ISSN 2616-5775 Vol. 8, Issue 9: 9-16, DOI: 10.25236/AJCIS.2025.080902

Application of Improved Genetic Algorithm Based on Gene Duplication Rate in Flexible Job Shop Scheduling Problem

Bin Li

School of Intelligent Transportation Modern Industry, Anhui Sanlian University, Hefei, 230601, China

Abstract: *Genetic algorithms often tend to exhibit premature convergence in solving the Flexible Job-shop Scheduling Problem (FJSP). To overcome this, we propose an improved genetic algorithm guided by gene repetition rates and formulate a mathematical model for dynamic adjustment of genetic operation parameters. The algorithm employs a two-layer chromosome encoding for operation sequences and machine assignments, integrating a gene repetition feedback mechanism to adaptively regulate crossover and mutation probabilities, thereby enhancing population diversity and global search capability. Experiments on benchmark datasets show that the proposed method achieves superior optimization accuracy and faster convergence compared to traditional genetic algorithms, providing an effective approach for solving FJSP.*

Keywords: *Flexible Job-Shop Scheduling Problem, Genetic Algorithm, Gene Repetition Rate, Adaptive Optimization, Chromosome Diversity*

1. Introduction

With the rise of intelligent manufacturing, the Flexible Job-Shop Scheduling Problem (FJSP) has become key to optimizing resource allocation in production systems[1], directly affecting efficiency. The advent of Industry 4.0, personalized customization, and lean production has broadened FJSP's solution space. Traditional rule-based scheduling struggles to achieve global optima under dynamic tasks and constraints. FJSP's flexibility in machine selection overcomes fixed-path limitations, coordinating heterogeneous devices, multi-step processes, and diverse tasks, making it a focal point in both research and industrial applications [2].

Genetic algorithm-based optimization has long been a primary focus in FJSP research. Researchers have proposed various strategies to enhance performance. Li[3] introduced elitism retention and dynamic crossover operators to improve genetic algorithm (GA). Long et al.[4] applied reinforcement learning to boost global search ability, while Jia et al.[5] combined adaptive parameters and chaotic local search for enhanced accuracy and speed. Other optimization approaches, such as knowledge-based ant colony optimization and hybrid particle swarm–tabu search, have also made significant advancements.

Despite these developments, a gap exists in the quantitative analysis of population gene distribution. A systematic adaptive mechanism based on gene diversity has yet to be established, presenting an opportunity for further research.

This study aims to optimize FJSP using GA, focusing on minimizing makespan. We introduce a novel metric, the "gene repetition rate," to quantify gene distribution in the population and develop a dynamic adaptive adjustment model. Based on this, an adaptive GA guided by gene repetition rate (GAGR) is introduced. This model dynamically optimizes genetic operation parameters, including crossover and mutation probabilities, according to the degree of gene repetition, enabling intelligent adjustment of the search strategy during iterations. By addressing the limitations of fixed genetic parameters, this approach offers an effective method for exploring FJSP's complex solution space, improving production efficiency, and advancing intelligent manufacturing.

2. Problem Description

The FJSP is a complex production optimization problem involving multiple jobs and machine types. The goal is to determine the optimal operation sequence and assign each operation to the most suitable

ISSN 2616-5775 Vol. 8, Issue 9: 9-16, DOI: 10.25236/AJCIS.2025.080902

machine.

Formally, there are n mutually independent jobs $\{J_1, J_2, \dots, J_n\}$ to be processed on m different machines $\{M_1, M_2, \dots, M_m\}$. Each job J_i consists of a sequence of operations u（$u = 1,2,\dots,3$）, each with a strictly positive and predetermined processing time.For every operation, there is at least one machine capable of performing it.

The model is subject to strict constraints: each machine handles only one operation at a time, operations are non-preemptive, and transfer times between machines are ignored. The primary objective is to minimize the makespan of all jobs to achieve systematic and quantifiable scheduling optimization. The constraints and objective of the model are shown in Formula 1.

$$f = \min\left(\sum_{i=1}^{n}\sum_{j=1}^{u_i} s_{i,j,k} + t_{i,j,k}\right) \tag{1}$$

Where, $S_{i,j,k}$ represents the start time of operation O_{ij} when processed on machine k. $t_{i,j,k}$ represents the processing time required for operation O_{ij} on machine k.

