP be more difficult than traditional JSP. In this paper, a variant of grasshopper optimization algorithm (GOA) named dynamic opposite learning assisted GOA (DOLGOA) is proposed to solve FJSP. The recently proposed dynamic opposite learning (DOL) strategy adopts the asymmetric search space to improve the exploitation ability of the algorithm and increase the possibility of finding the global optimum. Various popular benchmarks from CEC 2014 and FJSP are used to evaluate the performance of DOLGOA. Numerical results with comparisons of other classic algorithms show that DOLGOA gets obvious improvement for solving global optimization problems and is well-performed when solving FJSP.

1. Introduction

Global optimization has become an important research topic in scientific research and engineering applications. Neural network training [1, 2], path planning [3, 4], industrial design [5, 6], and many other complex problems hope to use optimization algorithms to achieve optimum solutions. Global optimization algorithms mainly consist of two classes: deterministic mathematical programming method [7–10] and metaheuristic algorithms (MAs) [11–14]. MAs have simple structure and strong adaptability, which is very advantageous in solving complex problems [15, 16]. In recent decades, inspired by nature, MAs with many different characteristics have been proposed by researchers. Glover et al. proposed tabu search (TS) [17] by simulating the feature that human has memory function. Simulated annealing algorithm (SA) [18] was proposed by simulating the industrial annealing process of high temperature object. Based on Darwin's evolution theory and Mendel's genetics, Holland and John proposed genetic algorithm (GA) [19] for the first time in the 1970s. Dorigo and Sttzle proposed ant colony optimization (ACO) [20] based on the study of ant colony behavior. Inspired by the foraging behavior of birds, American psychologist Kennedy and electrical engineer Eberhart proposed particle swarm optimization (PSO) in 1995 [13]. In addition to these classical algorithms, some new MAs have been proposed in recent years, such as grey wolf optimizer (GWO) [21], moth flame optimization (MFO) [22], firefly algorithm (FA) [23], teaching-learning-based optimization (TLBO) [24, 25] competitive swarm optimization (CSO) [26, 27], dragonfly algorithm (DA) [28], whale optimization algorithm (WOA) [29], and pigeon-inspired optimization [30].

The grasshopper optimization algorithm (GOA) is a recently proposed new MA. It was inspired by the foraging behavior of grasshopper and proposed by Saremi et al. in

2017 [31]. The algorithm has simple principle and distinctive features and demonstrated good performance in solving optimization problems. As soon as the algorithm was proposed, scholars applied it to different optimization problems. Mirjalili et al. applied GOA to the multiobjective optimization problem in 2017 [32]. Tumuluru and Ravi used GOA to update the weight of the deep confidence neural network and applied it to the classification of cancer in 2017 [33]. Aljarah et al. used GOA for feature selection and support vector machine optimization for solving feature selection task in 2018 [34].

In addition to the original GOA, many variants have emerged for the improvements of the algorithm performance. In 2017, Wu et al. proposed an adaptive grasshopper optimization algorithm (AGOA) and applied it to the UAV tracking trajectory optimization problem [35]. In 2018, Ewees et al. proposed an improved version of the grasshopper optimization algorithm based on the opposition-based learning strategy called OBLGOA for solving benchmark optimization functions and engineering problems [36]. Although the existing method has greatly improved GOA, there is still a large probability of falling into local optimum. This paper adopts the DOL strategy proposed by Xu et al. [37] to improve the optimization performance of GOA. The search space of DOL strategy is asymmetric and dynamic, which greatly improves the probability of GOA to obtain the optimal solution and avoids it falling into local optimum to some extent.

The job shop scheduling problem (JSP) [38] is a famous NP hard problem in the discrete manufacturing system, and flexible job shop scheduling problem (FJSP) is an extension of the classical JSP [39, 40]. In FJSP, there may be more than one machine for the same process, so two subproblems should be considered when solving FJSP: the machine selection (MS) and the operations sequencing (OS) [41]. Although there is only one more step than JSP to assign a series of optional machines to a specified process, it is much more difficult to get optimal solution to the FJSP in polynomial time. Intelligent optimization algorithm can obtain the approximate solution with better quality in a short time, due to which an increasing number of intelligent optimization algorithms are used to solve FJSP. The literature shows that many swarm intelligent optimization algorithms such as GA [42–44], TS [45, 46], and artificial bee colony algorithm (ABC) [47, 48] have been successfully applied to FJSP and obtained good results. Though numerous approaches have been proposed for solving the given problem, the complex FJSP problem calls for more competitive solvers for solving the problem. In order to further test the performance of DOLGOA, we use DOLGOA algorithm to solve FJSP.

The main contribution of this paper is as follows:

(1) An encoding scheme FJSP is discussed in detail, and the mathematical model and constrains are given
(2) A new variant of GOA by adopting a dynamic opposite learning strategy is proposed to improve the exploitation ability of GOA and increase the possibility of finding the global optimum
(3) Various popular benchmarks from CEC 2014 and FJSP are used to evaluate the performance of DOLGOA by comparing with other classic algorithms

In this paper, we introduce the background in Section 1, then the problem description of FJSP is discussed in Section 2, and followed by the GOA algorithm and DOL strategy which are demonstrated in Section 3. Then, the DOLGOA is proposed in Section 4. The experiment and discussion are given in Section 5 with the comparison of other algorithms. Finally, Section 6 concludes the paper.

2. Problem Formulation

In the FJSP, we assume that there are N jobs to be processed and M usable machines in a workshop, where each job contains multiple operations, and each operation can be processed by a series of specified machines. Each machine can only serve one process at a time, the purpose of optimization is to arrange the machines according to the process requirements to minimize the maximal completion time $C_{\max}$. In order to express the problem more clearly, we use notations as follows:

i: the ith job
j: the jth operation of the corresponding job
k: the kth machine
p_i: the total number of operations for job i
B_{ijk}: beginning time of operation j of job i on machine k
P_{ijk}: processing time of operation j of job i on machine k
C_{ijk}: completion time of operation j of job i on machine k
C_i: completion time for job
$C_{\max}$: maximal completion time
Objective function: minimize $C_{\max}$

The constraints of FJSP can be expressed as follows:

(i) All machines are available at time zero, and all jobs can start at time zero:

$$\begin{aligned} B_{ijk} &\geq 0, \\ C_{ijk} &\geq 0. \end{aligned} \tag{1}$$

(ii) Processing time of each operation on the corresponding machine is definite, and the transmission time is ignored:

$$\begin{aligned} C_i &\geq C_{ij}, \quad \forall j \in [1, p_i], \\ C_{\max} &\geq C_i, \quad \forall i \in [1, N]. \end{aligned} \tag{2}$$

(iii) Jobs are independent of each other and cannot be inserted or cancelled by force:

$$\begin{gathered} S_{ijk} + P_{ijk} = C_{ijk}, \\ \sum_k S_{ijk} \geq \sum_k C_{i(j-1)k}. \end{gathered} \tag{3}$$

(iv) Each machine can only process one job at a time:

$$B_{ik} + P_{ik} \leq B_{(i+1)k}, \quad \forall j \in [1, p_i]. \tag{4}$$

(v) Each operation of a job can only be processed by one machine at a time:

$$B_{ij} + P_{ij} \leq B_{i(j+1)}, \quad \forall k \in [1, M]. \tag{5}$$

3. Algorithm Preliminaries

3.1. Grasshopper Optimization Algorithm. GOA optimization algorithm is a featured intelligent optimization algorithm based on population, which is inspired by the foraging behavior of grasshopper population. The grasshoppers' position constitutes the solution space, and the optimal grasshopper's position corresponds to the optimal solution. Grasshopper is always moving during the foraging process, so the position is constantly changing. During each iteration, the optimal position will be updated if a better position is found until the iteration termination condition is satisfied.

The mathematical model employed to simulate the swarming behavior of grasshoppers is shown as follows:

$$X_i = S_i + G_i + A_i, \tag{6}$$

where X_i defines the position of the ith grasshopper, S_i is social interaction, G_i is the influence of gravity on the ith grasshopper, and A_i is the influence of wind on the ith grasshopper. However, in order to provide random behavior, the equation can be written as $X_i = r_1 S_i + r_2 G_i + r_3 A_i$, where r_1, r_2, and r_3 is a random number between $[0, 1]$:

$$S_i = \sum_{j=1, j\neq i}^{N} s\left(d_{ij}\right)\widehat{d_{ij}}, \tag{7}$$

where d_{ij} is the distance between the ith and the jth grasshopper, calculated as $d_{ij} = |x_j - x_i|$, s is a function to define the strength of social forces, which is shown in equation (8), and $\widehat{d_{ij}} = ((x_j - x_i)/d_{ij})$ is a unit vector from the ith grasshopper to the jth grasshopper.

The function s, which defines the strength of social forces, is calculated as follows:

$$s(r) = f e^{(-r/1)} - e^{-r}, \tag{8}$$

where f is the strength of attraction and l is the scale of attraction length.

The G component in equation (6) is calculated as follows:

$$G_i = -g\widehat{e_g}, \tag{9}$$

where g is the gravitational constant and $\widehat{e_g}$ is a unit vector towards the center of Earth.

The A component in equation (6) is calculated as follows:

$$A_i = -u\widehat{e_w}, \tag{10}$$

where u is an offset constant and $\widehat{e_w}$ is the unit vector along the direction of wind.

Substituting S, G, and A in equation (6), this equation can be expressed as follows:

$$X_i = \sum_{j=1, j\neq i}^{N} s\left(\left|x_j - x_i\right|\right)\frac{x_j - x_i}{d_{ij}} - g\widehat{e_g} + u\widehat{e_w}, \tag{11}$$

where $s(r) = f e^{(-r/l)} - e^{-r}$ and N is the number of grasshoppers.

However, this mathematical model cannot be used directly to solve the optimization problems mainly because the grasshoppers quickly reach their comfort zone and the swarm does not converge to the specified point. In order to solve the optimization problem, the modified form of this equation is as follows:

$$X_i^d = c\left(\sum_{j=1, j\neq i}^{N} c\frac{\mathrm{ub}_d - \mathrm{lb}_d}{2} s\left(\left|x_j^d - x_i^d\right|\right)\frac{x_j - x_i}{d_{ij}}\right) + \widehat{T_d}, \tag{12}$$

where ub_d is the upper bound of the dth dimension, lb_d is the lower bound of the dth dimension, $s(r) = f e^{(-r/l)} - e^{-r}$, $\widehat{T_d}$ is the value of dth dimension of the target (best solution found so far), and c is a decreasing coefficient of comfort zone, repulsion zone, and attraction zone, as defined in equation (13). Note that S is almost similar to the component S in equation (6). However, we do not consider the gravity (no G component) and assume that the wind direction is always toward a target $(\widehat{T_d})$. According to equation (12), the next position of the grasshopper is determined according to its current position, the position of the target, and the position of all other grasshoppers:

$$c = c_{\max} - l\frac{c_{\max} - c_{\min}}{L}, \tag{13}$$

where $c_{\max}$ is the maximum value, $c_{\min}$ is the minimum value, l is the current iteration number, and L is the maximum iteration number. In this paper, we take 1 and 0.00001 for $c_{\max}$ and $c_{\min}$, respectively.

