A Hybrid Gray Wolf Weed Algorithm for Flexible Job-shop Scheduling Problem

Yi Wang, Yu-chuan Song, Yu-ji Zou, Qi Lei, and Xin-kun Wang

College of Mechanical Engineering Chongqing University, Chongqing University, 400044 Chongqing, China

*Email: syc@cqu.edu.cn

Abstract: For the flexible job shop scheduling problem with the goal of makespan, a mathematical model is established and a hybrid algorithm (GIWO) based on gray wolf and invasive weeds is proposed. In this paper, the intrusive weed optimization algorithm is embedded on the basis of the gray wolf algorithm to improve the algorithm's global search and local search ability. And an effective initial population generation mechanism is added to improve the quality of the initial solution. A variable neighborhood search structure that adaptively changes according to the search is proposed. Finally, other improved gray wolf algorithms are compared with the algorithm in this paper through different benchmark instances. Experimental results show that the proposed algorithm performs better than the other algorithms in effectiveness and superiority. The algorithm can effectively solve the flexible job shop scheduling problem.

1. Introduction

The job shop scheduling problem (Job shop scheduling problem, JSP) has proved it’s an NP-hard problem by GAREY [1]. Flexible job shop scheduling problem (FJSP), as an extension of the classic job shop scheduling problem, is more suitable for new modern manufacturing models. FJSP was first proposed by BRUCKER [2] and had proved to be a more complex NP-hard problem. Compared with the job shop scheduling problem (JSP), in the FJSP problem each operation can be processed in a series of candidate machines, which increases the flexibility of scheduling in the processing process and is more in line with the actual factory production and processing. The research of FJSP problem has become very important in the field of workshop scheduling. At present, the meta-heuristic algorithm has become the main methods to solve the problem of workshop scheduling. The meta-heuristic algorithm provides a detailed solution method for the FJSP problem, and has attracted widespread attention from scholars at home and abroad, but no single algorithm can be comprehensive effectively solve FJSP problems.

Many the meta-heuristic algorithms are developed to solve the maximum completion time of the FJSP problem. Ghiath proposed an improved greedy algorithm to solve the makespan of the workshop in the flexible job shop problem, and constructed a scheduling optimization model of the maximum completion time [3]; Guiliang proposed a new decomposition-based the multi-objective solution algorithm of FJSP, in order to optimize the three goals of the makespan, total weighted early/tardiness penalty and total energy consumption in FJSP [4]; JUN-QIANG LI proposes an improved tabu search algorithm to optimize the four goals of early/delay penalty, maximum workload, completion period and total process workload in FJSP, and builds a multi-objective shop scheduling optimization model.
[5]. Currently, most of the problems of FJSP are solved by improved classic algorithm, that integrating a certain classic operator into another algorithm, or the parallel about two classic algorithms, etc. Less attention and research on the newer meta-heuristic algorithms and their extendible structure, so the excellent characteristics of the new meta-heuristics, such as fast convergence speed, strong local search ability, strong robustness are ignored. Therefore, this paper uses new meta-heuristic algorithm to establish a serial structure model, and uses the inter-algorithm dynamic and the excellent characteristic of the new algorithm to solve the FJSP problem.

Grey wolf optimization algorithm is a new meta-heuristic algorithm based on the principle of wolf predation process [6-8]. Although the algorithm has been studied and explored late, it has been applied to various optimization problems such as UAV navigation and rectangular part layout due to its excellent global search capabilities. Invasive weed optimization algorithm is a group meta-heuristic optimization algorithm that simulates the reproduction process of weeds in rice fields [9-11]. It is similar to the gray wolf algorithm. Due to its excellent local search ability, it has been applied to various fields in class optimization problem. However, the two algorithms are currently less used in the field of production scheduling, and there are only a few literatures that apply the gray wolf algorithm or the invading weed algorithm to the flexible job shop scheduling problem.

This paper focuses on the huge solution space of FJSP problem and the shortcoming of existing algorithms in optimization. A series of designs and improvements are made on the excellent characteristics of the gray wolf algorithm and the invading weed optimization algorithm and propose the hybrid gray wolf weed optimization algorithm (GIWO). This paper adopts two-stage random key coding combined with heuristic rules to improve the quality of the initial population, and uses variable neighborhood search to strengthen the local search ability of the overall algorithm. In this paper, the weed algorithm is embedded in the algorithm to form a serial structure of the algorithm, and make use of the dynamic change of fitness during the population iteration process to achieve better secondary exploration of excellent individuals. Experiments with benchmarks are tested, and the results verify its effectiveness and superiority.