3. Algorithm Design

3.1 Chromosome Representation

Solving FJSP requires coordinated optimization of the operation sequence (OS) and machine selection (MS)[6]. A two-layer chromosome encoding is effective for this. In the OS layer, each number represents a specific operation of a job, and the chromosome length equals the total number of operations, ensuring a one-to-one mapping. For example, the sequence '3-1-2-2-1' shown in Figure 1, '3' represents the first operation of the third job O_{31}, '1' the first operation of the first job O_{11}, and '2' the first operation of the second job O_{21}. The next '2' and '1' correspond to O_{22} and O_{12}, respectively. Thus, the sequence '3-1-2-2-1' can be decoded as the operation order 'O_{31}-O_{11}-O_{21}-O_{22}-O_{12}'. Each number uniquely maps to an operation, and the chromosome length equals the total number of operations, providing an intuitive numerical representation for optimization.

The chromosome construction logic of the MS layer is similar to that of the OS layer, with its length equal to the total number of operations. For example, in the MS coding shown in Figure 1, the gene corresponding to operation O_{22} is '3', indicating that this operation is assigned to the third machine capable of processing it. It should be noted that the total number of actually available machines is determined by the equipment configuration in specific problem scenarios. Based on this rule, each value in the MS segment is endowed with a clear semantic meaning, exclusively referring to the machine number that executes the corresponding operation, thereby establishing an accurate mapping between operations and processing equipment.

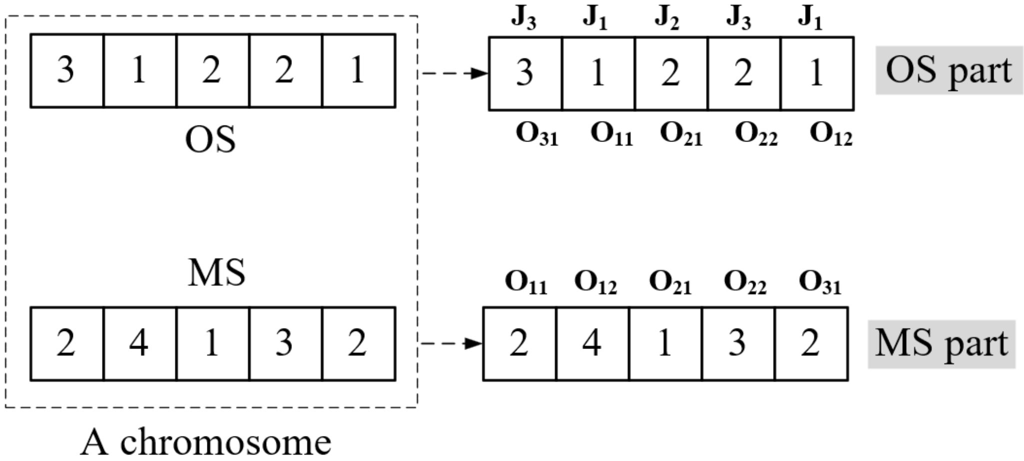

Figure 1: Example of a double-layered chromosome.

ISSN 2616-5775 Vol. 8, Issue 9: 9-16, DOI: 10.25236/AJCIS.2025.080902

3.2 Population Initialization and Evolution Operator

3.2.1 MS initialization

Traditional GA for FJSP typically initialize the population by randomly generating chromosomes, which can result in uneven gene distribution and excessive repetition, thereby limiting the algorithm's global search capability. A widely adopted strategy for MS initialization is the global-local selection method proposed by Zhang[7], which is also employed in this study. Beyond this approach, we propose a new strategy based on the gene repetition rate metric 'hierarchical encoding → repetition rate evaluation → dynamic adjustment' to dynamically regulate population initialization. This method improves the initial population's coverage of the solution space, providing a high-quality starting point for subsequent iterations. The specific process is as follows:

(1) Based on the determined operation layer coding, available machines are randomly assigned to each operation. It is ensured that the assigned machines support the processing requirements of the corresponding operations, generating legal machine layer coding to form the MS segment of the initial population.

(2) MS layer repetition rate evaluation and grouping: Calculate the gene repetition rate of the machine layer, and count the proportion of chromosomes with identical machine coding combinations. Group by operation layer coding, compare the repetition of machine layer coding within each group, and mark groups with high repetition rates.