3.2. Oppositional-Based Learning (OBL). Opposition-based learning (OBL) [49] is one of the most successful learning strategies and is widely used in the effective learning phase to enhance the search capability of population-based algorithms. When finding the solution X of a given problem, OBL makes the candidate solution more likely to approach the optimal solution by simultaneously computing the current X and the inverse solution of X.

3.2.1. Opposite Number. X is a real number in $[a, b]$, where a and b are the boundaries of X. X^O is the opposite number of X, which can be defined as follows:

$$X^O = a + b - X. \tag{14}$$

3.2.2. Opposite Point. Assume that $X = (X_1, X_2, \ldots, X_D)$ is a point in a D-dimensional space, and $X_j \in [a_j, b_j]$, where $j = 1 : D$ and a_j and b_j are the low and high boundaries of the current population respectively, which change with iteration. A D-dimension opposite point is defined as follows:

$$X_j^{\mathrm{O}} = a_j + b_j - X_j, \quad j = 1\colon D. \tag{15}$$

3.3. Dynamic-Opposite Learning (DOL). On the basis of OBL strategy, many new ideas such as quasi-opposite-based learning (QOBL) [50] and quasi-reflection-based learning (QRBL) [51] are proposed, and the quasi-opposite number in QOBL and the quasi-reflection number in QRBL are both closer to the global optimum than the opposite number in OBL. However, if there is a local optimum in the search space, all strategies including OBL, QOBL, and QRBL tend to converge the search space to the local optimal location. Concerning this problem, a dynamic opposite learning (DOL) strategy is proposed in [37], which can improve the probability of convergence to the global optimum while avoiding falling into the local optimum.

In order to avoid the situation of falling into the local optimum in the space from the current number to the opposite number in OBL theory, a random opposite number X^{RO} was proposed, and $X^{\mathrm{RO}} = \mathrm{rand} * X^{O}, \mathrm{rand} \in [0, 1]$. We can see that X^{RO} can be not only between X and X^{O} but also greater or less than X or X^{O}. DOL strategy is formulated by randomly selecting a DOL number X^{DO} between X and X^{RO}. When $X^{\mathrm{RO}} \in [a, b]$, $X^{\mathrm{DO}} = X + \mathrm{rand}\,(X^{\mathrm{RO}} - X)$; otherwise, if X^{RO} exceeds the range of $[a, b]$, it is necessary to check whether the number obtained by DOL has a boundary: if $X + \mathrm{rand}\,(X^{\mathrm{RO}} - X) \in [a, b]$, X^{DO} is equal to $X + \mathrm{rand}\,(X^{\mathrm{RO}} - X)$; otherwise, X^{DO} should be reset as a random number between a and b. Although X^{RO} can enhance the diversity of the search space, the search space may shrink along with the iteration, which will lead to the deterioration of the exploitation capability. In light of this, a positive weighting factor w is used to balance the region and diversity of the search space. The mathematical model of DOL can be described as follows.

3.3.1. Dynamic Opposite Number. X is a real number in [a, b], where a and b are the boundaries of X. The dynamic opposite number X^{DO} can be defined as equation (16), where a and b are the boundaries and rand is a random number between 0 and 1. Moreover, w is a positive weighting factor, and X^{O} is the opposite number defined as equation (14):

$$X^{\mathrm{DO}} = X + w * \mathrm{rand} * \left(\mathrm{rand} * X^{O} - X\right). \tag{16}$$

3.3.2. Dynamic Opposite Point. $X = (X_1, X_2, \ldots, X_D)$ is a point in a D-dimensional space, and $X_j \in [a_j, b_j]$, where $j = 1 : D$ and a_j and b_j are the low and high boundaries of the current population, respectively, which change with iteration. A D-dimension dynamic opposite point is defined as follows:

$$X_j^{\mathrm{DO}} = X_j + w * \left(\mathrm{rand} * X_j^{\mathrm{O}} - X_j\right), \quad j = 1\colon D, \tag{17}$$

where rand is a random number between 0 and 1, w is a positive weighting factor, and X_j^{O} is the opposite point defined as equation (15).

3.3.3. DOL-Based Optimization. Assume that $X = (X_1, X_2, \ldots, X_D)$ is a point in a D-dimensional space, and $X_j \in [a_j, b_j]$, where $j = 1 : D$ and a_j and b_j are the low and high boundaries of the current population, respectively, which change with iteration. Dynamic opposite point $X^{\mathrm{DO}} = (X_1^{\mathrm{DO}}, X_2^{\mathrm{DO}}, \ldots, X_D^{\mathrm{DO}})$ is defined according to equation (17) and is updated by X in each generation. X should be replaced by X^{DO} if the fitness of X^{DO} is better than X. Otherwise, X stays the same. It is important to note that X_j^{DO} should be in $[a_j, b_j]$, otherwise, X_j^{DO} needs to be redefined as a random number in this interval.

4. DOL-Based GOA Algorithm

Conventional GOA algorithm has limitation in the exploitation ability, which calls for novel strategies in enhancing the solution diversity. In this section, an algorithm named DOLGOA is proposed, which applies DOL strategy to GOA algorithm in accelerating the convergence speed and preventing the algorithm from falling into local optimal. DOL strategy contributes to GOA mainly in two aspects, including population initialization and generation jumping.

4.1. DOL Population Initialization. The DOL initialization is shown as follows:

$$X_{i,j}^{\mathrm{DO}} = X_{i,j} + r1_i * \left(r2_i * \left(a_j + b_j - X_{i,j}\right) - X_{i,j}\right), \tag{18}$$

where $X_{i,j}$ is the randomly generated initial population and $X_{i,j}^{\mathrm{DO}}$ is the population obtained by DOL strategy. N is the population size, i represents the ith individual $(i = 1, 2, \ldots, Np)$, j represents the jth dimension $(j = 1, 2, \ldots, D)$, and $r1_i$ and $r2_i$ are two random numbers in [0, 1]. In DOL initialization, the weighting factor w is set as 1.

To ensure the effectiveness of DOL, boundary detection is also required:

$$X_{i,j}^{\mathrm{DO}} = \mathrm{rand}\left(a_j, b_j\right), \quad \text{if } X_{i,j}^{\mathrm{DO}} < a_j \text{ or } X_{i,j}^{\mathrm{DO}} > b_j. \tag{19}$$

In the initialization step, the optimal individual is selected from the new population consisting of $X_{i,j}$ and $X_{i,j}^{\mathrm{DO}}$.

4.2. DOL Generation Jumping. In each iteration, if the selected probability (random number between [0, 1]) is smaller than the jump rate (Jr), the population will be updated through DOL strategy. The DOL jump process is shown as follows:

$$X_{i,j}^{\mathrm{DO}} = X_{i,j} + w * r3_i * \left(r4_i * \left(a_j + b_j - X_{i,j}\right) - X_{i,j}\right), \tag{20}$$

where $X_{i,j}^{\mathrm{DO}}$ also needs to meet the boundary conditions. In order to lock the new candidate objects generated by DOL into a smaller search space, we dynamically update the boundary $[a_j, b_j]$ as follows:

$$a_j = \min\left(X_{i,j}\right),\quad b_j = \max\left(X_{i,j}\right). \tag{21}$$

In DOL generation jumping step, the optimal individuals are selected from the new population consisting of $X_{i,j}$ and $X_{i,j}^{\mathrm{DO}}$.

4.3. DOLGOA Algorithm Steps. When DOL population initialization and generation jumping are added, a new GOA variant, named dynamic opposite learning assisted GOA (DOLGOA), is proposed. In order to visualize the concrete algorithm, the steps of DOLGOA algorithm are described in Algorithm 1. The procedure of DOLGOA is shown in Figure 1.

4.4. Encoding and Decoding. To solve FJSP with DOLGOA, we encode the problem. In the FJSP, we define three variables X, T_1, and T_2, all of which are matrix with m rows and n columns, where m is the total number of jobs and n is the maximum number of operations.

The specific meaning of these three variables is shown as follows:

(i) X represents the machine number used in each operation, and the initial value of X is randomly selected from the set of alternative machines.

(ii) T_1 represents the beginning time of each process, and T_2 represents the completion time of each process. The values of T_1 and T_2 depend on X.

(iii) We treat each X as an individual and randomly generate Np individuals X at initialization time to form a population with Np number of individuals.

In addition, the decoding process can be expressed as follows:

(1) For each individual X, we calculate the beginning time T_1 and completion time T_2 of the first process for each job

(2) Then, calculate the arrangement of the remaining operations, where the beginning time of one operation is equal to the completion time of the previous one which is processed on the same machine, and the completion time of each operation T_2 is equal to the sum of the corresponding starting time T_1 and processing time

(3) Calculate the maximum completion time of each individual X, which is equal to the completion time of the last operation of the whole work

(4) With the goal of minimizing the maximum completion time, DOLGOA was used to select the optimal individual X, that is, the optimal machine allocation strategy

(5) Calculate the T_1 and T_2 corresponding to each procedure in the optimal strategy, and then draw the Gantt chart. The number i on the Gantt chart represents the ith job, and the tth i represents the tth process of the ith job

5. Experiment and Discussion

5.1. Benchmark Function. In this section, we use 23 benchmark test functions to evaluate the performance of DOLGOA. These 23 benchmark functions are derived from CEC2014 and are shown in Table 1. They include 3 unimodal functions, 13 multimodal functions, 6 hybrid functions, and 1 composite function. Unimodal functions are supposed to test the exploitation capability of DOLGOA because it has no local optimum in the searching space. On the contrary, multimodal functions have many local optimum and are used for exploration capability test of DOLGOA. In order to better mimic real search spaces, we also use hybrid functions and composite functions to test the performance of the algorithm.

5.2. Parameter Settings. The 23 test functions above were used to test and compare DOLGOA with four other algorithms, including GOA, GWO, TLBO, and Jaya. The parameter settings for these experiments are listed in Table 2.

For DOLGOA algorithm, weight factor w and jump rate Jr are two important parameters, and their analysis is shown in Table 3. We divide w and Jr into ten levels and analyze the effects of w and Jr on the optimization results by using orthogonal experiment [52]. For every pair of w and Jr, we run the corresponding algorithm 10 times and record the average results in Table 3. As shown in Table 3, when $w = 5$ and $\mathrm{Jr} = 0.9$, DOLGOA gets the best result. Therefore, $w = 5$ and $\mathrm{Jr} = 0.9$ are the most appropriate parameter settings.