2. Formulation of the Problem
The flexible job shop scheduling problem is a discrete job planning and coordination problem. It is described as: \( J = (J_1, J_2, \cdots, J_n) \) is a set of \( n \) independent jobs to be processed by a set of machines \( \{M_1, M_2, \cdots, M_n\} \). During processing the operation follows the constraints: each machine can only process one process at the same time; each process can only be processed on one machine at the same time; only consider the order constraint relationship between the same job’s process.

This paper aims to optimize the maximum completion time, as follows:

\[
T_{\text{max}} = \min \left( \max_{i=1}^{n} (T_i) \right)
\]  

(1)

constraints:

\[
TE_{ij} = TS_{ij} \times x_{ijh} + P_{ijh}
\]  

(2)

\[
TE_{ij} \leq TS_{ij(h+1)}
\]  

(3)

\[
\sum_{h=1}^{m_i} x_{ijh} = 1
\]  

(4)

\[
TE_{kl} \times x_{klh} \leq TS_{kl} \times x_{klh}, (TS_{ij} \geq TS_{kl})
\]  

(5)

\[
TE_{ij} \times x_{ijh} \leq TS_{ij} \times x_{ijh}, (TS_{kl} \geq TS_{ij})
\]  

(6)

where, the total jobs is \( n \); \( T_i \) is the completion time of job \( J_i \); \( TS_{ij} \) is the start time of operation \( OP_{ij} \); \( TE_{ij} \) is the completion time of operation \( OP_{ij} \); \( P_{ijh} \) is the processing time of operation \( OP_{ij} \) on machine \( h \); \( x_{ijh} \) If the operation \( OP_{ij} \) processed by machine \( h \), \( x_{ijh} \) will be 1, else will be 0; Eq.(2) ensure the processing time sustained, Eq.(3) ensure the processing sequence; Eq.(5)-(6) guarantee that two operations can’t be processed on the same machine at the same time.
3. The Proposed GIWO

3.1. Overview of GWO and IWO
Gray wolf optimization (GWO) algorithm is a meta-heuristic optimization algorithm based on swarm intelligence, which is mainly proposed by imitating the predation behavior of wolves. In the algorithm evolution iterative process, according to individual fitness, the population individuals need to be divided into four levels, arranged in descending order according to the population individual fitness value, and the top three individuals in the sequence are defined as $\alpha, \beta, \delta$. The remaining individuals in the population are $\omega$. In each iteration of the algorithm, it first finds out the three best-fits in the population and then guides the changes of the individual positions in the offspring wolves. Through continuous iterations, food is finally found and captured, which can be given as follows Eq.(7)-(15). Invasive weed optimization (IWO) algorithm is a group meta-heuristic optimization algorithm that simulates the propagation of weeds in rice fields.

$$D_\alpha = |C_1X_\alpha(t) - X(t)|$$
$$D_\beta = |C_1X_\beta(t) - X(t)|$$
$$D_\delta = |C_1X_\delta(t) - X(t)|$$
$$X_1 = X_\alpha(t) - A_\alpha D_\alpha$$
$$X_2 = X_\beta(t) - A_\beta D_\beta$$
$$X_3 = X_\delta(t) - A_\delta D_\delta$$
$$X(t+1) = \left(\frac{X_1 + X_2 + X_3}{3}\right)$$
$$A = 2ar_1 - a$$
$$C = 2r_2$$

3.2. The Framework of the GIWO Algorithm
The gray wolf invasion weed optimization hybrid algorithm mainly uses the good characteristics of the two algorithms to perform collaborative processing. The basic flow chart of proposed GIWO is shown in Figure 1. The detail of steps as follows:
3.3. Encoding and Decoding

The standard GWO or IWO algorithm is mainly used to solve continuous variable optimization problems. The FJSP problem can be subdivided into two problems: machine allocation and process sequencing, which are discrete optimization problems. In order to achieve the correspondence of continuous spatial variables to discrete variables. The individuals of the population are divided into two-segment coding and the length is \(2l\), corresponding to the machine allocation plan and the process sorting plan, as shown in Figures 2 and 3. This article uses ROV rules based on random key coding [12,13].

1) Machine assignment

The machine part is mainly located in the first half of the individual \(X = \{x(1), x(2), \ldots, x(l)\}\), and its sorting adopts GLR machine selection methods, namely: global selection (GS), local selection (LS) and random selection (RS), to achieve the balance of machine workload and high utilization rate, and improve the purpose of the initial solution quality. Its use ratio is 0.6, 0.3, 0.1.