(3) For groups with high repetition rates, retain 1-2 chromosomes with the optimal machine layer coding (evaluated by initial fitness). For the remaining chromosomes, perform "machine reallocation" by randomly selecting unused machines to replace the original coding; or generate new machine layer coding chromosomes and add them to the population until the machine layer repetition rate drops below the threshold.

Table 1: Operational procedures based on gene duplication rate indicators.

Phase	Operation process	Details
1.Initialization	Randomly assign compatible machines to each operation	Ensure that machine assignments meet production requirement constraints and generate valid MS encoding.
2.Evaluate repetition rate	Calculate the proportion of chromosomes that share the same MS encoding (gene repetition rate = number of duplicate chromosomes / population size). Group by the OS and flag groups whose repetition rate exceeds the threshold.	Chromosomes sharing the same OS encoding are grouped into the same set.
3.Adaptive adjustment	Within each group, preserve the 1–2 chromosomes with the highest fitness. For the remaining chromosomes whose repetition rate is below the threshold, apply: ① machine reassignment; ② generation of additional new MS encodings.	Fitness evaluation criteria: makespan and degree of machine load balance.

3.2.2 OS initialization

Due to the lack of a comprehensive OS initialization scheme, this study adopts the Remaining Operations Prioritized Initialization (ROPI) method. Two arrays are maintained: one tracks the selection count of each job, and the other records the remaining operations. Iteratively, the job with the most remaining operations is selected (randomly if tied), recorded, and its remaining operations decremented. This process repeats until all operations are assigned, producing a complete OS sequence for the chromosome.

3.2.3 Evolutionary Operator

The core of a GA lies in simulating natural selection, with the ternary tournament selection strategy serving as a key mechanism. In this strategy, three individuals are randomly chosen from the population for fitness comparison, and the superior individuals are selected for inclusion in the gene pool. This enables directional retention of high-quality genetic information. Over multiple generations of reproduction and iteration, the accumulation of superior genes provides an efficient genetic foundation for subsequent evolutionary processes, guiding the population toward higher adaptability.

In view of the characteristics of OS and MS modules in the double-layer chromosome structure, a differentiated crossover operation strategy is adopted. In the OS module processing, the improved precedence order crossover (IPOX)[8] is adopted.

In the MS layer, each gene represents a machine assigned to a specific operation. A two-point crossover is applied, exchanging gene segments between two positions while ensuring machine compatibility and operation timing constraints. This expands the search space while maintaining feasibility. To prevent premature convergence, a two-point mutation is applied in the OS layer, swapping two genes to enhance exploration of the solution space.

3.3 Evolutionary algorithm guided by gene repetition rate

To prevent premature convergence of the GA, this study introduces the gene repetition rate as an important indicator of population diversity. The flowchart of evolution guided by gene duplication rate is shown in Figure 2. The gene repetition rate reflects the degree of gene repetition within the population, and when the repetition rate is too high, it indicates insufficient diversity, potentially causing the algorithm to get trapped in a local optimum. The gene repetition rate R is calculated using the following formula:

$$R=\frac{\sum_{i=1}^{N}\sum_{j=i+1}^{N}\left(\delta(OS(C_i),OS(C_j))+\delta(MS(C_i),MS(C_j))\right)}{N(N-1)/2} \tag{2}$$

Where:

$\delta(\mathrm{os}(c_i),\mathrm{os}(c_j))=1$ if the operation sequence part (os) of individual c_i and c_j are the same, otherwise $\delta(\mathrm{os}(c_i),\mathrm{os}(c_j))=0$;

$\delta(\mathrm{ms}(c_i),\mathrm{ms}(c_j))=1$ if the machine selection part (ms) of individual c_i and c_j are the same, otherwise $\delta(\mathrm{ms}(c_i),\mathrm{ms}(c_j))=0$;

·n is the number of individuals in the population;

l is the length of each chromosome, representing the total number of operations in the tasks.

