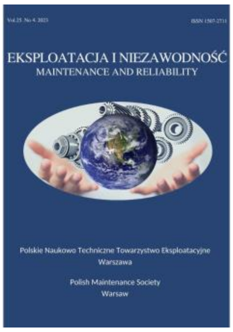


**Eksploatacja i Niezawodnosc – Maintenance and Reliability**

**Volume 25 (2023), Issue 4**

journal homepage: http://www.ein.org.pl

Article citation info:
Liang Z, Zhong P, Zhang C, Yang W, Xiong W, Yang S, Meng J, A genetic algorithm-based approach for flexible job shop rescheduling problem with machine failure interference, Eksploatacja i Niezawodnosc – Maintenance and Reliability 2023: 25(4) http://doi.org/10.17531/ein/171784

# A genetic algorithm-based approach for flexible job shop rescheduling problem with machine failure interference

Indexed by:

**Zhongyuan Liang[a], Peisi Zhong[a,*], Chao Zhang[a], Wenlei Yang[b], Wei Xiong[c], Shihao Yang[a], Jing Meng[d]**

[a] Advanced Manufacturing Technology Center, Shandong University of Science and Technology, China
[b] Qingdao Innovation and Development Base, Harbin Engineering University, China
[c] China Railway Jinan Bureau Group Co., Ltd, China
[d] Qingdao Haier Air Conditioner Co., Ltd, China

## Highlights

- Based on genetic algorithm, a complete and detailed method for solving FJSP is proposed.
- Rescheduling strategy for FJSP in dynamic environment is established.
- The complete rescheduling solution results in better results than right-shift rescheduling.
- The proposed method can make an effective response to the flexible job-shop rescheduling with machine failure interference.



## Abstract

Rescheduling is the guarantee to maintain the reliable operation of production system process. In production system, the original scheduling scheme cannot be carried out when machine breaks down. It is necessary to transfer the production tasks in the failure cycle and replan the production path to ensure that the production tasks are completed on time and maintain the stability of production system. To address this issue, in this paper, we studied the event-driven rescheduling policy in dynamic environment, and established the usage rules of right-shift rescheduling and complete rescheduling based on the type of interference events. And then, we proposed the rescheduling decision method based on genetic algorithm for solving flexible job shop scheduling problem with machine fault interference. In addition, we extended the "mk" series of instances by introducing the machine fault interference information. The solution data show that the complete rescheduling method can respond effectively to the rescheduling of flexible job shop scheduling problem with machine failure interference.

**Keywords**
genetic algorithm, rescheduling, machine failure, flexible job shop scheduling.

## 1. Introduction

As the core of product production process management decision-making stage, shop scheduling plays a key role in realizing intelligent manufacturing industry and digital production management. Obtaining excellent scheduling solution through scientific scheduling decision theory is helpful to excavate the production capacity of existing production resources and improve the utilization rate of job shop production equipment, which has important theoretical significance and engineering application value to shorten the production cycle and save the production cost of enterprises.

The highly integrated function of production equipment is a major feature of production equipment in manufacturing industry, such as CNC machine tools, machining centers, etc. One equipment can be competent for a variety of processing

(*) Corresponding author.
E-mail addresses: Z. Liang (ORCID: 0000-0002-8372-3815) zyliang_sdust@163.com, P. Zhong (ORCID: 0000-0003-3162-8098) pszhong_sdust@163.com, C. Zhang (ORCID: 0000-0002-6770-4294) cc_sdust@163.com, W. Yang wlyang_heu@163.com, W. Xiong xiongwei_crjinan@163.com, S. Yang shyang_sdust@163.com, J. Meng mj_mengjing@126.com

needs, to some extent, reduce the cost of enterprise equipment procurement and workshop area. The scheduling problem in this flexible production environment has gradually become a research hotspot. At the same time, researchers have designed a variety of methods to solve the flexible job shop scheduling problem (FJSP). De Giovanni and Pezzella[8] proposed an improved genetic algorithm to solve FJSP, which a new local search-based operator was used to improve the quality of the available solutions by optimizing the most potential individuals in each generation. Ziaee[35] developed an efficient heuristic based on a constructive procedure to obtain high-quality schedules very quickly and it can be used to improve the quality of the initial feasible solution when solving a problem by a metaheuristic algorithm, since choosing a good initial solution is an important aspect that affects the performance of the algorithm. Xing et al.[31] proposed a co-evolutionary algorithm, which combined ant colony algorithm and genetic algorithm. The two algorithms evolved their respective populations independently to improve the performance of solving FJSP. Sun et al.[28] considered FJSP with uncertain processing time represented by fuzzy numbers, and combined particle swarm optimization with the genetic algorithm to improve the convergence ability. Zeng and Wang[33] took particle swarm optimization algorithm as the operator to embed into manual immune algorithm for maintaining the diversity of population and prevent obtaining local optimal solution in solving FJSP. Denkena et al.[9] used the concept of quantum computing based optimization for FJSP and the new approach demonstrated the good performance and practicability in the application to a realistic use-case. Li and Lei[21] developed an imperialist competitive algorithm with feedback to solve the multi-objective optimization problem of FJSP. Li and Gao[20] proposed a multi strategy slime mould algorithm named GCSMA for global optimization and the simulation experiment was verified that GCSMA can be effectively applied to FJSP, and the optimization results were satisfactory. Huo and Wang[17] proposed a hybrid dynamic scheduling method with digital twin and improved bacterial foraging algorithm. Sharifi and Taghippour[27] respectively used genetic algorithm, simulated annealing algorithm and teaching-learning-based optimization algorithm to solve the scheduling problem and proved the superiority of genetic algorithm in solving the scheduling problem by enumeration method. Similarly, the solution method based on genetic algorithm has also been adopted in some literatures and has shown its superiority in solving the job shop scheduling problem and other combinatorial optimization problems[10,14,16,19,23,24,26,32].