2) Operation sequence

The process sequence part is mainly located in the latter half of the individual \(X = \{x(l + 1), x(l + 2), \ldots, x(2l)\}\). The process part will be arranged according to the ROV rule. First, a position value will be randomly generated for each process position, the value range is \([-\varepsilon, \varepsilon]\). Then, according to the random value, the process arrangement is arranged of the ROV ascending order rule.

**Figure 1.** Hybrid algorithm flow chart.
In the process arrangement part of decoding, arrange the position elements according to the ascending ROV rule, and convert the position elements into the process sequence numbers they represent, as shown in Figure 4. At the same time, in order to realize that the process can be processed at the earliest possible processing time on the corresponding selectable machine, and to reduce the maximum completion time, the decoding method of active decoding scheduling is adopted.

3.4. Initialization
According to the vector representation of individual position, the initialization process is carried out in two steps. The first step is to complete the allocation and sorting of the machine, which is mainly the GLR selection sorting method, which is selected according to the ratio of 0.6, 0.3, and 0.1; the second step is the process sorting part, which is carried out according to the random key coding method.

3.5. The Weed Optimization Algorithm
The gray wolf algorithm forms the initial population, and sorts out the top half of the gray wolf individuals according to the fitness of the population, and uses it as the initial parent population of the weed algorithm, and starts to search for the optimal individual iteratively. In this way, the invading weed optimization algorithm is used to find the optimal solution [14]. When the invasive weed optimization algorithm iteratively searches for the optimal value in the population, the encoding and decoding rules are consistent with the gray wolf algorithm. The detail steps as follows:

Step 1: Select the number of individuals in the invading weed algorithm population $P_n$ according to the population number in the gray wolf algorithm, and determine the maximum number of iterations $I_n$.

Step 2: Calculate the fitness of the population individuals, and calculate the number of seeds $N_e$ produced by one individual, as follows:
\[ N_s = \left( f_n - f_m \right) \times \left( \frac{S_m - S_n}{S_m - S_n + 1} \right) + S_n \]  

(16)

where, \( f_n \) represents the fitness of the current individual; \( f_m \) represents the minimum fitness in the current population; \( S_m, S_n \) represents the maximum and minimum number of seeds that individual of the population can produce; \( \left\lceil \right\rceil \) represents rounding.

Step 3: Calculate the variance of individual, as follows:

\[ D_n = \left( (I_n - I_n)^{(L)} \times D_n + I_n^{(L)} + D_m \right) \]  

(17)

where, \( I_n, I_n \) represents the maximum and minimum iterations; \( D_n, D_m \) represents the maximum and minimum variance number. \( L \) represents total processes.

Step 4: If the number of individuals in the current population reaches the maximum number of individuals, replace individuals with poorer parents based on fitness to maintain the population size. When the maximum number of iterations is reached, the invasive weed optimization algorithm is jumped out.

In the iterative search process of invading weeds algorithm, the mutation uses position mutation, as follows:

\[ X_{(i+1)} = X_i + \xi \]  

(18)

where \( \xi \) distributed evenly in \([-1,1]\).

3.6. Neighborhood Structure

In the entire iterative search process of the algorithm, the quality of the population individuals, especially the three wolves that guide the decision-making level, will determine the final result of the entire optimization algorithm. This paper embeds variable neighborhood search into the whole algorithm, and acts on \( \alpha, \beta, \delta \) three individuals in each iteration. For the individuals, the following three neighborhood structures are designed as follows:

NS1: In the process arrangement part, choose two position elements of different processes, and exchange these two position elements.

NS2: Choose a position element in the process arrangement part of the individual, read the value of its position, and change the value of its position through a multi-strategy method, that using different search methods for different individuals. The fitness of the top three of wolves is retained, and according to the fitness difference \( \alpha, \beta, \delta \), as Eq. (19)-(21), it is transformed into the number of changing positions, for \( \alpha \) the process change position is one. This structure can adaptively change the number of position searches in each iteration of the evolution process.

\[ \alpha : \]

\[ X_{\alpha(i+1)} = X_{\alpha(i)} + a \]  

(19)

\[ \beta : \]

\[ X_{\beta(i+1)} = X_{\beta(i)} + a \times (X_{\alpha(i)} - X_{\beta(i)}) \]  

(20)

\[ \delta : \]

\[ X_{\delta(i+1)} = X_{\delta(i)} + a \times (X_{\alpha(i)} - X_{\delta(i)}) + b \times (X_{\beta(i)} - X_{\delta(i)}) \]  

(21)

where \( a, b \) are random numbers.