Equations (3) and (4) present the specific calculation processes for the crossover and mutation probabilities of the algorithm.

$$P_c=P_{c0}\times\exp\left(1-\frac{R-R_{\min}}{R_{\max}-R_{\min}}\right) \tag{3}$$

$$P_m=P_{m0}\times\left(1+\frac{R-R_{\min}}{R_{\max}-R_{\min}}\right) \tag{4}$$

Where, $P_{c0}=0.8, P_{m0}=0.1$ represent the initial replication rate and initial crossover rate. $R_{\min}, R_{\max}$ indicating the historical minimum/maximum gene duplication rate (R) in the current evolutionary stage. This mechanism ensures that when the R exceeds the threshold, the algorithm triggers enhanced mutation operations to jump out of local optima. When this situation persists for 10 generations, the algorithm activates the "gene recombination" mechanism:

(1) Retain the top 10% of elite chromosomes.

(2) Replace 30% of the population with new chromosomes generated by the Gene Diversity

ISSN 2616-5775 Vol. 8, Issue 9: 9-16, DOI: 10.25236/AJCIS.2025.080902

Injection (GDI) operator, which introduces random gene segments from historically optimal solutions.

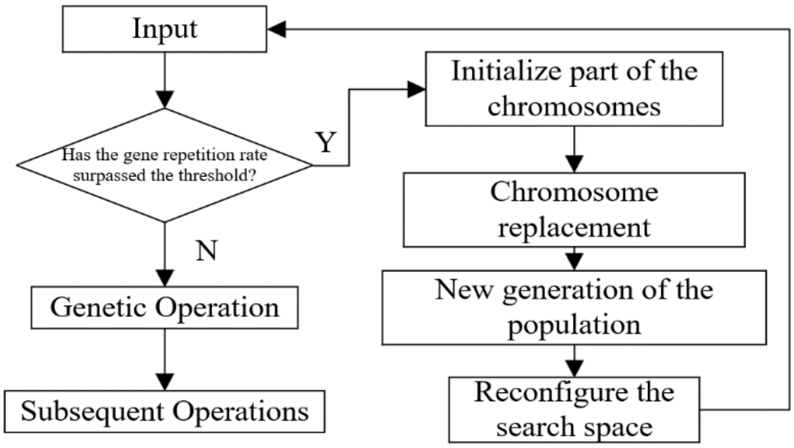

Figure 2: Flowchart of evolution guided by gene repetition rate.

3.4 The framework of Improved GA

Figure 3 presents the framework of the algorithm derived from the preceding discussion.