In static scheduling, all manufacturing resources are persistent, that is, it is assumed that the production environment is an ideal interference-free environment, and the machines can run continuously according to the original scheduling plan. However, in real production, the manufacturing system will encounter unexpected disturbances, such as machine breakdowns and emergency orders. In this case, the scheduled schedule will lose its optimality or even become inexecutable.[7,11,25] The significance of rescheduling is to formulate the corresponding rescheduling scheme through the re-selection of machines to deal with the deterioration of the initial scheduling scheme caused by interference factors.

Ghaleb et al.[12] considered processing times and energy consumption affected by machine deterioration and failures, built maintenance and scheduling decisions based on the machine’s degradation level, and proposed an effective genetic algorithm for solving. Wang et al.[30] studied the scheduling problem for the flexible manufacturing systems under uncertain machine failure disruptions and proposed a robust scheduling optimization model based on the concept of threshold scenario to achieve a set of production due-date requirements as well as possible. Tubilla and Gersgwin[29] studied a variety of scheduling policies in a failure-prone machine and shed light on the most adequate operating conditions for their implementation. Azimpoor[5] proposed a branch and bound algorithm that restricted the search time and space and demonstrated the effectiveness of integrating inspection and maintenance operations with the jobs sequence to minimize the expected makespan. Gui et al.[13] proposed a scheduling method based on deep reinforcement learning so that it could offer better scheduling performance than using an individual SDR in solving dynamic flexible job shop scheduling problem. An et al focused on the integrated optimization of real-time order acceptance and flexible job-shop rescheduling with multi-level imperfect maintenance constraints[1], addressed an adaptive flexible job-shop rescheduling problem with real-time order acceptance and condition-based preventive maintenance[2], studied an integrated optimization problem of condition-based preventive maintenance and production rescheduling with multi-phase processing speed selection and old machine scrap[4], and researched the joint optimization of preventive maintenance and flexible job-shop rescheduling with processing speed selection, and the dynamic arrival of the new machine is considered to enhance productivity[3].

With the motivations noted above, we considered the machine flexibility in real working environment, in this paper, we studied the event-driven rescheduling policy in dynamic environment, established the use rules of right-shift rescheduling and complete rescheduling based on the type of interference events and on the basis, we proposed the rescheduling decision method based on genetic algorithm for solving flexible job shop scheduling problem with machine fault interference. In addition, we extended the "mk" series of instances by introducing the machine fault interference information. The solution data show that the complete rescheduling method can respond effectively to the rescheduling of FJSP with machine failure interference.

The rest of this paper is organized as follows. Section 2 builds the mathematical mode of flexible job shop scheduling problem and the solution flow of genetic algorithm for it. Section 3 introduces the proposed method of rescheduling with machine failure interference. Section 4 presents the experimental results and analysis. Finally, the conclusion and future work are given in Section 5.

## 2. Genetic algorithm for flexible job shop scheduling problem

### 2.1. Problem description and modeling

Flexibility generally refers to the flexibility of the machine, that is, in the workpiece to be processed, all the workpiece contains multiple processes, there are multiple machines in the processing system, each process can choose multiple processing machines, but only one machine can be selected for processing, the same process can choose to process in different machines, then there will inevitably be different processing time. The scheduling problem is flexible job-shop scheduling problem. A schematic diagram of the flexible job-shop scheduling problem is shown in Fig. 1.

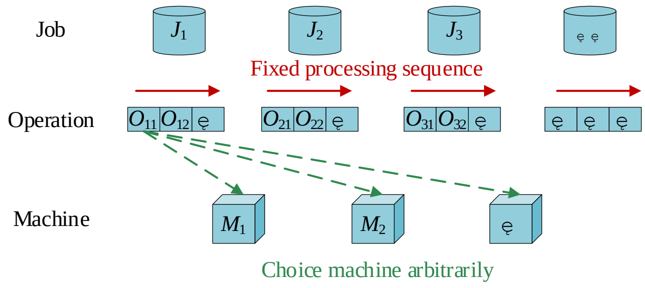

Fig. 1. Tournament selection operator.

Table 1 shows an instance of flexible job-shop scheduling problem (FJSP). This FJSP instance contains 3 jobs, each job contains 3 operations, and the processing system contains 3 machines, the average flexibility of the machine $r$=2.222.

Different from job shop scheduling, the processing machine of an operation in flexible job shop scheduling is not unique, that is, it can be selected from several machines. Due to the machine selectivity of the processing for operations, there is a combinatorial explosion in the scheduling solution space, and the difficulty of the solution increases sharply. The scale of the flexible job-shop scheduling problem can be simply calculated as $n\times o\times m$, where $n$ is the number of jobs, $o$ is the average number of operations included in each part, and $m$ is the number of machines. Then, for a flexible job-shop scheduling problem with scale $n\times o\times m$, the number of all solutions in the solution space is $(n\times o\times m)\times(n\times o)!/(o!)^n$, where $r$ is the average flexibility of the machine, that is, the average number of selectable machines for each operation.