In addition, the total number of individuals for all tests is 100. The function evaluations (FES) is set as 300,000, and all tests were executed 10 times.

5.3. Unimodal/Multimodal Test Functions and Their Analysis. Unimodal functions and multimodal functions are used to test the exploitation and exploration capability of DOLGOA. The mean and standard values of the result under these functions are shown in Table 4.

It can be seen that DOLGOA and GOA algorithms are superior to other algorithms in unimodal test functions F1–F3. Especially in terms of F1 and F3, DOLGOA is greatly improved on the basis of original GOA, which indicates that DOL strategy can improve the exploitation of individuals and help them converge to the global optimum in asymmetric space.

Multimodal functions contain multiple local optimum, which are used to test the exploration capability of DOLGOA under various local optimal conditions. It can be seen from F4–F16 that compared with other algorithms, GOA and DOLGOA algorithms show better exploration capability, especially on F5, F12, and F13, and DOLGOA has the best performance.

```
(1) Randomly generate an initial population $X$;
(2) for $i = 1; i \leq Np; i++$ do
(3)   $r1_i = \text{rand}(0,1)$, $r2_i = \text{rand}(0,1)$;
(4)   for $j = 1; j \leq D; j++$ do
(5)     $X_{ij}^{\text{DO}} = X_{ij} + r1_i * (r2_i * (a_j + b_j - X_{ij}) - X_{ij})$;
(6)     Check the boundaries;
(7)   end for
(8) end for
(9) Select $N$ number of the fittest individuals from $X \cup X^{\text{DO}}$;
(10) Set $G = 0$;
(11) while $G \leq$ maximal iteration do
(12)   Evaluate all learners by the fitness function $f(.)$;
(13)   if $G = 1$ then
(14)     Sort individuals by the fitness value to get the best grasshopper $X_{\text{best}}$ in the first population;
(15)   else
(16)     for $i = 1; i \leq Np; i++$ do
(17)       Update the position of the individuals $X_i$ according to update mechanism (equation (12));
(18)       Check the boundaries;
(19)       Evaluate the fitness values of the new individuals $X'$;
(20)       if $f(X_i') < f(X_{\text{best}})$ then
(21)         Replace $X_{\text{best}}$ with $X_i'$;
(22)       end if
(23)     end for
(24)   end if
(25)   $G++$;
(26)   if rand < Jr then
(27)     for $i = 1; i \leq Np; i++$ do
(28)       $r3_i = \text{rand}(0,1)$, $r4_i = \text{rand}(0,1)$;
(29)       for $j = 1; j \leq D; j++$ do
(30)         $a_j = \min(X_{i,j}), b_j = \max(X_{i,j})$;
(31)         $X_{i,j}^{\text{DO}} = X_{i,j} + w * r3_i * (r4_i * (a_j + b_j - X_{i,j}) - X_{i,j})$;
(32)         Check boundaries;
(33)       end for
(34)     end for
(35)     Select $N$ number of the fittest individuals from $X \cup X^{\text{DO}}$;
(36)     $G++$;
(37)   end if
(38) end while
```

Algorithm 1: The DOLGOA algorithm.

5.4. Analysis of Hybrid and Composition Test Functions. The hybrid function combines both unimodal and multimodal functions to better simulate the real search space. When dealing with the hybrid function, it needs to balance the exploitation ability and exploration ability, which requires higher requirements on the algorithm. The mean and standard values of all hybrid and composition test functions are shown in Table 5.

As can be seen from the table, DOLGOA performs very well in hybrid functions, especially in F17, F20, F21, and F22. On the composition function F23, the result of DOLGOA is not worse than other algorithms, which indicates that DOLGOA can well balance the exploitation and exploration capability in practical application.

5.5. Statistical Test Results. Two independent-sample T tests are applied to test whether there is a significant difference in the mean value of the two samples. The t values are shown in Table 6, and the P values are shown in Table 7, which is marked as "+" and "−".

Set 0.05 as the level of significance: $t(30)_{0.05} = 2.0423$. When $t < 2.0423$, it indicates $P > 0.05$, and P is marked as "+" in Table 7. In this case, the null hypothesis is eligible and accepted, which means there is no significant difference between the two algorithms. On the contrary, when $t > 2.0423$, $P < 0.05$ is indicated, and P is marked as "−" in Table 7; the null hypothesis is rejected, which means the two algorithms are significantly different. In Table 7, "Same" means the total number of DOLGOA that is not significantly different from other algorithms, and "Better" means the total number of DOLGOA that is significantly different from other algorithms.

From Tables 6 and 7, it can be seen that, for most test functions, DOLGOA shows significant difference from other algorithms, which means that DOL greatly improves the performance of GOA.

5.6. Analysis of Convergence. The convergence trends of all algorithms for unimodal and multimodal test functions are

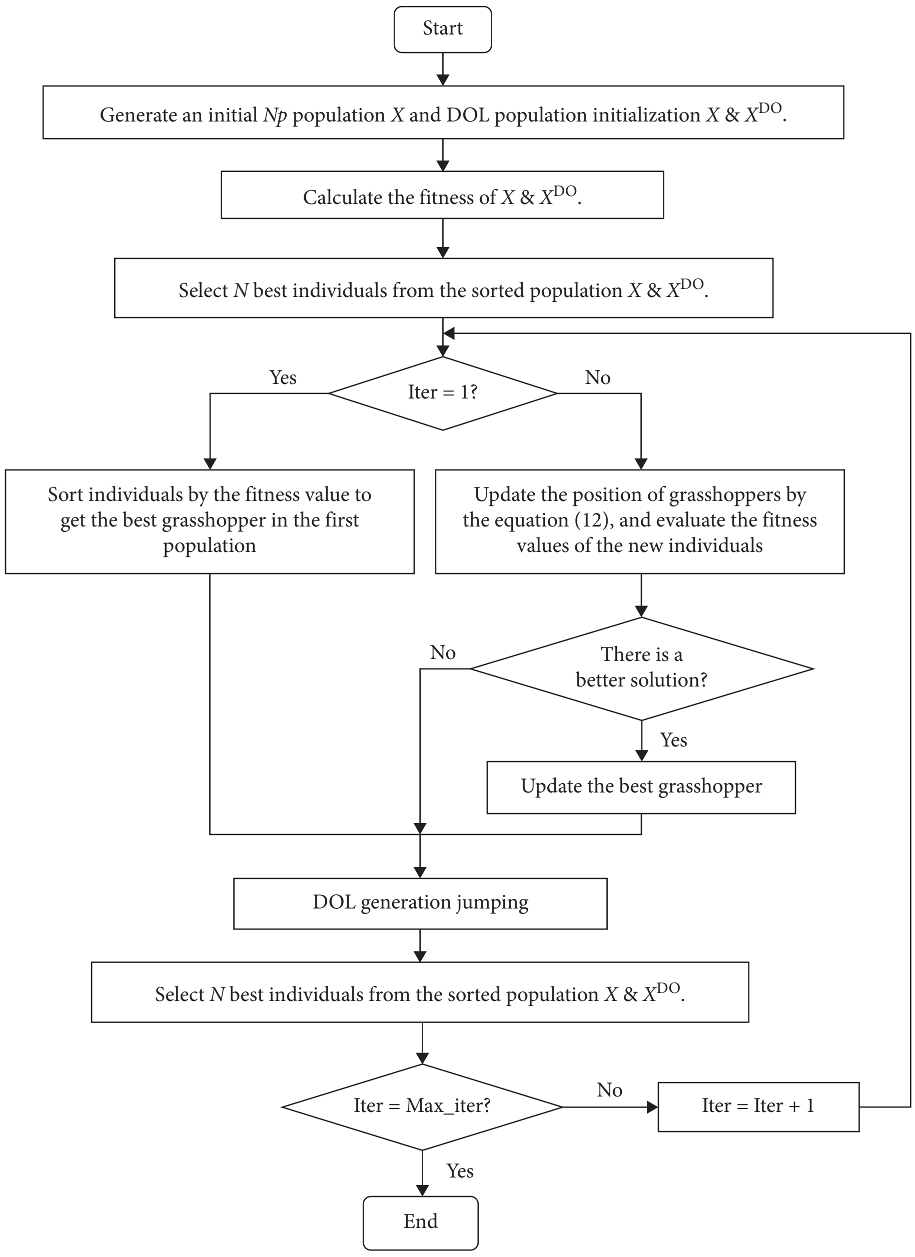

Start
Generate an initial Np population X and DOL population initialization X & X^{DO}.
Calculate the fitness of X & X^{DO}.
Select N best individuals from the sorted population X & X^{DO}.
Iter = 1?
Yes
No
Sort individuals by the fitness value to get the best grasshopper in the first population
Update the position of grasshoppers by the equation (12), and evaluate the fitness values of the new individuals
There is a better solution?
No
Yes
Update the best grasshopper
DOL generation jumping
Select N best individuals from the sorted population X & X^{DO}.
Iter = Max_iter?
No
Iter = Iter + 1
Yes
End

Figure 1: The flowchart of DOLGOA algorithm.

shown in Figures 2 and 3, respectively, and the convergence trends of all algorithms for hybrid function and composite function are shown in Figure 4.

As can be seen from the figure, the convergence result of DOLGOA is the best for most of the test functions, which indicates that DOLGOA has good exploration capability to avoid falling into the local optimum. The convergence curve of DOLGOA eventually tends to the optimal value, thanks to the exploitation capability of the DOL strategy.

5.7. Application to FJSP. In addition to the test on numerical benchmark functions, DOLGOA is adopted to solve the FJSP to further evaluate its performance. The purpose of FJSP in this paper is to minimize the maximal completion time. We analyze the performance of DOLGOA by comparing with GOA, PSO, Jaya, DE , GWO [53–56], HTS/TA, ITS, and ISA. Some of the data of these experiments are adopted from the literature [57]. These algorithms are implemented on 21 different problems, which are classified into three classes: small (SFJS01–SFJS10), medium (MFJS01–MFJS10), and large ((LFJS01)) size FJSP problems. The data of small and medium size experiments is adopted from the literature [57], and the data of the large size experiment is adopted from the literature [58]. Table 1 shows the parameter settings of these optimization algorithms. To eliminate the randomness, each problem implements 10 independent runs. Table 8 shows the experimental results of these algorithms in small and medium size experiments, and the results with * are the best solution results of the corresponding problem.

Table 1: Benchmark functions provided by CEC 2014 test set.