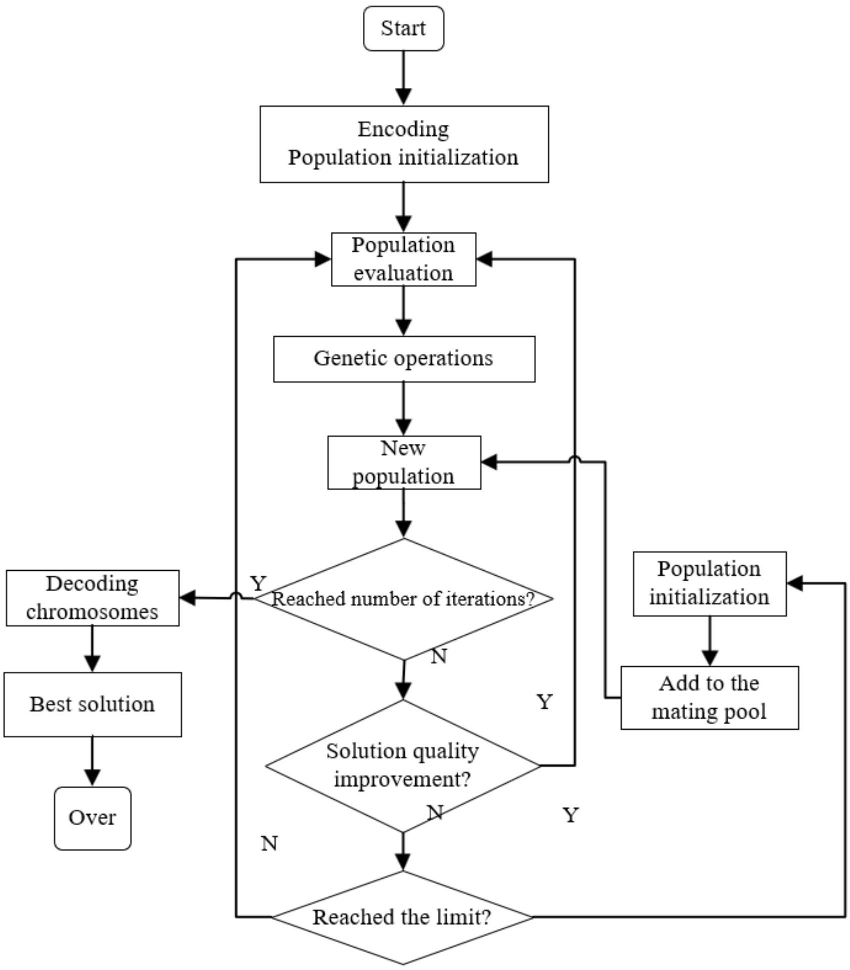

Figure 3: Framework diagram of the algorithm.

4. Simulation Experiments and Performance Analysis

The algorithm presented in this study is implemented using Matlab 2020a, and the experimental

ISSN 2616-5775 Vol. 8, Issue 9: 9-16, DOI: 10.25236/AJCIS.2025.080902

environment is configured with an Intel Core i5 processor, 12GB of RAM, and a Windows 10 operating system. To validate the effectiveness of the algorithm, two widely recognized FJSP benchmark datasets are selected. One set is the relatively small-scale Kacem datasets[9], while the other is the more complex Brandimarte datasets[10]. For each test instance, 20 experimental runs are conducted. The population size of the algorithm is set to $5\times m\times n$ (where m,n represent the instance size), and the maximum number of iterations is set to $10\times m\times n$.

To evaluate the overall performance of GAGR, this section compares it with three other algorithms: SAGA[11], SLGA[12], and edPSO[13]. Table 1 presents the comparison results of these algorithms on the Kacem dataset, where "-" indicates that the result was not reported in the literature. Due to the lack of the original Kacem05, GAGR was able to achieve the lower bound (LB) results for the remaining four instances. This indicates that GAGR demonstrates better performance when handling smaller-scale problems.

Table 2: Best makespan of each algorithm on the Kacem datasets.

Instances	LB	SAGA	edPSO	SLGA	GAGR
Kcaem01	11	11	11	11	11
Kcaem02	14	14	17	14	14
Kcaem03	11	11	-	11	11
Kcaem04	7	8	8	-	7
Kcaem05	11	-	-	-	-

Table 2 presents the experimental results of several algorithms on 10 medium- to large-scale BR datasets. As can be seen from the data in this table, each algorithm can generally achieve the theoretical optimal value on the slightly simpler MK03 and MK08 instances. Although it does not achieve the best results on the smaller MK instances, the proposed GAGR algorithm demonstrates significant superiority on the more complex MK09 and 10 instances. These experimental results demonstrate that GAGR exhibits greater adaptability when handling large-scale problems.

Table 3: Best makespan of each algorithm on the BR datasets.

Instances	LB	SAGR	edPSO	SLGA	GAGR
MK01	36	40	41	40	41
MK02	24	29	26	27	27
MK03	204	204	207	204	204
MK04	48	60	65	60	60
MK05	168	176	171	172	172
MK06	33	67	61	69	69
MK07	133	144	173	144	144
MK08	523	523	523	523	523
MK09	299	312	307	320	312
MK10	165	209	312	254	204

The Actual Relative Percentage Deviation (ARPD) is a metric used to further assess the performance of an algorithm by comparing its solution with the optimal solution or known lower bound.

Table 3 provides a detailed presentation of the actual number of solved instances (ASI) by each algorithm along with the ARPD values for each algorithm. The specific calculation method involves first determining the difference between the algorithm's solution and the lower bound, then dividing by the optimal solution or lower bound, and finally multiplying the resulting ratio by 100 to express the deviation of the algorithm's solution from the theoretical value. The formula is as follows:

$$ARPD = \frac{\sum_{i=1}^{ASI} \frac{(Best - LB) \times 100}{LB}}{ASI} \tag{5}$$

Where, 'Best' denotes the optimal result achieved over 20 runs for each instance by the respective algorithms. For the same instance, the ARPD value of GAGR is denoted as 'GAGR' s ARPD'. The 'Improvement' column specifically refers to the performance improvement in ARPD achieved by the GAGR algorithm compared to other algorithms tested on the two datasets. The higher the value, the more significant the improvement.