Table 1. An instance of FJSP.

| Jobs | Operations | Machines | | |
|---|---|---|---|---|
| | | $M_1$ | $M_2$ | $M_3$ |
| $J_1$ | $O_{11}$ | 2 | 4 | - |
| | $O_{12}$ | - | 3 | 6 |
| | $O_{13}$ | 7 | 6 | 5 |
| $J_2$ | $O_{21}$ | 7 | 8 | - |
| | $O_{22}$ | - | 9 | 7 |
| | $O_{23}$ | 7 | 5 | 6 |
| $J_3$ | $O_{31}$ | - | 5 | 8 |
| | $O_{32}$ | 7 | 8 | - |
| | $O_{33}$ | - | 8 | 9 |

For the flexible job-shop scheduling problem, the parameters are defined:

$$x_{ijh}=\begin{cases}1 & \text{If operation } O_{jh} \text{ choose machine } M_i\\ 0 & \text{Otherwise}\end{cases} \tag{1}$$

$$y_{ijhkl}=\begin{cases}1 & \text{If operations } O_{jh} \text{ on machine } M_i\\ & \text{is processed before operation } O_{kl}\\ 0 & \text{Otherwise}\end{cases} \tag{2}$$

then the scheduling model has the following constraints:

$$s_{jh} + x_{ijh} \times p_{ijh} \leq c_{jh} \tag{3}$$

where, $S_{jh}$ is the start processing time of operation $O_{jh}$, $p_{ijh}$ is the processing time of operation $O_{jh}$ on machine $M_i$, and $c_{ji}$ is the completion processing time of operation $O_{jh}$.

$$c_{jh} \leq s_{j(h+1)} \tag{4}$$

$$c_{jh_j} \leq C_{max} \tag{5}$$

where, $j$=1, 2, ..., $n$, $h_j$ is the number of operations that the job $J_j$ contains.

$$s_{jh} + p_{ijh} \leq s_{kl} + L\left(1 - y_{ijhkl}\right) \tag{6}$$

$$c_{jh} \leq s_{j(h+1)} + L\left(1 - y_{iklj(h+1)}\right) \tag{7}$$

where, $L$ is a sufficiently large positive number.

$$\sum_{i=1}^{m_{jh}} x_{ijh} = 1 \tag{8}$$

In the above equations, Eqs. (3) and (4) describe the process sequence constraints inside jobs, Eq. (5) describes the jobs' completion time constraints, that is, the completion time of each operation should not exceed the makespan of the scheduling scheme, Eqs. (6) and (7) constrain that only one operation can be processed by the same machine at the same time. Eq. (8) restricts that the same operation can only be processed by one machine at the same time.

## 2.2. Basic solution flow of GA

The steps of genetic algorithm[15,22,34] are similar to the evolution process of species in nature: firstly, the feasible solution of a problem is encoded to form an initialized population, and secondly, the fitness of individuals in the population is evaluated by the objective function, and then the individuals in the population are selected according to some rules. Let the selected individuals cross over and pass on the excellent genes to their offspring, at the same time, in order to meet the diversity of "genes", individuals are allowed to "mutate" according to certain rules. Repeat the above process until the termination condition is reached. The flow chart of GA with specific execution methods for solving FJSP is shown in Fig. 2.

**Encode:** The chromosome of a complete scheduling scheme consists of a machine chain and an operation chain. The value of the gene on machine chain represents the machine number selected by the operation, the gene value on operation chain represents the job number, and the sequence of occurrence of the same gene value represents the operation's serial number in the job.

**Initial population:** The machine selection of partial FJSP (P-FJSP) is irregular, such as the instance in Table 1. Kacem[18] set the processing time of unselected machines to "999" during encoding and P-FJSP was converted into total FJSP(T-FJSP), which makes the algorithm of encoding machine chains more general, and this approach has also been adopted in some subsequent literatures. Although the elimination mechanism of GA can eliminate the cases of non-selectable machine is selected, it increases a lot of redundant information, and increases the amount of calculation and search difficulty. Therefore, this paper designs a method of machine chain initialization for P-FJSP. The optional machines of all processes are counted and stored in "*Ms_celldata*", and the index value is randomly generated during coding to generate machine chain. The details are given in Algorithm 1.

Algorithm 1. Machine chain initialization algorithm for FJSP.

**Algorithm 1**

**Input:** *Ms_celldata* –An array of cells for optional machines;
*pop* –Population size;
**Output:** *M_chromosome* –Machine chain;

```
for ( i←1; i≤pop )
for ( j←1; j≤No. of all operations )
OM= Ms_celldata{j};    // Get the set of selectable machines for the current operation
randindex= randperm(length(OM),1);   // Randomly generate the current operation to select the machine index
M_chromosome(i,j)= OM (randindex);  // Generate machine chain gene value
end for
end for
output(M_chromosome);    // Output machine chain
```

**Decode:** We designed two counters to aid in decoding. *J_count* records the number of decoded operations within the job to which the current decoding operation belongs. *M_count* records the number of decoded operations on the machine to which the current decoding operation belongs. The specific decoding rules are shown in Fig. 2, where *M_num* denotes the machine number where the current decoding operation is processed and *M_op_num* denotes the machine number where the previous operation within the current decoding operation's job is processed.