NS3: In the machine arrangement part, an element at a certain position is randomly selected. If there is more than one candidate machine corresponding to this position element, a new machine is randomly selected; if there is only one candidate machine, another position element is randomly selected.

The details of steps can be described as follows:

Step 1: Take the first three wolf individual positions as the initial solution \( X \) and set the maximum number of iterations \( q_{\text{aux}} \).
Step 2: Calculate the difference between the individual fitness of the three wolves. When the individual fitness of the three wolves or the two wolves are the same, the nearest individual fitness must be selected for replacement, and a random number \( r \) will be generated, as follows:

\[
f_1 = \frac{(f_\beta - f_\alpha)}{(f_\delta - f_\alpha)} \quad (22)
\]

\[
f_2 = \frac{(f_\alpha - f_\beta)}{(f_\delta - f_\alpha)} \quad (23)
\]

According to the difference between \( \alpha, \beta \) and the difference between \( \beta, \delta \) to form the roulette rule.

Step 3: vibration

\[
\begin{align*}
N1 & : (r < f_1) \\
N2 & : (f_1 < r \leq f_2) \\
N3 & : (f_2 < r \leq f_3)
\end{align*}
\]

According to the \( r \), chooses different neighborhood search structures to generate new solutions \( X' \).

Step 4: If the fitness of \( X' \) less than the fitness of \( X \), the individual \( X \) will replace \( X \), and \( q \) increases 1.

Step 5: If \( q > q_{\text{max}} \), go to Step 6. If \( q \leq q_{\text{max}} \), go to Step 2.

Step 6: The search is end.

3.7. Across and mutation

In the iterative of GIWO, it is easy to fall into a local optimal solution and reduce the diversity of the population, when only the individuals in the decision-making layer are changed. A better optimization algorithm is not only limited to local search capabilities, but global search capabilities are also very important. This paper, the process part adopts the IPOX cross method, and the machine part uses the two-point cross method [15]. In the mutation, the process part adopts the mutation method based on neighborhood search, as shown in Figure 5. The machine part adopts the mutation method based on random replacement of the machine at one location.

![Figure 5. Search variation based on neighborhood.](image)

3.8. GIWO Step

The details of steps can be described as follows:

Step1: Set initial population parameters;
Step2: Determine the gray wolf rank according to the fitness value of the individuals;
Step3: Individuals of gray wolf population perform invading weed algorithm for iterative optimization,
and determine the individual of $\alpha, \beta, \delta$;

Step 4: Performing variable neighborhood search in the three individuals of $\alpha, \beta, \delta$, and update the three individuals based on the results;

Step 5: Performing crossover and mutation operations on individuals, and update the whole population according Eq (7) – (15);

Step 6: Whether the algorithm meets the termination condition, if it is satisfied, go to Step 8; otherwise, go to Step 2;

Step 7: The algorithm ends and outputs the result.

4. Results & Discussion

In this section, this paper simulates the benchmark example of FJSP problem. All algorithms are coded in Matlab R2016b and run on a computer with Intel Core i5, with Windows 10 operating system.

4.1. Parameter Settings

The setting of algorithm parameters has a great impact on the overall performance. In order to achieve a better comparison, the algorithm parameters in this article are consistent with other algorithms, which is the population is 100, and the maximum iterations is 200, maximum iteration number of variable neighborhood search is 20, the cross probability is 0.8, the mutation probability is 0.2, and the $Dn$ is 4, $Dw$ is 20, $p_m$ is 50. In order to avoid random errors, each algorithm was run independently twenty times for different benchmark instances and then compared.