Label	Functions	Optimum	Dims	Character
F1	High conditioned elliptic	100	30	Unimodal
F2	Bent cigar	200	30	Unimodal
F3	Discus	300	30	Unimodal
F4	Rosenbrock	400	30	Multimodal
F5	Ackely	500	30	Multimodal
F6	Weierstrass	600	30	Multimodal
F7	Griewank	700	30	Multimodal
F8	Rastrigin	800	30	Multimodal
F9	Rotated rastrigin	900	30	Multimodal
F10	Schwefel	1000	30	Multimodal
F11	Rotated schwefel	1100	30	Multimodal
F12	Katsuura	1200	30	Multimodal
F13	HappyCat	1300	30	Multimodal
F14	HGBat	1400	30	Multimodal
F15	Expanded griewank plus rosenbrock	1500	30	Multimodal
F16	Expanded scaffer	1600	30	Multimodal
F17	HF1	1700	30	Hybrid
F18	HF2	1800	30	Hybrid
F19	HF3	1900	30	Hybrid
F20	HF4	2000	30	Hybrid
F21	HF5	2100	30	Hybrid
F22	HF6	2200	30	Hybrid
F23	CF1	2300	30	Composition

Table 2: The parameters settings of optimization algorithms.

Parameters	Value
Size of population	100
Total generation number for test functions	3000
Total generation number for FJSP	200
Times conducting the experiment	10
$c_{\max}$ of GOA (maximum value)	1
$c_{\min}$ of GOA (minimum value)	0.00001
c of GOA (decreasing coefficient)	$c_{\max} - \text{iter} * ((c_{\max} - c_{\min})/\text{Max}_{\text{iter}})$
Jr of DOLGOA	1
w of DOLGOA	4
Inerita weight of PSO	$0.9 - 0.4 * (G/G_m)$
Scale factor of DE	0.5
Crossover constant of DE	0.5

Table 3: The analysis of w and Jr.

DOLGOA	$w = 1$	$w = 2$	$w = 3$	$w = 4$	$\boldsymbol{w = 5}$	$w = 6$	$w = 7$	$w = 8$	$w = 9$	$w = 10$
Jr = 0.1	$5.68E+06$	$8.08E+06$	$9.04E+06$	$1.13E+07$	$4.98E+06$	$3.16E+06$	$8.49E+06$	$8.23E+06$	$7.78E+06$	$1.06E+07$
Jr = 0.2	$1.39E+07$	$5.65E+06$	$1.06E+07$	$8.21E+06$	$4.33E+06$	$4.56E+06$	$2.09E+06$	$5.58E+06$	$6.16E+06$	$3.02E+06$
Jr = 0.3	$2.25E+06$	$2.59E+06$	$8.37E+06$	$6.23E+06$	$9.51E+06$	$4.18E+06$	$4.69E+06$	$3.14E+06$	$4.63E+06$	$2.28E+06$
Jr = 0.4	$6.06E+06$	$2.62E+06$	$6.35E+06$	$1.98E+06$	$7.11E+06$	$5.48E+06$	$6.07E+06$	$5.37E+06$	$7.52E+06$	$8.97E+06$
Jr = 0.5	$4.69E+06$	$2.45E+06$	$5.14E+06$	$3.53E+06$	$4.22E+06$	$1.84E+06$	$5.63E+06$	$7.82E+06$	$4.79E+06$	$6.35E+06$
Jr = 0.6	$5.62E+06$	$1.17E+07$	$7.51E+06$	$4.16E+06$	$3.17E+06$	$2.59E+06$	$3.19E+06$	$1.82E+06$	$1.55E+06$	$7.45E+06$
Jr = 0.7	$6.57E+06$	$3.74E+06$	$5.18E+06$	$2.25E+06$	$3.91E+06$	$9.26E+06$	$6.98E+06$	$3.54E+06$	$3.26E+06$	$6.74E+06$
Jr = 0.8	$1.54E+06$	$7.66E+06$	$8.43E+06$	$3.67E+06$	$5.07E+06$	$2.82E+06$	$2.10E+06$	$3.08E+06$	$3.60E+06$	$2.35E+06$
Jr = **0.9**	$3.90E+06$	$6.37E+06$	$3.47E+06$	$3.90E+06$	$\mathbf{1.11E+06}$	$4.90E+06$	$1.56E+06$	$8.06E+06$	$6.84E+06$	$1.60E+06$
Jr = 1	$2.40E+06$	$3.34E+06$	$4.59E+06$	$2.57E+06$	$2.23E+06$	$1.40E+06$	$1.94E+06$	$1.55E+06$	$3.38E+06$	$4.14E+06$

It can be seen that DOLGOA obtains the best results for 14 problems. Although DOLGOA does not get the best results in other medium size problems, the difference is not significant from the best results. From Table 8, we can see that almost all algorithms can get the best result for small size FJSP, but as the scale of the problem gets larger, the

Table 4: The mean and standard value of unimodal/multimodal test functions.

Algorithms		DOLGOA	GOA	GWO	TLBO	Jaya
F1	Mean	**1.11E + 06**	1.17E + 07	5.71E + 07	1.27E + 08	4.70E + 06
	Std	**4.32E + 06**	9.91E + 06	3.26E + 07	8.84E + 07	1.27E + 06
F2	Mean	5.66E + 03	**3.79E + 03**	1.73E + 09	2.25E + 10	4.73E + 08
	Std	4.38E + 03	**2.21E + 03**	9.04E + 08	6.61E + 09	1.43E + 08
F3	Mean	**2.75E − 02**	4.30E + 02	2.13E + 04	3.63E + 04	2.87E + 04
	Std	**2.28E − 02**	1.89E + 02	7.27E + 03	5.82E + 03	4.28E + 03
F4	Mean	1.05E + 02	**1.04E + 02**	3.09E + 02	2.57E + 03	2.12E + 02
	Std	**6.07E + 01**	6.55E + 01	1.84E + 02	1.53E + 03	**1.40E + 01**
F5	Mean	**2.00E + 01**	**2.00E + 01**	2.09E + 01	2.04E + 01	2.09E + 01
	Std	9.33E − 04	**1.95E − 06**	6.04E − 02	1.43E − 01	5.19E − 02
F6	Mean	1.59E + 01	1.71E + 01	**1.29E + 01**	2.69E + 01	2.24E + 01
	Std	**2.09E + 00**	5.52E + 00	3.85E + 00	3.89E + 00	4.18E + 00
F7	Mean	2.91E − 02	**1.32E − 02**	7.38E + 00	2.61E + 02	5.68E + 00
	Std	3.15E − 02	**1.51E − 02**	4.22E + 00	8.86E + 01	1.76E + 00
F8	Mean	1.10E + 02	1.10E + 02	**8.41E + 01**	1.91E + 02	1.83E + 02
	Std	3.50E + 01	2.65E + 01	**1.48E + 01**	3.18E + 01	**1.45E + 01**
F9	Mean	1.06E + 02	**6.50E + 01**	7.82E + 01	2.16E + 02	1.84E + 02
	Std	1.95E + 01	2.40E + 01	**8.74E − 02**	2.21E + 01	1.60E + 01
F10	Mean	2.45E + 03	3.49E + 03	**2.09E + 03**	5.02E + 03	5.03E + 03
	Std	6.29E + 02	**3.02E + 02**	6.41E + 02	3.65E + 02	8.82E + 02
F11	Mean	2.85E + 03	2.96E + 03	**2.09E + 03**	5.04E + 03	6.23E + 03
	Std	6.49E + 02	**5.70E + 02**	6.25E + 02	3.71E + 02	4.48E + 02
F12	Mean	**1.65E − 01**	5.26E − 01	1.87E + 00	1.13E + 00	2.39E + 00
	Std	**7.88E − 02**	2.47E − 01	1.03E + 00	3.46E − 01	**2.71E − 01**
F13	Mean	**3.92E − 01**	4.21E − 01	4.21E − 01	3.88E + 00	5.89E − 01
	Std	9.92E − 02	8.89E − 02	**7.52E − 02**	4.79E − 01	1.01E − 01
F14	Mean	2.29E − 01	**2.21E − 01**	1.03E + 01	9.67E + 01	2.79E − 01
	Std	6.40E − 02	**2.98E − 02**	1.62E + 01	1.52E + 01	2.10E − 02
F15	Mean	7.03E + 00	**6.87E + 00**	2.21E + 01	8.24E + 03	2.20E + 01
	Std	2.04E + 00	**9.11E − 01**	9.62E + 00	8.24E + 03	1.75E + 00
F16	Mean	1.18E + 01	1.15E + 01	**1.09E + 01**	1.15E + 01	1.26E + 01
	Std	5.11E − 01	6.85E − 01	**4.16E − 01**	3.96E − 01	2.09E − 01
Best num		5	7	5	0	0

Table 5: The mean and standard value of hybrid test functions.

Algorithms		DOLGOA	GOA	GWO	TLBO	Jaya
F17	Mean	**3.89E + 04**	8.39E + 05	9.13E + 05	7.88E + 05	2.79E + 05
	Std	**2.11E + 04**	1.06E + 06	1.03E + 06	6.08E + 05	1.38E + 05
F18	Mean	2.95E + 03	5.90E + 03	8.69E + 06	**2.17E + 03**	7.73E + 05
	Std	2.94E + 03	3.81E + 03	1.56E + 07	**2.92E + 03**	3.06E + 05
F19	Mean	1.50E + 01	1.37E + 01	4.51E + 01	5.37E + 01	**1.20E + 01**
	Std	1.97E + 00	4.99E + 00	2.33E + 01	2.77E + 01	**1.39E + 00**
F20	Mean	**1.71E + 02**	3.10E + 02	1.06E + 04	5.49E + 03	1.04E + 03
	Std	**2.67E + 01**	1.00E + 02	2.17E + 03	6.47E + 03	9.24E + 02
F21	Mean	**2.05E + 04**	1.30E + 05	2.51E + 05	2.51E + 04	4.14E + 04
	Std	**8.65E + 03**	7.83E + 04	3.33E + 05	1.27E + 04	9.43E + 03
F22	Mean	**2.40E + 02**	4.64E + 02	2.49E + 02	5.03E + 02	3.85E + 02
	Std	**1.48E + 02**	2.25E + 02	1.01E + 02	1.44E + 02	7.13E + 01
F23	Mean	**2.00E + 02**	3.21E + 02	3.32E + 02	**2.00E + 02**	3.25E + 02
	Std	**1.18E − 05**	4.80E + 00	8.01E + 00	2.03E − 13	2.94E + 00
Best num		5	0	0	2	1

advantage of DOLGOA becomes more apparent. For the 10 medium size FJSPs, DOLGOA obtains the best results on four of them, and the results are much better than GOA. This demonstrates that DOL strategy improves the exploitation capability of the algorithm, which makes DOLGOA more effective for solving FJSP. Figure 5 shows the mean convergence curve of the makespan in MFJS03 of the six algorithms in 10 separate runs. Figure 6 shows the convergence curve of the makespan in MFJS09 of six algorithms in 10 independent runs. It can be seen that DOLGOA obtains the best result. Figures 7 and 8 illustrate the Gantt charts of the optimal solutions obtained by DOLGOA in a

Table 6: The *t* values of DOLGOA and other algorithms.