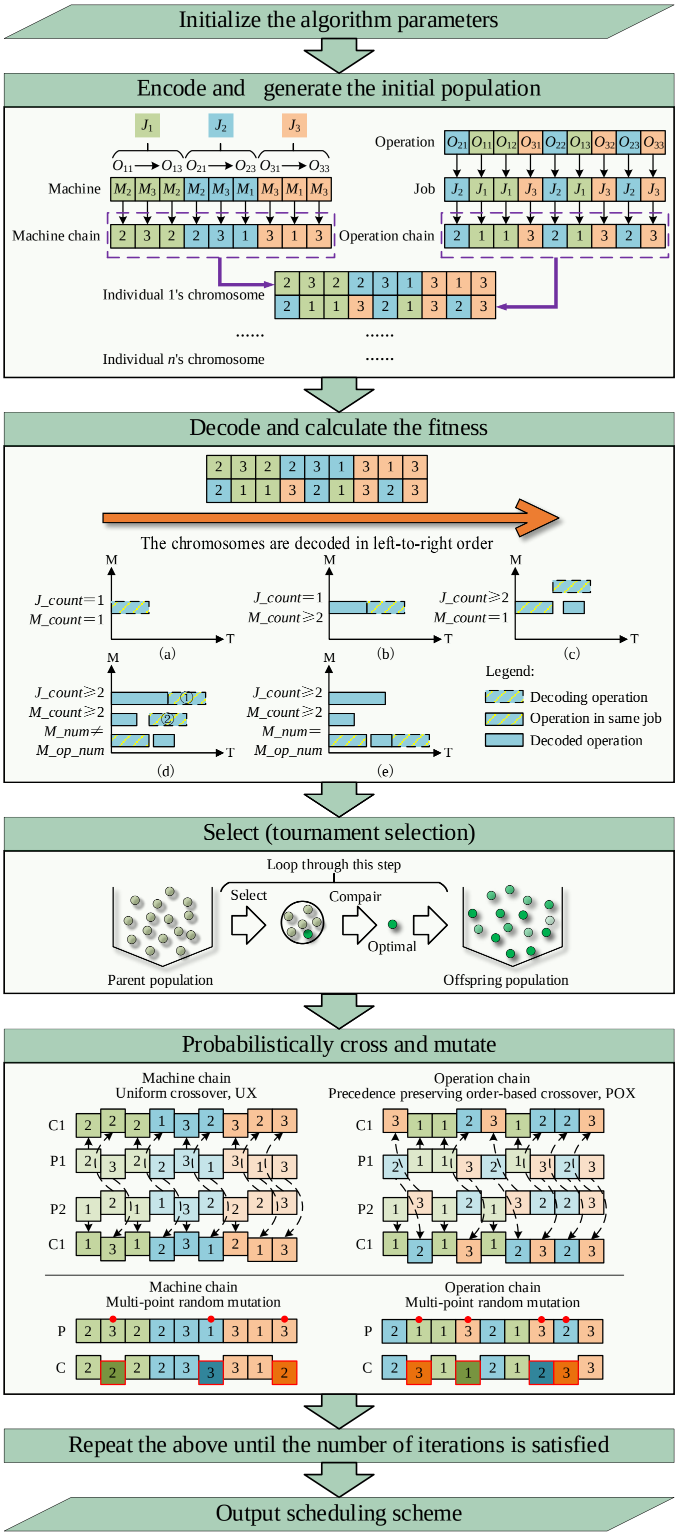


Fig. 2. Flow chart of GA for solving FJSP.

**Select:** The tournament selection operator is adopted because of the advantage that it does not need to convert the fitness calculation, and can be achieved by comparing the value of the objective function. Select $k$ individuals from the population randomly, and determine the optimal individual among the $k$ individuals across their fitness, then put the first $\lambda$th optimal into the offspring population.

**Cross:** The machine chain employs a uniform crossover operator, the basic idea of which is to exchange gene values in situ according to randomly generated crossover points. A set of jobs is randomly generated, and all genes representing the job are passed to the offspring in situ, and the remaining genes are replenishable to the offspring in order.

**Mutate:** Multi-point random mutation operator is used for both machine chain and operation chain. For machine chain, the genes at mutation points mutate to appropriate values according to the alternative machines of the operations. For operation chain, take out the genes at the mutation points, re-sort randomly and fill them back to the original position.

## 3. Method

### 3.1. Rescheduling strategy in dynamic environment

Periodic rescheduling is a scheduling method that assigns tasks to resources periodically based on rolling horizon. In essence, the static scheduling is divided into multiple scheduling time Windows, and the static scheduling is implemented in each time window. This scheduling method has high robustness for the production system, when the disturbance occurs, the scheduling system can make a timely response. The smaller the time window, the more aggressive the response, and the more computationally intensive it is. The disadvantage is that when no external disturbance occurs, unnecessary computation will be generated, and the optimum in the time window is a local optimum, which cannot represent the global optimum

Event-driven rescheduling is a scheduling method that the scheduling system regenerates the scheduling scheme when the external disturbance occurs. In the production environment where interference events do not occur frequently, this scheduling method can save computing resources and respond positively to interference events. Event-driven rescheduling includes:

(1) Right shift rescheduling

After the occurrence of the disturbance event, such as order insertion or machine failure, a simple rescheduling method is right shift, that is, the subsequent operations on the time node of the machine where the disturbance event occurred are delayed, which is essentially to delay the related links in the production system without taking any measures for the disturbance event. When the interference duration is small, the idle time of the machine in the scheduling scheme has the ability to absorb the interference factors, and has little impact on the makespan of the overall scheduling scheme. When the interference duration is large, the idle time of the machine cannot absorb the processing delay caused by the interference, which will cause the overall scheduling scheme to produce tardiness, resulting in production delay. The principle of right-shift rescheduling is shown in Fig. 3.

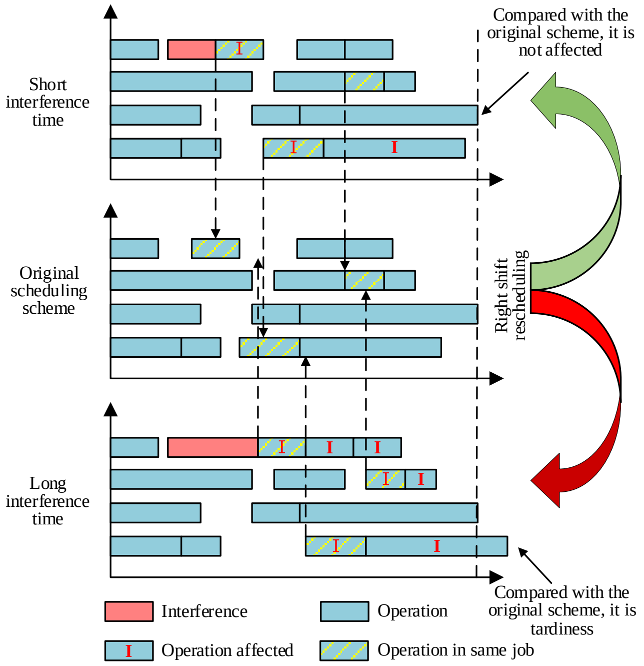

Fig. 3. Schematic diagram of right shift rescheduling.