4.2. Performance Analysis

In this paper, the GIWO algorithm compares with the basic gray wolf algorithm (GWO), the population initialization method is added to the gray wolf algorithm, and the variable neighborhood search algorithm is embedded on the basis of the gray wolf algorithm initialization[13]. $\text{Avg}$ represents the average value of 20 experiments, $\text{dev}$ represents relative deviation, $\text{dev} = 100 \times (T_{\text{best}} - T_{\text{min}})/T_{\text{min}}$. $T_{\text{min}}$ is the minimum value and $T_{\text{best}}$ is the best value in the twenty times. As shown in Table 1, GIWO can always find the optimal solution, compared with the original GWO algorithm, this algorithm can quickly find better solutions with 10. Also, compared with the GWOIn algorithm, in this paper can find better solutions with 8 and compared with the GWOIn+vns algorithm, this paper can basically find the optimal solution accurately with 5. And the $\text{dev}$ in this paper is minimum compared with others.

| Experiment number | $n \times m$ | GWO | GWOIn | GWOIn+vns | GIWO |
|-------------------|-------------|-----|-------|-----------|------|
| MK01              | 10×6        | 42  | 45.2  | 5.0       | 42   |
| MK02              | 10×6        | 37  | 39.1  | 27.6      | 33   |
| MK03              | 15×8        | 221 | 229.7 | 8.3       | 204  |
| MK04              | 15×8        | 73  | 76.5  | 12.3      | 68   |
| MK05              | 15×4        | 179 | 188.6 | 2.3       | 181  |
| MK06              | 10×15       | 123 | 129.6 | 66.2      | 92   |
| MK07              | 20×5        | 184 | 193.4 | 23.5      | 159  |
| MK08              | 20×10       | 523 | 549.4 | 0         | 523  |
| MK09              | 20×10       | 382 | 400.4 | 17.5      | 364  |
| MK10              | 20×15       | 327 | 345.6 | 29.2      | 300  |

In this paper, GIWO algorithm also is compared with SMGAIWO [14], as shown in Table 2.

In Table 1: Comparison with other improved GWO algorithm.
It can be seen from Table 3 above that the improved algorithm in this paper has the same effectiveness as SMGAIWO in solving the benchmark examples.

This paper uses the GIWO algorithm to simulate based on the benchmark examples and calculates the optimal solution as shown in Table 3. The GIWO algorithm’s results are better than those in the TS except for the MK08 result [16]. For the results in TSPCB, in addition to the same results obtained by the calculation examples MK03 and MK10, the results obtained by other calculation examples are better [17]. For the results in PSO+TS except for the MK01, MK07, and MK08, the results obtained are better [18]. For the results in MAPSO2, the algorithm in this paper is in the calculation examples MK04 [19]. Except that the results obtained are equal, the results obtained by other calculation examples are better. It further proves the superiority of the proposed algorithm in this paper. It further proves the superiority of the proposed algorithm.

### Table 2. Comparison with SMGAIWO.

| Experiment number | SMGAIWO | GIWO |
|-------------------|---------|------|
|                   | Best    | Success rate | Best    | Success rate |
| Kacme01           | 11      | 100           | 11      | 100          |
| Kacme02           | 14      | 100           | 14      | 100          |
| Kacme03           | 11      | 100           | 11      | 100          |
| Kacme04           | 7       | 100           | 7       | 100          |
| Kacme05           | 11      | 100           | 11      | 100          |

From the Tables 1-3, the GIWO algorithm is accurate and effective in finding the optimal solution. Therefore, the proposed GIWO algorithm is effective and superiority in finding the optimal maximum completion time in solving flexible job shop scheduling.

### Table 3. Comparison results by different optimizers through Brandimarte.

| Experiment number | TS   | PSO+TS | TSPCB | MAPSO2 | GIWO |
|-------------------|------|--------|-------|--------|------|
| MK01              | 42   | 40     | 42    | 41     | 40   |
| MK02              | 32   | 32     | 28    | 26     | 32   |
| MK03              | 211  | 207    | 204   | 207    | 204  |
| MK04              | 81   | 67     | 75    | 65     | 65   |
| MK05              | 186  | 188    | 179   | 171    | 177  |
| MK06              | 86   | 85     | 69    | 61     | 84   |
| MK07              | 157  | 154    | 149   | 173    | 156  |
| MK08              | 523  | 523    | 555   | 555    | 523  |
| MK09              | 369  | 437    | 342   | 307    | 331  |
| MK10              | 296  | 380    | 242   | 312    | 242  |

From the Tables 1-3, the GIWO algorithm is accurate and effective in finding the optimal solution. Therefore, the proposed GIWO algorithm is effective and superiority in finding the optimal maximum completion time in solving flexible job shop scheduling.

### 5. Conclusions

Aiming at the flexible job shop scheduling problem, this paper studies the gray wolf optimization and invasive weed optimization algorithm and proposes a serial structure, also the population initialization method, variable neighborhood search and global search mechanism are studied. The main conclusions are as follows:

1. In the population initialization, two-stage coding based on random keys is used to realize the conversion from continuous space to discrete scheduling space. Heuristic rules, active and delay-free scheduling are incorporated. Improve the quality of the initial solution.