		GOA	GWO	TLBO	Jaya
DOLGOA	F1	$1.56E+01$	$1.20E+02$	$2.83E+02$	$7.32E-01$
	F2	$4.26E+00$	$3.95E+06$	$5.14E+07$	$1.08E+06$
	F3	$1.89E+05$	$9.35E+06$	$1.60E+07$	$1.26E+07$
	F4	$2.91E-01$	$3.35E+01$	$4.06E+02$	$1.75E+02$
	F5	$7.37E-01$	$1.00E+04$	$4.45E+03$	$9.86E+03$
	F6	$5.73E+00$	$1.39E+01$	$5.29E+01$	$3.11E+02$
	F7	$5.05E+00$	$2.33E+03$	$8.29E+04$	$1.79E+03$
	F8	$1.89E-02$	$7.39E+00$	$2.04E+01$	$2.08E+01$
DOLMFO	F9	$2.14E+01$	$1.46E+01$	$5.61E+01$	$3.94E+01$
	F10	$1.65E+01$	$5.80E+00$	$4.08E+01$	$4.10E+01$
	F11	$1.72E+00$	$1.17E+01$	$3.37E+01$	$5.20E+01$
	F12	$4.58E+01$	$2.17E+02$	$1.22E+02$	$2.83E+02$
	F13	$3.02E+00$	$2.94E+00$	$3.52E+02$	$1.99E+01$
	F14	$1.21E+00$	$1.58E+03$	$1.51E+04$	$7.86E+00$
	F15	$7.55E-01$	$7.37E+01$	$4.03E+04$	$7.33E+01$
	F16	$6.32E+00$	$1.87E+01$	$5.79E+00$	$1.45E+01$
	F17	$3.78E+02$	$4.13E+02$	$3.54E+02$	$1.14E+02$
	F18	$1.00E+01$	$2.95E+04$	$2.67E+00$	$2.61E+03$
	F19	$6.96E+00$	$1.52E+02$	$1.95E+02$	$1.54E+01$
	F20	$5.18E+01$	$3.92E+03$	$1.98E+03$	$3.27E+02$
	F21	$1.26E+02$	$2.67E+02$	$5.25E+00$	$2.41E+01$
	F22	$1.51E+01$	$5.88E-01$	$1.77E+01$	$9.75E+00$
	F23	$1.02E+08$	$1.12E+08$	$1.50E+01$	$1.06E+08$

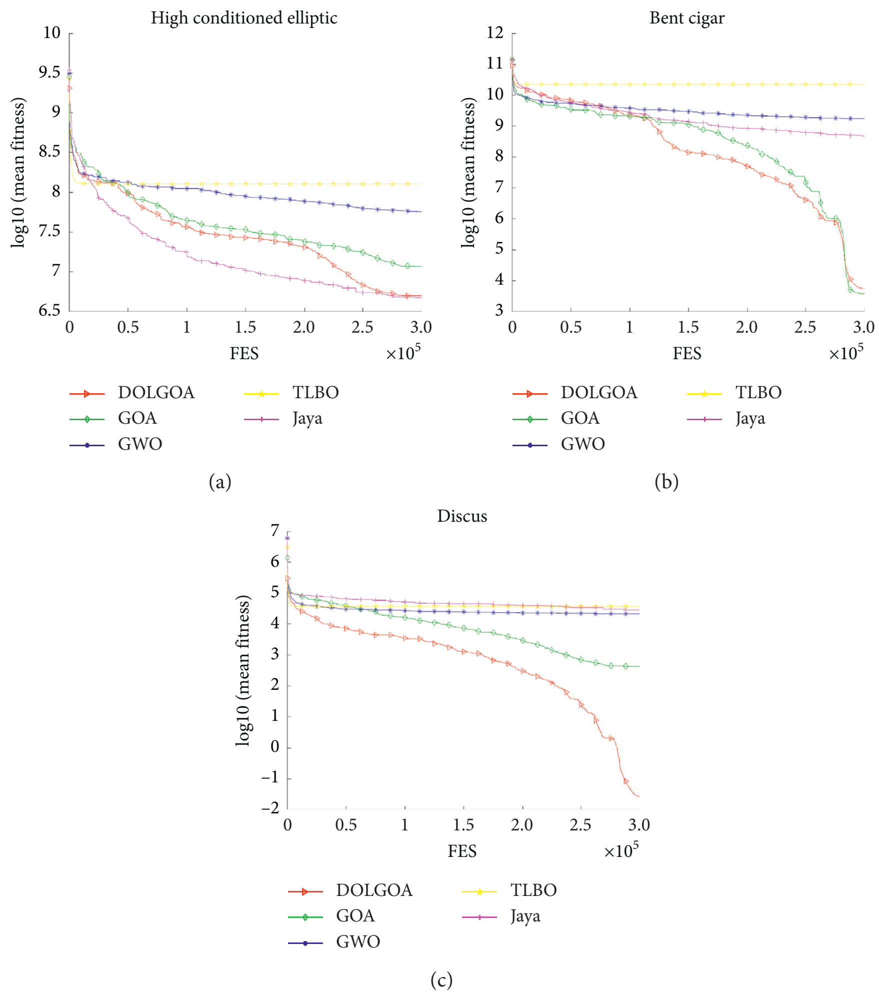


Figure 2: The convergence trends of all algorithms on unimodal functions.

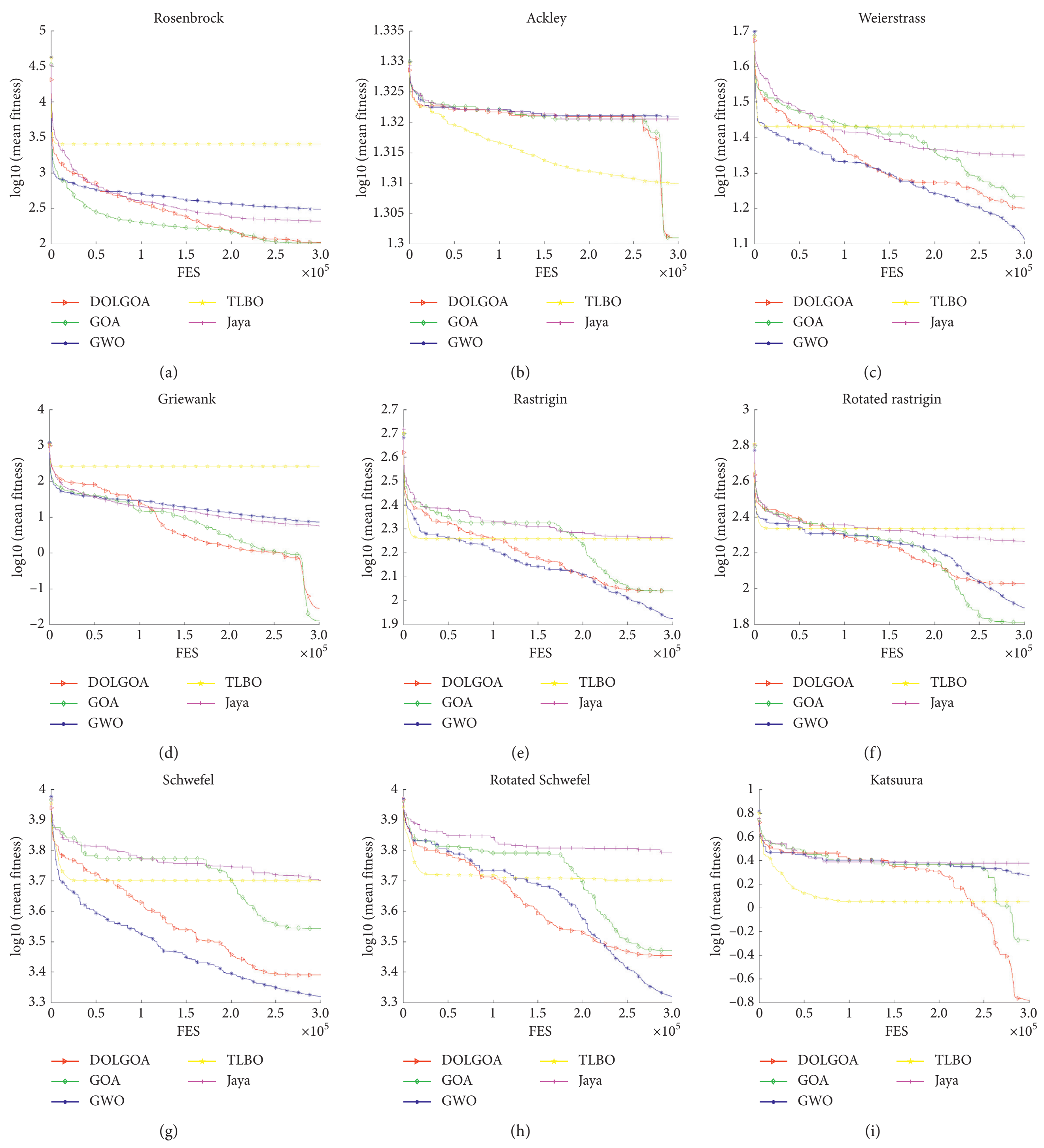
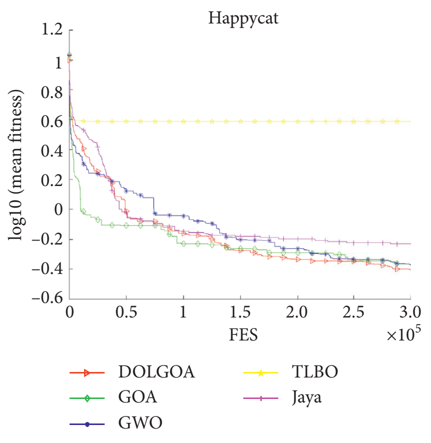
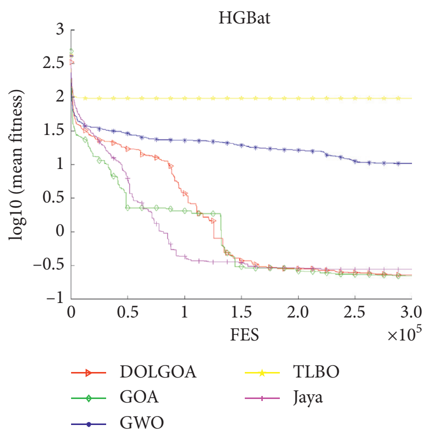


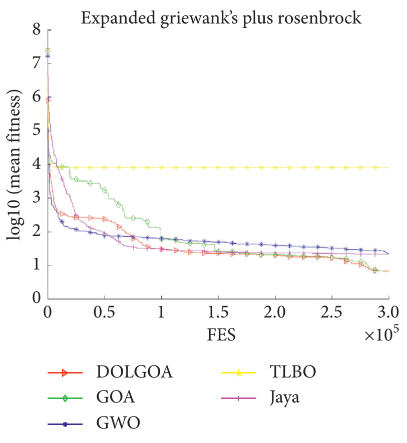
FIGURE 3: Continued.