(2) Complete rescheduling

In order to reduce the delay effect caused by interference, another rescheduling method is to replan the path of the subsequent unprocessed operations of the interference node, that is, complete rescheduling based on path change. For operations that have been processed and are being processed, the original scheduling scheme is kept unchanged. For operations that have not been processed, the machine selection and process sequencing are rescheduled. The principle of full rescheduling based on path changes is shown in Fig. 4. According to the type of disturbance event, it is necessary to make assumptions about the operation being processed by the disturbed machine. The workpiece being processed must be processed on the disturbed machine before it can be rescheduled. If the machine is disturbed by failure, the processing of the operation in process will stop immediately on the disturbed machine and need to be reprocessed on other machines under rescheduling.

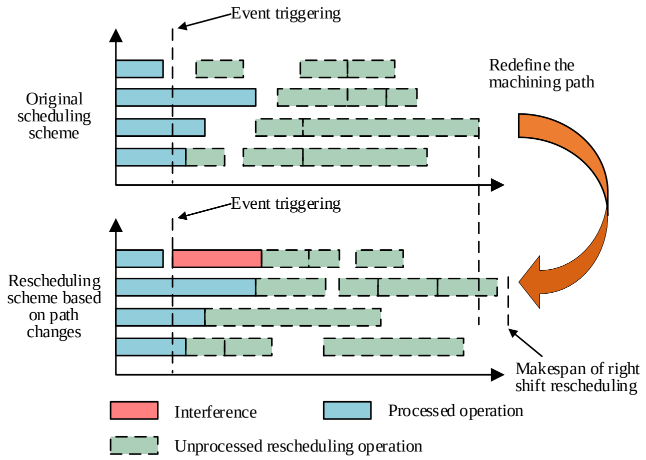

Fig. 4. Schematic diagram of complete rescheduling.

In general, right shift rescheduling has a smaller amount of computation than complete rescheduling, and may have the characteristics of absorbing less interference. However, when the machine fault is unrepairable, the initial scheduling scheme cannot be completed, and the processing path of the unprocessed parts on the faulty machine must be replanned. Complete rescheduling has a better solution than right shift rescheduling in theory, but the computation is relatively large. In the event-driven rescheduling strategy, according to the advantages and disadvantages of right shift rescheduling and complete rescheduling, a more reasonable rescheduling strategy is selected according to the type of interference events in dynamic environment, which is more helpful to reduce the interference of dynamic events to the original production scheduling scheme of the workshop. The flowchart of dynamic rescheduling is shown in Fig. 5.

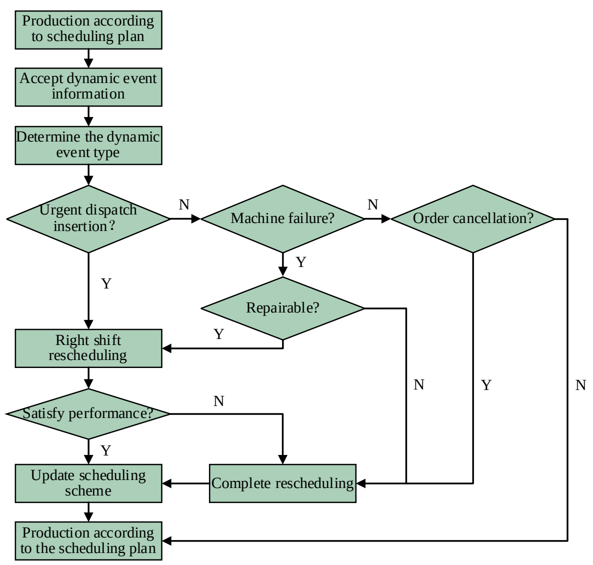

Fig. 5. Flow chart of dynamic rescheduling.

### 3.2. Rescheduling decision method for FJSP with machine failure interference

The FJSP rescheduling problem in dynamic environment considers a variety of disturbance factors, which are random and uncertain, and make the production mobilization process fluctuate. The above sections give the selection rules of scheduling strategies according to different working conditions. Considering the most common machine failure interruption factor in production scheduling, this paper studies the specific implementation method of the complete rescheduling strategy, and adopts the studied complete rescheduling method to reduce the impact of machine failure disturbance.

#### 3.2.1. Performance metrics and assumptions

The most direct impact caused by machine failure is the delay of the construction period, so the difference between the actual scheduling scheme and the original scheduling scheme can be used as a performance index to measure the rescheduling, as shown in Eq. (9).

$$\delta(S_{pr}) = Crp_{max_{max}} \tag{9}$$

where $S_r$ is the rescheduling scheme, $S_p$ is the original scheduling scheme, $C_{\max}$ is the makespan of the scheduling scheme, and $\delta(S_{pr})$ is the difference value of the objective function between the rescheduling scheme and the original scheduling scheme. In order to make Eq. (9) more general, the relative deviation is used to represent the difference of performance index between two scheduling schemes, as shown in Eq. (10).

$$RM = \frac{\delta(S_{pr})}{Cp_{max}} \tag{10}$$

In order to simplify the problem, the following assumptions are made for the FJSP rescheduling problem with machine failures:

(1) Only one machine is down at a time.

(2) It takes negligible time to transfer the workpiece from the failed machine to a functioning machine, and the operation needs to be reprocessed.