2. Embedding the invasive weed optimization algorithm on the basis of the gray wolf algorithm improves the excellent performance of the population, promotes the secondary evolution optimization of
the excellent individuals. The GIWO algorithm can converge quickly, greatly improving the efficiency of finding the optimal solution.

(3) Propose an adaptive variable neighborhood search structure to improve the algorithm's ability to find optimal solutions. Simulations are carried out on the FJSP benchmark examples to verify the effectiveness and superiority of the GIWO algorithm, and verify the effectiveness of solving the flexible job shop scheduling problem.

Acknowledgments
This research work is supported by the Ministry of Industry and Information Technology of the Republic of China No. CSICXX002, the National Natural Science Foundation of China under Grant No. 512045429, the Ministry of Industry and Information Technology of the Republic of China No. YYJD-003.

References
[1] Garey M R, Johnson D S, Sethi R 1976 The complexity of flowshop and jobshop scheduling Mathematics of Operations Research, 1976, (2) doi: 10.1287/moor.1.2.117
[2] Brucker P, Schlie R 1990 Job-shop scheduling with multipurpose machines Computing 45(4) 369-75
[3] Al Aqel G, Li X, Gao L 2019 Modified iterated greedy algorithm for flexible job shop scheduling problem Chinese Journal of Mechanical Engineering 32 211
[4] Gong G, Chiong R, Deng Q et al. 2020 Energy-efficient flexible flow shop scheduling with worker flexibility Expert Systems with Applications 141 112902
[5] Li J, Duan P, Cao J et al. 2018 A Hybrid pareto-based tabu search for the distributed flexible job shop scheduling problem with E/T Criteria IEEE Access 6 58883-97
[6] Mirjalili S, Mirjalili S M, Lewis A. 2014 Grey wolf optimizer Advances In Engineering Software 69 46-61
[7] Luo S, Zhang L and Fan Y. 2019 Energy-efficient scheduling for multi-objective flexible job shops with variable processing speeds by grey wolf optimization Journal of Cleaner Production 234 1365-84
[8] Lu C, Gao L, and Yi J 2018 Grey wolf optimizer with cellular topological structure Expert Systems with Applications 107 89-114
[9] Mehrabian A R and Lucas C 2006 A novel numerical optimization algorithm inspired from weed colonization Ecological Informatics 1(4) 355-66
[10] Atabaki M S, Mohammadi M, and Naderi B 2017 Hybrid genetic algorithm and invasive weed optimization via priority based encoding for location-allocation decisions in a three-stage supply chain Asia-pacific Journal of Operational Research 34 17500082
[11] Mishra S K, Bose P S C, Rao C S P 2017 An invasive weed optimization approach for job shop scheduling problems International Journal of Advanced Manufacturing Technology 91(9-12) 4233-41
[12] Zhao S K 2018 Improved Hybrid algorithm based on improved neighborhood structure for flexible job shop scheduling Computer Integrated Manufacturing System 24(12) 3060-72 (in Chinese)
[13] Jang T H 2018 Flexible job shop scheduling problem with hybrid grey wolf optimization algorithm Control and Decision 33(03) 503-8. (in Chinese)
[14] Shi X Q, Li Y Y, Deng D S, et al. 2019 Self-adaptive Multistage GA-IWO for Solving Flexible Job Shop Scheduling Problem Journal of Mechanical Engineering 55(06) 223-32 (in Chinese)
[15] Gao L, Zhang G H et al. 2012 Intelligent Algorithm for Flexible Job Shop Scheduling and It's Application (Wuhan, China: Huazhong University of Science & Technology Press) (in Chinese)
[16] Brandimarte P 1993 Routing and scheduling in a flexible job shop by tabu search Annals of
Operations Research 41(3) 157-83
[17] Ziaee M 2014 A heuristic algorithm for solving flexible job shop scheduling problem International Journal of Advanced Manufacturing Technology 71(1-4) 519-28
[18] Henchiri A, Ennigrou M 2013 Particle Swarm Optimization Combined with Tabu Search in a Multi-agent Model for Flexible Job Shop Problem Int Conf in Swarm Intelligence (Harbin: Springer) pp 385-394
[19] Nouiri M, Bekrar A, Jemai A et al. 2018 An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem Journal of Intelligent Manufacturing 29(3) 603-15