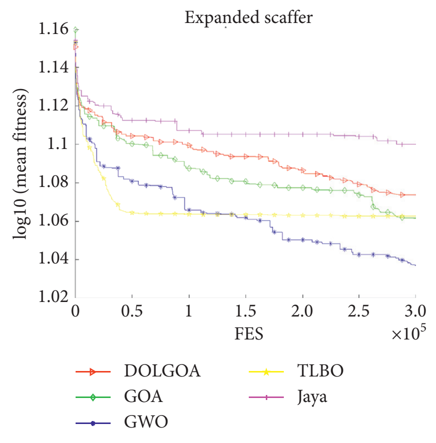
(j)



(k)

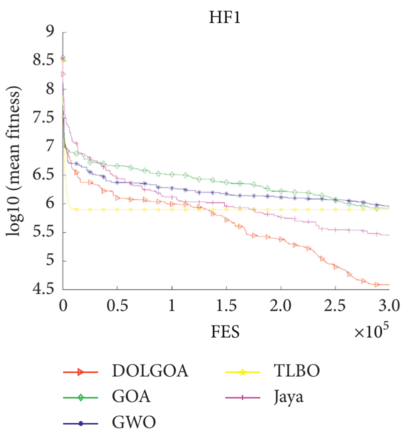


(l)

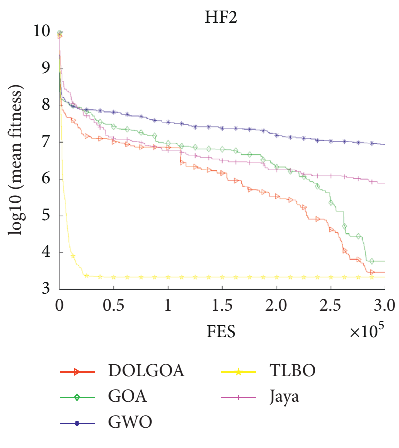


(m)

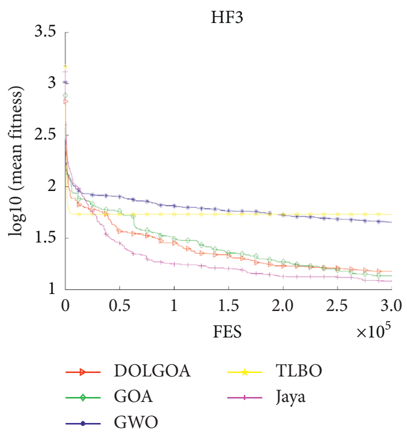
Figure 3: The convergence trends of all algorithms on multimodal functions.



(a)



(b)



(c)

Figure 4: Continued.

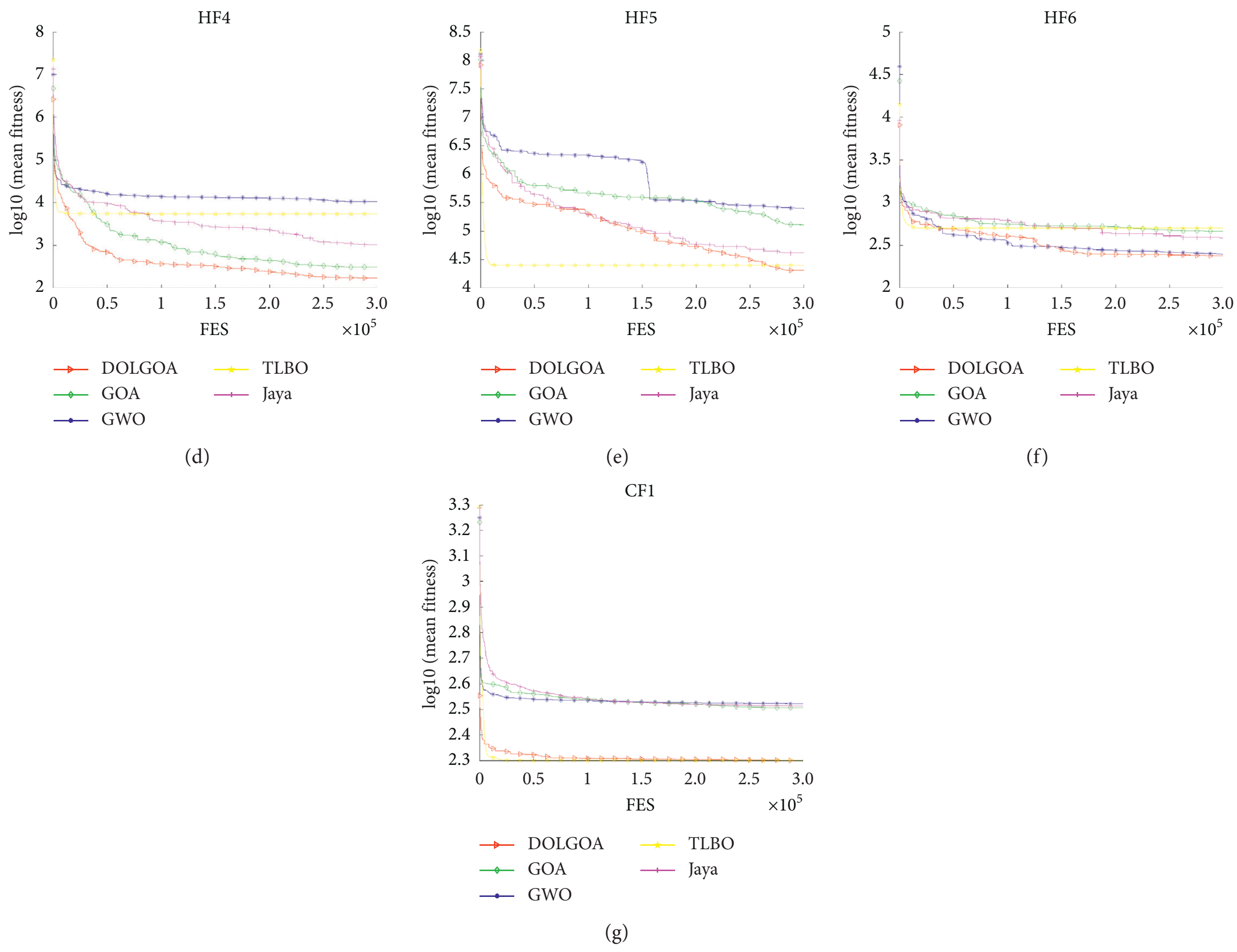


Figure 4: The convergence trends of all algorithms on hybrid and composition functions.

Table 7: The P values of DOLGOA and other algorithms.

		GOA	GWO	TLBO	Jaya
DOLGOA	F1	–	–	–	+
	F2	–	–	–	–
	F3	–	–	–	–
	F4	+	–	–	–
	F5	–	–	–	–
	F6	–	–	–	–
	F7	–	–	–	–
	F8	+	–	–	–
	F9	–	–	–	–
	F10	–	–	–	–
	F11	+	–	–	–
	F12	–	–	–	–
DOLMFO	F13	–	–	–	–
	F14	+	–	–	–
	F15	+	–	–	–
	F16	–	–	–	–
	F17	–	–	–	–
	F18	–	–	–	–
	F19	–	–	–	–
	F20	–	–	–	–
	F21	–	–	–	–
	F22	–	+	–	–
	F23	–	–	–	–
Better		18	22	23	22
Same		5	1	0	1

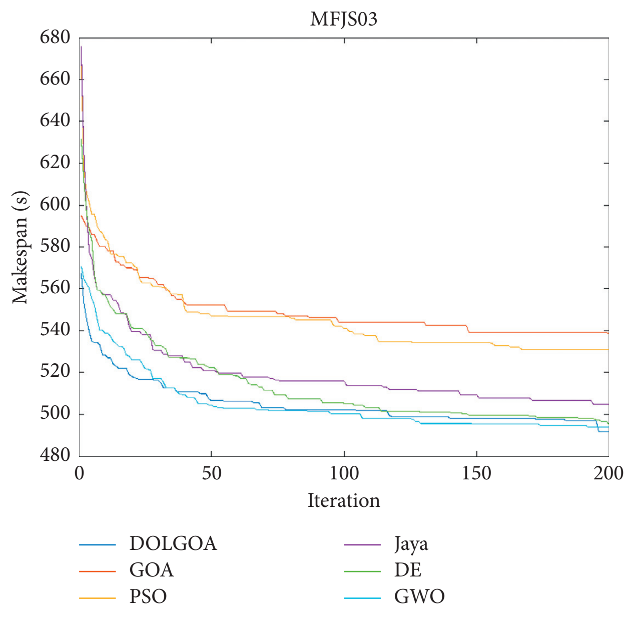


Figure 5: Convergence results of MFJS03 for all the compared algorithms.

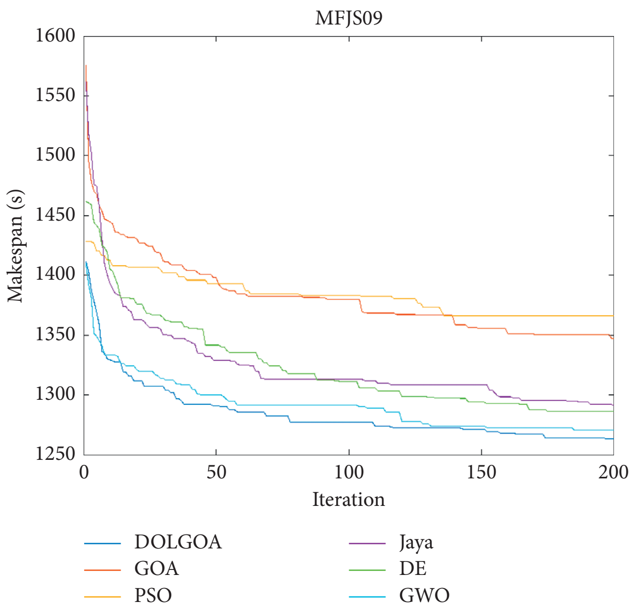


Figure 6: Convergence results of MFJS09 for all the compared algorithms.

featured run for problem MFJS03 and MFJS10, respectively. We can see that the shortest completion time solved by DOLGOA algorithm is 489 and 1517 for MFJS03 and MFJS10, respectively. Figure 9 shows the convergence curve of the average results of ten runs obtained by DOLGOA for the large size experiments. In addition, Figure 10 shows the Gantt chart of the optimal solutions obtained by DOLGOA for the large size experiments. In the large size problem LFJS01, there are 15 jobs to be processed by 10 machines in a workshop, where each job consists of 2–4 processes and each process can only be completed by several specified machines. It can be seen that DOLGOA can assign a series of optional machines to a specified process to obtain the best solution. As shown in Figure 7, the optimal solution obtained by DOLGOA for LFJS01 is 31. In Table 9, we list the computational time of these 6 algorithms to solve FJSP, and all the simulations run on an Intel(R) Core(TM) i5-9400F CPU@ 2.90 GHz PC and the Matlab(R) 2019a software platform.