(3) Repair the machine immediately after its failure.

#### 3.2.2. Principle and algorithm of complete rescheduling

The optimization algorithm used to solve the rescheduling problem under machine fault interference is the same as the algorithm used to solve the initial scheduling scheme. The difference is that the rescheduling information input adds the machine fault information, that is, the machine cannot be selected during the machine fault period, and can be selected again after the fault is repaired. So, one solution cycle of rescheduling should start by determining the chromosome gene position corresponding to the faulty machine.

Fig. 6 shows the Gantt chart of the optimal solution for the FJSP instance shown in Table 1 and the machine failure situation as follows: suppose that machine $M_2$ breaks down at time 5, and the maintenance time takes 10 units' time. According to the scheduling scheme shown in Fig. 6, the chromosome coding of the scheduling scheme is first obtained, where the machine selection chain is 2-2-2-1-3-3-1-2 and the operation sequence chain is 2-1-1-1-3-2-3-2.

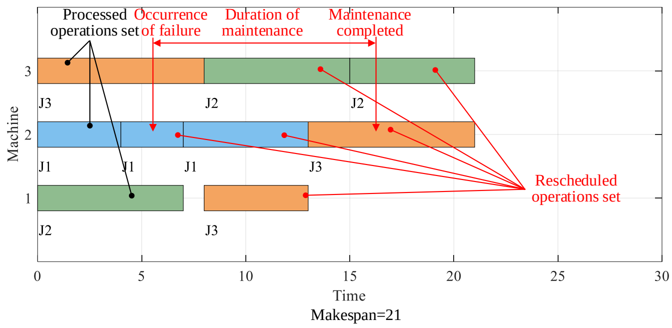


Fig. 6. Rescheduling to the instance shown in Table 1.

When the fault occurs, the operations being processed by the non-faulty machine continue to be processed, and the operations being processed by the faulty machine stop processing. The operation requires the replacement of processing machines and reprocessing. Therefore, according to the machine fault information, the processes in the original scheduling scheme are divided into two sets: the processed operations set and the rescheduled operations set. The generation of new individuals in rescheduling, that is, when chromosome re-coding and genetic operation, should meet the rules: the machine selection chain and the operation sequence gene of the processed operation set remain unchanged; rescheduling operation set machine selection chain and process operation gene replanning, and the principle of rescheduling is shown in Fig. 7.

It should be noted that in the machine selection chain of the rescheduling scheme, there may still be the case that the faulty machine is selected, such as gene "2". At this time, the scheduling scheme is generated according to the new scheduling chromosome decoding, which needs to meet the fault constraint condition, that is, the operation selecting the faulty machine in the rescheduling operation set cannot be arranged in the fault maintenance cycle.

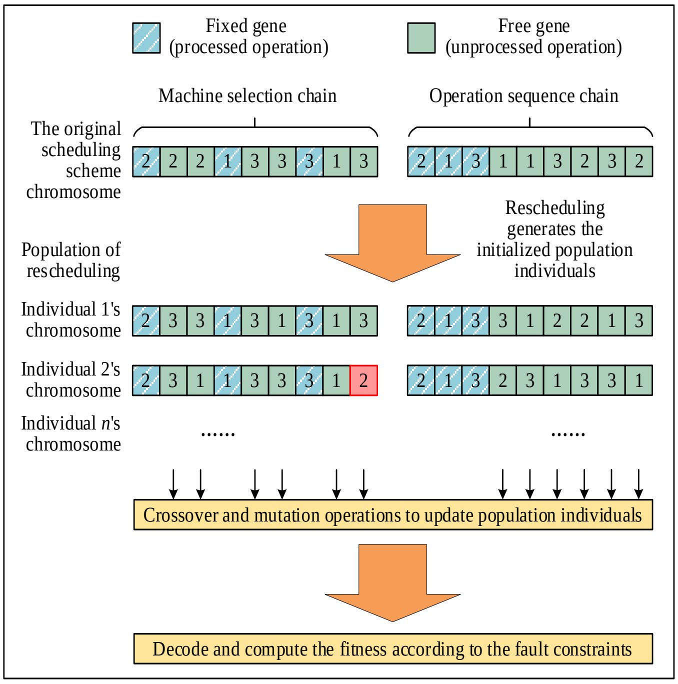


Fig. 7. Principle of rescheduling.

The chromosome decoding algorithm with machine fault interference is shown in Algorithm 2.

Algorithm 2. Decoding algorithm with fault constraints.

**Algorithm 2**

**Input:** The original chromosome;
*M* –Scheduling data matrix;
*n* –Number of jobs;
*m* –Number of machines;
*M_num* –Machine number;
*FM_num* –Faulty machine number;
*FS_time* –Failure start time;
*FE_time* –Failure end time;
*ReS_set* –Rescheduled operations set;

**Output:** *Makespan*;
*s_time* –The start time of the operations;
*e_time* –The end time of the operations;

```
for ( i←1; i≤o_num; o_num=length of operation chain )
    if O_i ∉ ReS_set then
        Decode directly;
    else
        if M_num_i ≠ FM_num then
            Decode directly;
        else
            Decode directly;
            if s_time_i ≥ FS_time and s_time_i <FE_time then
                s_time_i= FE_time;
            end if
        end if
    end if
    e_time_i=s_time_i+p_time_i;
end for
Makespan= max_{i=1}^{n_num}(e_time_i);
output (Makespan, s_time, e_time);
```

## 4. Results and discussion

In order to evaluate the performance of the proposed algorithm in solving FJSP rescheduling optimization, we extended the "mk" instances proposed by Brandimarte[6] and added the machine fault information, so as to meet the FJSP solving conditions under fault interference. The fault information includes: fault machine number (M_No.), fault start time (s_t), and fault duration (d_t). According to the previous assumption, the failure start time is the failure repair time, and it is also the rescheduling start time.