The results of each algorithm are presented in Table 4. From the data, it is evident that GAGR achieved varying degrees of improvement in ARPD. Although GAGR did not find the optimal solution for smaller-scale instances, it reduced the ARPD values in the more complex MK09 and MK10 instances. These results confirm that GAGR outperforms other methods in optimizing the makespan of FJSP.

Table 4: Comparison of ARPD values for each algorithm.

Algorithms	ASI	ARPD (%)	GAGR's ARPD (%)	Improvement (%)
SAGA	14	15.6	14.2	1.4
edPSO	13	17.7	15.3	2.4
SLGA	13	17.6	15.3	2.3

5. Conclusions

To solve the FJSP, we proposed an adaptive GA based on gene repetition rate (GAGR). During implementation, the algorithm detects the repetition rate of genes in different chromosomes and uses it to guide the evolutionary process. Furthermore, a two-layer gene chain chromosome representation and adaptive evolutionary operators are used for genetic manipulation to increase chromosome diversity and enrich the population's gene pool.

Finally, GAGR was tested on selected benchmarks. Experimental validation showed that GAGR demonstrated good performance in solving the FJSP. This research not only provides a new and effective approach for solving the FJSP but also offers valuable insights for practical optimization problems in production scheduling. However, this algorithm may occasionally converge to local optima on small-scale problems, which will be our future research direction.

Acknowledgements

Funded Project: Anhui Sanlian University 2024 School-level Research Platform General Project:

Research on Solving Flexible Job Shop Scheduling Problems Based on Evolutionary Algorithms

(Project No: KJYB2024011)

References

[1] Tu J. Research on mechanical design, manufacturing and automation technology in the era of intelligent manufacturing[J]. New Technology in Engineering Construction, 2022, 1(2): 153-155.
[2] Wei W, Tan J, Feng Y, et al. Research on multi-objective optimization method for flexible job shop scheduling problem [J]. Computer Integrated Manufacturing Systems, 2009, 15(8): 1592-1598.
[3] Li X, Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling

ISSN 2616-5775 Vol. 8, Issue 9: 9-16, DOI: 10.25236/AJCIS.2025.080902

problem[J]. International Journal of Production Economics, 2016, 174: 93-110.

[4] Long X, Zhang J, Qi X, et al. A self-learning artificial bee colony algorithm based on reinforcement learning for a flexible job-shop scheduling problem[J]. Concurrency and Computation: Practice and Experience, 2022, 34(4): e6658.

[5] Jia Z, Chen H, Tang J. An improved particle swarm optimization for multi-objective flexible job-shop scheduling problem[C], 2007 IEEE international conference on grey systems and intelligent services. IEEE, 2007: 1587-1592.

[6] Li B, Xia X. A Self-Adjusting Search Domain Method-Based Genetic Algorithm for Solving Flexible Job Shop Scheduling Problem. Computational Intelligence and Neuroscience. 2022 Oct 10;2022: 4212556. doi: 10.1155/2022/4212556. PMID: 36262613; PMCID: PMC9576347.

[7] Zhang G, Shao X, Li P, et al. An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem[J]. Computers & Industrial Engineering, 2009, 56(4): 1309-1318.

[8] Zhang C, Rao Y, Liu X, et al. Genetic algorithm based on POX crossover for solving job-shop scheduling problem[J]. China Mechanical Engineering, 2004, 15(23): 2149-2153.

[9] Kacem I, Hammadi S, Borne P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems[J]. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 2002, 32(1): 1-13.

[10] Brandimarte P. Routing and scheduling in a flexible job shop by tabu search[J]. Annals of Operations research, 1993, 41(3): 157-183.

[11] Li B. Improved genetic algorithm for solving flexible job shop scheduling problem[J]. Computer Knowledge and Technology, 2024, 20(27):79-82.

[12] Chen R, Yang B, Li S, et al. A self-learning genetic algorithm based on reinforcement learning for flexible job-shop scheduling problem[J]. Computers & Industrial Engineering, 2020, 149: 106778.

[13] Jiang T, Zhang C. Application of grey wolf optimization for solving combinatorial problems: Job shop and flexible job shop scheduling cases[J]. IEEE Access, 2018, 6: 26231-26240.