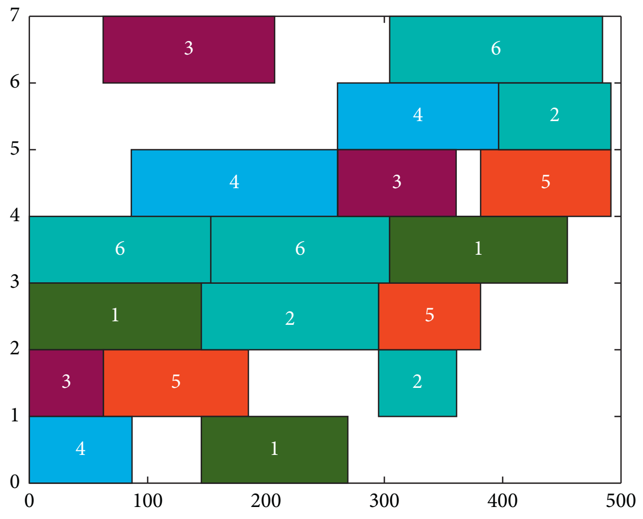


Figure 7: Gantt chart of an optimal solution to MFJS03 obtained by DOLGOA.

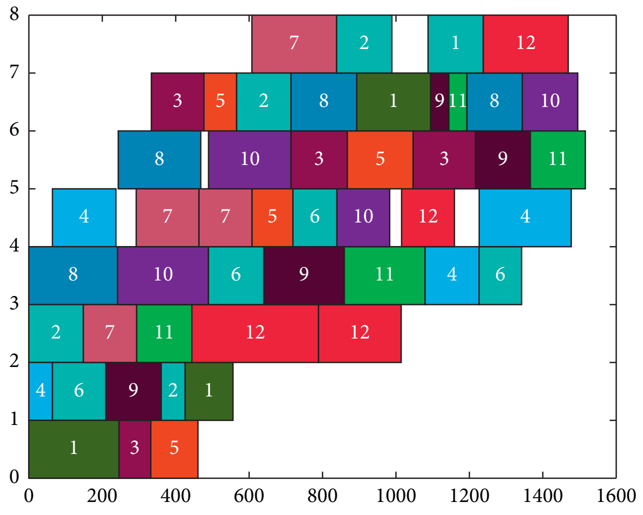


Figure 8: Gantt chart of an optimal solution to MFJS10 obtained by DOLGOA.

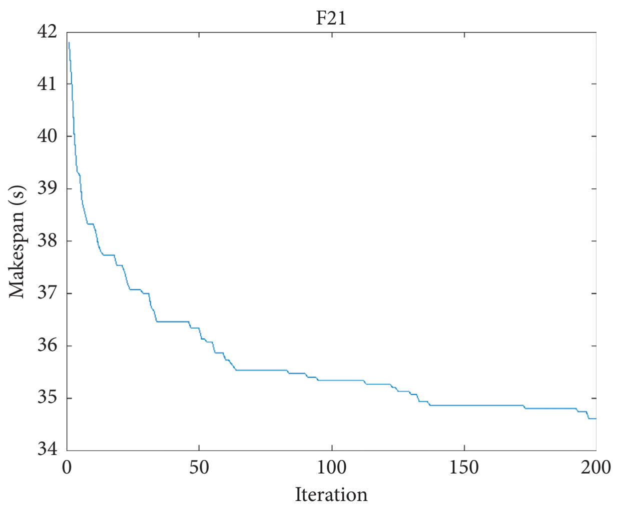


Figure 9: Convergence results on LFJS01.

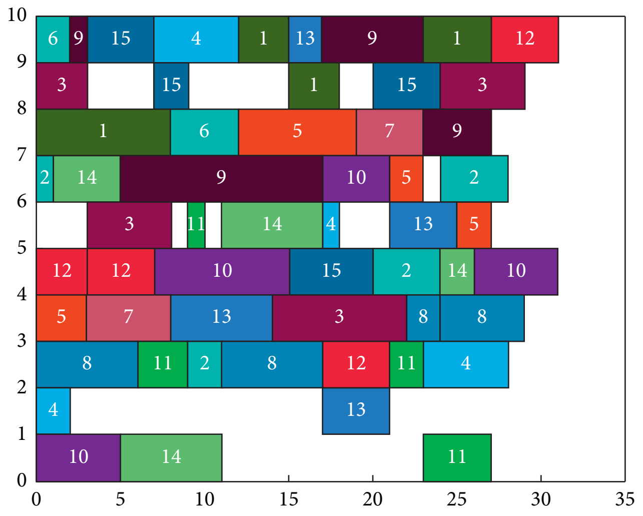


Figure 10: Gantt chart on LFJS01.

Table 8: The statistical results obtained by algorithms for solving FJSP.

Algorithms	DOLGOA	GOA	PSO	Jaya	DE	GWO	HTS/TA	ITS	ISA
MFJS01	481	482	484	490	469*	481	503	584	518
MFJS02	456*	463	466	457	456*	467	494	544	494
MFJS03	491*	538	531	505	495	494	540	606	611
MFJS04	653	651	646	620	600	562*	660	870	798
MFJS05	593	613	614	558	538	508*	633	729	706
MFJS06	643*	749	756	708	682	651	741	876	889
MFJS07	1093	1092	1100	1009	990	933*	999	1127	1307
MFJS08	997*	1100	1108	1022	1025	1009	1000	1352	1437
MFJS09	1263	1347	1366	1291	1286	1271	1158*	1219	1218
MFJS10	1517	1567	1592	1458*	1488	1500	1511	1737	1733
SFJS01	66*	66*	66*	66*	66*	66*	66*	66*	66*
SFJS02	107*	107*	107*	107*	107*	107*	107*	107*	107*
SFJS03	221*	221*	221*	221*	221*	221*	221*	221*	231
SFJS04	355*	355*	355*	355*	355*	355*	355*	390	375
SFJS05	119*	119*	119*	119*	119*	119*	119*	137	137
SFJS06	320*	320*	320*	320*	320*	320*	320*	320*	336
SFJS07	397*	397*	397*	397*	397*	397*	397*	397*	397*
SFJS08	253*	253*	253*	253*	253*	253*	253*	253*	254
SFJS09	210*	210*	210*	210*	210*	210*	210*	218	228
SFJS10	533*	533*	533*	533*	533*	533*	533*	624	570
Best num	14	10	10	11	12	13	11	6	3

Table 9: The CPU time periods obtained by algorithms for solving FJSP.

Algorithms	DOLGOA	GOA	PSO	Jaya	DE	GWO
MFJS01	6.89	7.53	6.88	4.19	4.18	3.02
MFJS02	6.54	7.81	6.66	4.75	4.26	3.03
MFJS03	7.18	8.20	7.86	5.34	4.86	3.38
MFJS04	7.75	8.52	8.73	5.91	5.45	3.82
MFJS05	7.82	8.51	8.63	6.00	5.50	3.85
MFJS06	8.55	9.06	9.92	6.68	6.16	4.34
MFJS07	10.31	10.57	12.85	8.33	8.37	5.74
MFJS08	11.13	11.16	14.30	8.87	9.02	6.25
MFJS09	12.68*	12.28	16.66	10.44	10.57	7.46
MFJS10	13.68	12.99	18.24	10.88	11.34	7.85

Table 9: Continued.

Algorithms	DOLGOA	GOA	PSO	Jaya	DE	GWO
SFJS01	3.75	5.70	2.23	1.19	1.36	0.97
SFJS02	3.59	5.32	2.21	1.16	1.34	1.00
SFJS03	4.23	5.58	2.84	1.53	1.67	1.33
SFJS04	4.19	5.72	2.98	1.53	1.75	1.33
SFJS05	4.21	5.81	3.06	4.19	1.70	1.31
SFJS06	5.02	6.22	4.33	4.75	2.52	1.95
SFJS07	5.31	6.53	4.67	5.34	2.86	1.99
SFJS08	5.13	6.65	4.49	5.91	2.67	1.91
SFJS09	5.08	6.51	4.45	6.00	2.56	1.95
SFJS10	5.78	6.95	5.89	6.68	3.35	2.51

6. Conclusion

In this paper, a new variant of GOA named DOLGOA is proposed, which embeds DOL strategy into GOA to prevent it from falling into local optimum. The asymmetric and dynamic nature of DOL helps DOLGOA possess better exploitation and exploration capabilities than GOA. The performance of the proposed DOLGOA algorithm is evaluated in 23 benchmarks from CEC2014, and the proposed algorithm is applied in 21 FJSP problems. Comprehensive results show that DOLGOA behaves the best when solving numerical benchmarks and is well-performed for FJSP problems in different scales. The proposed algorithm is promising to be applied in solving various engineering optimization problems.

Data Availability

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

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research work was supported by the National Key Research and Development Project under Grant 2018YFB1700500.

References

[1] C. Jin and H. Wu, "A neural networks training algorithm based on adaptive genetic algorithm," *Control & Automation*, vol. 10, no. 1, pp. 49–51, 2005.

[2] A. Likas, D. A. Karras, and I. E. Lagaris, "Neural network training and simulation using a multidimensional optimization system," *International Journal of Computer Mathematics*, vol. 67, no. 1-2, 1998.

[3] O. Castillo, L. Trujillo, and P. Melin, "Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots," *Soft Computing*, vol. 11, no. 3, pp. 269–279, 2007.

[4] M. L. Cummings, J. J. Marquez, and N. Roy, "Human-automated path planning optimization and decision support," *International Journal of Human-Computer Studies*, vol. 70, no. 2, pp. 116–128, 2012.

[5] T. Bhatia and L. T. Biegler, "Dynamic optimization in the design and scheduling of multiproduct batch plants," *Industrial & Engineering Chemistry Research*, vol. 35, no. 7, pp. 2234–2246, 1996.