In the actual production system, the generation of machine failures satisfies the Poisson distribution and the failure probability is not large, so this paper assumes that the failure must occur to verify the solution performance of the algorithm. These faults are generated randomly, and the fault machine number obeys the uniform distribution of the machine set of the processing system. The fault start time obeying the uniform distribution between the time 0 of the original scheduling scheme and the makespan. The fault duration follows a uniform distribution between 25% and 50% completion time. For the 10 instances of the "mk" series, five sets of failure information are generated for each instance, with a total of 50 rescheduling events.

As shown in Table 2 and Fig. 8, the solution results of complete rescheduling have a significant optimization effect on makespan compared with right-shift rescheduling, and complete rescheduling results are the best among all the solution results of rescheduling events. Because the data attributes of the scheduling instance itself and the data attributes of the fault information are different in different scheduling cases, the RM value of the rescheduling scheme is also different from that of the initial optimal scheduling scheme, and it cannot be stabilized around a certain percentage, that is, "40%" may be the optimal rescheduling (Fig. 8 mk04-1), Or it may be that rescheduling is not optimal (Fig. 8 mk01-1).

In order to visually compare the results of right-shift rescheduling with complete rescheduling, the *RM* value is calculated for each group of results, and the comparison results are shown in Fig. 8.

The machine fault information and rescheduling results are shown in Table 2.

Table 2. Rescheduling results of extended "mk" instances.

| | Instance | Scale | Original makespan | M_No. | s_t | d_t | RSR | CR |
|---|---|---|---|---|---|---|---|---|
| 1 | mk01 | 10×6 | 40 | 5 | 3 | 20 | 56 | 42 |
| | | | | 4 | 18 | 19 | 61 | 54 |
| | | | | 1 | 22 | 20 | 49 | 44 |
| | | | | 3 | 23 | 14 | 55 | 46 |
| | | | | 5 | 13 | 10 | 51 | 46 |
| 2 | mk02 | 10×6 | 26 | 1 | 15 | 12 | 43 | 40 |
| | | | | 2 | 4 | 16 | 38 | 34 |
| | | | | 6 | 18 | 9 | 35 | 32 |
| | | | | 6 | 1 | 12 | 41 | 35 |
| | | | | 4 | 2 | 9 | 39 | 37 |
| 3 | mk03 | 15×8 | 204 | 4 | 113 | 70 | 279 | 239 |
| | | | | 6 | 45 | 66 | 272 | 246 |
| | | | | 2 | 55 | 59 | 268 | 221 |
| | | | | 2 | 75 | 53 | 272 | 219 |
| | | | | 2 | 1 | 65 | 246 | 238 |
| 4 | mk04 | 15×8 | 60 | 4 | 6 | 31 | 102 | 84 |
| | | | | 3 | 1 | 17 | 84 | 77 |
| | | | | 3 | 49 | 27 | 109 | 94 |
| | | | | 2 | 30 | 17 | 84 | 72 |
| | | | | 2 | 11 | 19 | 87 | 75 |
| 5 | mk05 | 15×4 | 172 | 2 | 30 | 81 | 286 | 227 |
| | | | | 3 | 116 | 50 | 230 | 225 |
| | | | | 2 | 84 | 48 | 229 | 206 |
| | | | | 4 | 124 | 48 | 230 | 216 |
| | | | | 3 | 28 | 48 | 234 | 210 |
| 6 | mk06 | 10×15 | 58 | 7 | 6 | 30 | 97 | 83 |
| | | | | 7 | 57 | 33 | 93 | 79 |
| | | | | 8 | 25 | 25 | 85 | 67 |
| | | | | 8 | 37 | 26 | 88 | 73 |
| | | | | 8 | 18 | 43 | 103 | 87 |
| 7 | mk07 | 20×5 | 139 | 1 | 43 | 59 | 226 | 200 |
| | | | | 5 | 112 | 77 | 244 | 221 |
| | | | | 5 | 8 | 73 | 237 | 208 |
| | | | | 2 | 65 | 75 | 240 | 196 |
| | | | | 4 | 52 | 75 | 244 | 199 |
| 8 | mk08 | 20×10 | 523 | 7 | 292 | 250 | 775 | 604 |
| | | | | 7 | 125 | 192 | 715 | 582 |
| | | | | 1 | 94 | 153 | 681 | 642 |
| | | | | 3 | 242 | 185 | 701 | 577 |
| | | | | 9 | 86 | 207 | 727 | 567 |
| 9 | mk09 | 20×10 | 307 | 2 | 189 | 132 | 567 | 413 |
| | | | | 7 | 244 | 97 | 531 | 488 |
| | | | | 10 | 68 | 107 | 441 | 382 |
| | | | | 9 | 50 | 94 | 424 | 379 |
| | | | | 1 | 115 | 97 | 423 | 352 |
| 10 | mk10 | 20×15 | 197 | 8 | 1 | 148 | 363 | 231 |
| | | | | 9 | 57 | 79 | 287 | 245 |
| | | | | 9 | 88 | 132 | 306 | 237 |
| | | | | 1 | 203 | 130 | 316 | 244 |
| | | | | 1 | 41 | 86 | 285 | 221 |