[6] V. V. Toropov, L. F. Alvarez, and O. M. Querin, "Applications of GA and GP to industrial design optimization and inverse problems," *Advances of Soft Computing in Engineering*, vol. 512, pp. 133–189, 2010.

[7] D. Bongartz and A. Mitsos, "Deterministic global optimization of process flowsheets in a reduced space using Mccormick relaxations," *Journal of Global Optimization*, vol. 69, no. 4, pp. 761–796, 2017.

[8] J. W. Gillard, "Deterministic global optimization: an introduction to the diagonal approach," *Optimization Methods & Software*, vol. 34, no. 1, pp. 1-2, 2018.

[9] A. M. Niziolek, O. Onel, and C. A. Floudas, "Municipal solid waste to liquid transportation fuels, olefins, and aromatics: process synthesis and deterministic global optimization," *Computers & Chemical Engineering*, vol. 102, pp. 169–187, 2017.

[10] Y. D. Sergeyev and D. E. Kvasov, "Deterministic global optimization," *Springer Optimization & Its Applications*, vol. 37, Springer, Berlin, Germany.

[11] D. Chen, F. Zou, Z. Li, J. Wang, and S. Li, "An improved teaching-learning-based optimization algorithm for solving global optimization problem," *Information Sciences*, vol. 297, pp. 171–190, 2015.

[12] S. Das and P.N. Suganthan, "Differential evolution: a survey of the state-of-the-art," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[13] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95—International Conference on Neural Networks*, Perth, Australia, 2002.

[14] J. J. Q. Yu and V. O. K. Li, "A social spider algorithm for global optimization," *Applied Soft Computing*, vol. 30, pp. 614–627, 2015.

[15] O. Serrano-Prez, M. G. Villarreal-Cervantes, J. C. Gonzlez-Robles, and A. Rodrguez-Molina, "Meta-heuristic algorithms for the control tuning of omnidirectional mobile robots," *Engineering Optimization*, vol. 52, no. 2, pp. 325–342, 2020.

[16] P. Zou, Z. Zhou, Y.-Y. Wan, G.-L. Chen, and J. Gu, "New meta-heuristic for combinatorial optimization problems: intersection based scaling," *Journal of Computer Ence and Technology*, vol. 19, no. 6, pp. 740–751, 2004.

[17] F. Glover, M. Laguna, and R. Mart, "Tabu search," *General Information*, vol. 106, no. 2, pp. 221–225, 1997.

[18] C. R. Hwang, "Simulated annealing: theory and applications," *Acta Applicandae Mathematica*, vol. 12, no. 1, pp. 108–111, 1988.

[19] J. H. Holland and H. John, “Genetic algorithms and the optimal allocation of trials,” *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.

[20] M. Dorigo and T. Sttzle, *Ant Colony Optimization Theory*, 2004.

[21] S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer,” *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.

[22] S. Mirjalili, “Moth-flame optimization algorithm: a novel nature-inspired heuristic paradigm,” *Knowledge-Based Systems*, vol. 89, pp. 228–249, 2015.

[23] X. S. Yang and X. She, “Firefly algorithm, stochastic test functions and design optimisation,” *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[24] R. V. Rao, V. J. Savsani, and D. P. Vakharia, “Teaching-learning-based optimization: a novel method for constrained mechanical design optimization problems,” *Computer-Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.

[25] Z. Yang, K. Li, Q. Niu, Y. Xue, and A. Foley, “A self-learning TLBO based dynamic economic/environmental dispatch considering multiple plug-in electric vehicle loads,” *Journal of Modern Power Systems and Clean Energy*, vol. 2, no. 4, pp. 298–307, 2014.

[26] Y. Wang, Z. Yang, M. Mourshed, Y. Guo, Q. Niu, and X. Zhu, “Demand side management of plug-in electric vehicles and coordinated unit commitment: a novel parallel competitive swarm optimization method,” *Energy Conversion and Management*, vol. 196, pp. 935–949, 2019.

[27] Z. Yang, M. Mourshed, K. Liu, X. Xu, and S. Feng, “A novel competitive swarm optimized rbf neural network model for short-term solar power generation forecasting,” *Neurocomputing*, vol. 397, pp. 415–421, 2020.

[28] 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.

[29] G. Hou, L. Gong, Z. Yang, and J. Zhang, “Multi-objective economic model predictive control for gas turbine system based on quantum simultaneous whale optimization algorithm,” *Energy Conversion and Management*, vol. 207, Article ID 112498, 2020.

[30] Z. Yang, K. Liu, J. Fan, Y. Guo, Q. Niu, and J. Zhang, “A novel binary/real-valued pigeon-inspired optimization for economic/environment unit commitment with renewables and plug-in vehicles,” *Science China Information Sciences*, vol. 62, no. 7, p. 70213, 2019.

[31] S. Saremi, S. Mirjalili, and A. Lewis, “Grasshopper optimisation algorithm: theory and application,” *Advances in Engineering Software*, vol. 105, pp. 30–47, 2017.

[32] S. Z. Mirjalili, S. Mirjalili, S. Saremi, H. Faris, and I. Aljarah, “Grasshopper optimization algorithm for multi-objective optimization problems,” *Applied Intelligence*, vol. 48, pp. 805–820, 2018.

[33] P. Tumuluru and B. Ravi, “Goa-based DBN: grasshopper optimization algorithm-based deep belief neural networks for cancer classification,” *International Journal of Applied Engineering Research*, vol. 12, no. 24, pp. 14218–14231, 2017.

[34] 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, pp. 478–495, 2018.

[35] J. Wu, H. Wang, N. Li et al., “Distributed trajectory optimization for multiple solar-powered uavs target tracking in urban environment by adaptive grasshopper optimisation algorithm,” *Aerospace Science & Technology*, vol. 70, pp. 497–510, 2017.

[36] 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.

[37] Y. Xu, Z. Yang, X. Li, H. Kang, and X. Yang, “Dynamic opposite learning enhanced teaching-learning-based optimization,” *Knowledge-Based Systems*, vol. 188, Article ID 104966, 2020.

[38] B. J. Park, H. R. Choi, and H. S. Kim, “A hybrid genetic algorithm for the job shop scheduling problem,” *Computers & Industrial Engineering*, vol. 167, no. 4, pp. 77–95, 2003.

[39] F. Pezzella, G. Morganti, and G. Ciaschetti, “A genetic algorithm for the flexible job-shop scheduling problem,” *Computers & Operations Research*, vol. 35, no. 10, pp. 3202–3212, 2011.

[40] W. Xia and Z. Wu, “An effective hybrid optimization approach for multi-objective flexible job-shop scheduling problems,” *Computers & Industrial Engineering*, vol. 48, no. 2, pp. 409–425, 2005.

[41] V. Lal, C. An, and D. Durai, “A survey on various optimization techniques with respect to flexible job shop scheduling,” *International Journal of Scientific and Research Publications*, vol. 4, no. 2, 2014.

[42] M. Mohamed El-Amine, B. Bouziane, and A. Nassima, “Hybrid genetic algorithm for flexible job-shop scheduling problem,” *Computers & Structures*, vol. 10, no. 3, pp. 267–274, 2014.

[43] C. Gutiérrez and I. García-Magariño, “Modular design of a hybrid genetic algorithm for a flexible jobcshop scheduling problem,” *Knowledge-Based Systems*, vol. 24, no. 1, pp. 102–112, 2011.

[44] D. Lei, “A genetic algorithm for flexible job shop scheduling with fuzzy processing time,” *International Journal of Production Research*, vol. 48, no. 10–12, pp. 2995–3013, 2010.

[45] J.-q. Li, Q.-k. Pan, and Y.-C. Liang, “An effective hybrid Tabu search algorithm for multi-objective flexible job-shop scheduling problems,” *Computers & Industrial Engineering*, vol. 59, no. 4, pp. 647–662, 2010.

[46] X. Li and L. Gao, “An effective hybrid genetic algorithm and Tabu search for flexible job shop scheduling problem,” *International Journal of Production Economics*, vol. 174, pp. 93–110, 2016.

[47] L. Wang, G. Zhou, Y. Xu, and M. Liu, “A hybrid artificial bee colony algorithm for the fuzzy flexible job-shop scheduling problem,” *International Journal of Production Research*, vol. 51, no. 11-12, pp. 3593–3608, 2013.

[48] L. Wang, G. Zhou, Y. Xu, S. Wang, and M. Liu, “An effective artificial bee colony algorithm for the flexible job-shop scheduling problem,” *International Journal of Advanced Manufacturing Technology*, vol. 60, no. 1–4, pp. 303–315, 2012.

[49] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, “Enhancing particle swarm optimization using generalized opposition-based learning,” *Information Sciences*, vol. 181, no. 20, pp. 4699–4714, 2011.

[50] C. Zhang, Z. Ni, Z. Wu, and L. Gu, “A novel swarm model with quasi-oppositional particle,” in *Proceedings of the 2009 International Forum on Information Technology & Applications*, Chengdu, China, May 2009.

[51] A. Saha, P. Das, and A. K. Chakraborty, “Quasi-reflection based symbiotic organisms search algorithm for solving static

optimal power flow problem," *Scientia Iranica*, vol. 26, no. 3, 2018.

[52] C. Yan, S. Li, L. Yang, and L. He, "Investigation of fef process liquid phase migration using orthogonal design of experiments," *Rapid Prototyping Journal*, vol. 25, no. 2, 2018.

[53] M. Akhshabi, M. Akhshabi, and J. Khalatbari, "A particle swarm optimization algorithm for solving flexible job-shop scheduling problem," *Journal of Basic and Applied Scientific Research*, vol. 1, no. 12, pp. 3240–3244, 2011.

[54] R. H. Caldeira and A. Gnanavelbabu, "Solving the flexible job shop scheduling problem using an improved jaya algorithm," *Computers & Industrial Engineering*, vol. 137, Article ID 106064, 2019.

[55] T. Jiang and C. Zhang, "Application of grey wolf optimization for solving combinatorial problems: job shop and flexible job shop scheduling cases," *IEEE Access*, vol. 6, pp. 26231–26240, 2018.

[56] Y. Yuan and H. Xu, "Flexible job shop scheduling using hybrid differential evolution algorithms," *Computers & Industrial Engineering*, vol. 65, no. 2, pp. 246–260, 2013.

[57] P. Fattahi, M. Saidi Mehrabad, and F. Jolai, "Mathematical modeling and heuristic approaches to flexible job shop scheduling problems," *Journal of Intelligent Manufacturing*, vol. 18, no. 3, pp. 331–342, 2007.

[58] I. Kacem, S. Hammadi, and P. Borne, "Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic," *Mathematics and Computers in Simulation*, vol. 60, no. 3–5, pp. 245–276, 2002.