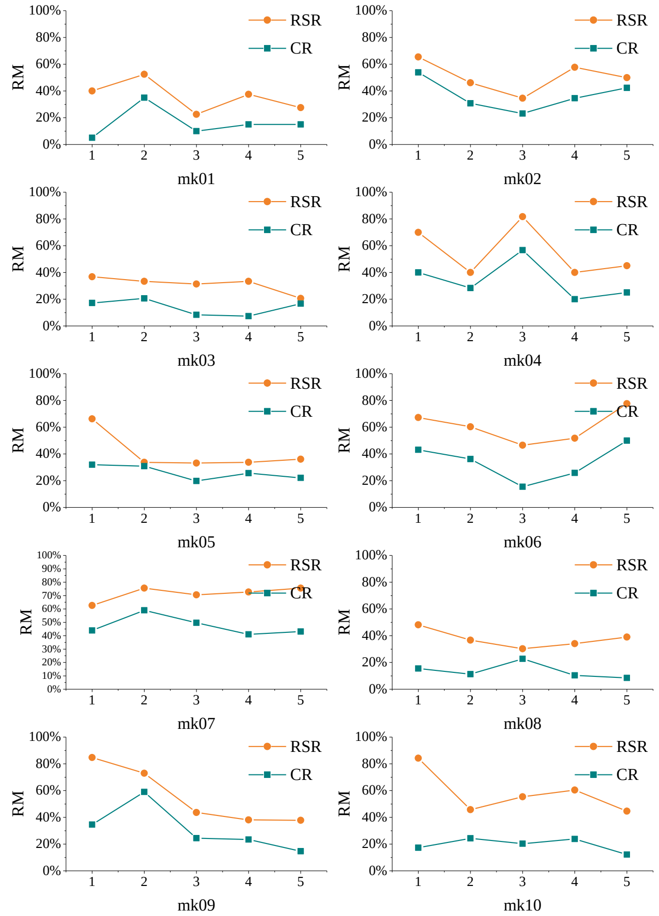


Fig. 8. Comparison of rescheduling results.

The solution result of right shift rescheduling is greatly affected by the fault duration, and the makespan increment of its rescheduling mostly swings around the value of the fault duration. For the mk01-1 rescheduling event in Table 2, the original optimal scheduling solution is 40, the failure occurs at time 3 on machine $M_5$, the failure duration is 20, and the final right shift rescheduling result is 56, and the added value is 16, which originates from the fact that the idle time of the machine in the original scheduling scheme absorbed the partial maintenance time of the machine failure (e.g., at time 3-5, 8-11 and 14-21, Fig. 9).

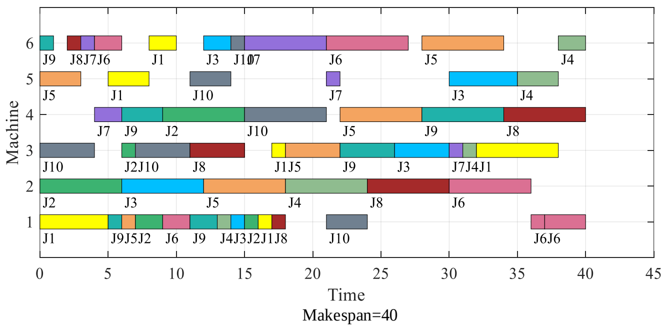


Fig. 9. Gantt chart of the original optimal scheduling for mk01

However, the result of complete rescheduling (Makespan=42) is significantly better than right shift rescheduling and has little difference on the makespan compared to the original scheduling scheme. For example, in the mk01-2 rescheduling event in Table 2, machine $M_4$ broke down at time 18, the fault duration is 19, and the final right shift rescheduling result is 54, and the added value is 21, which is due to the fact that the machine failure happened at the time of machine processing, so the operation needed to be reprocessed in rescheduling. At this time, the 5th operation of job $J_{10}$ is being processed (Fig. 9). Based on the solution results of 50 rescheduling events in Table 2, compared with the original scheduling scheme, the average delay ratio of right shift rescheduling is 49.70%, and the average delay ratio of complete rescheduling is 27.29%.

Figs. 10-14 list the Gantt chart of the resulting right shift rescheduling versus complete rescheduling for the extended mk01 rescheduling event in Table 2. The rescheduling Gantt chart of mk01-1 in Fig. 10 shows that, for machine $M_5$ which is greatly affected by fault interference, the 2nd operation of job $J_1$ and the 3rd operation of job $J_{10}$ move to the right directly leads to the delay of subsequent operations.

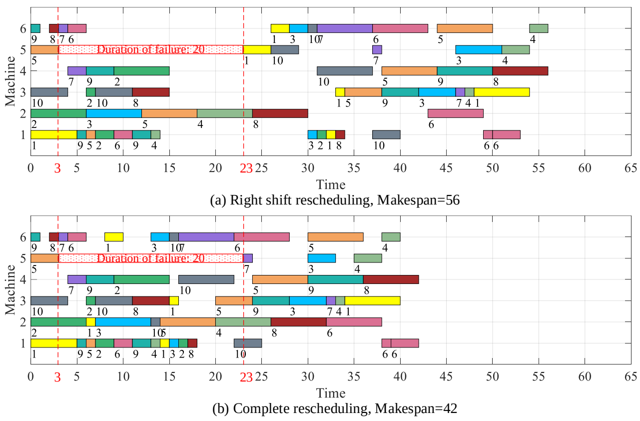


Fig. 10. Rescheduling Gantt charts of mk01-1.

In the complete rescheduling scheme, due to the processing flexibility of the machine, these two operations are assigned to machine $M_2$ for processing. This makes the processing start time delay of subsequent processes less influential. The 4th operation of the job $J_7$ on machine $M_5$ is also disturbed by the fault, but the delayed start of the operation $O_{74}$ does not affect the global scheduling scheme because there is a lot of idle time and the start processing time of the immediately after operation (the job $J_7$ on machine $M_3$) is far behind the end processing time of the operation.

In the complete rescheduling scheme shown in Fig. 11, operations $O_{10,5}$, $O_{54}$ and $O_{96}$ are all moved to machine $M_2$ for processing.

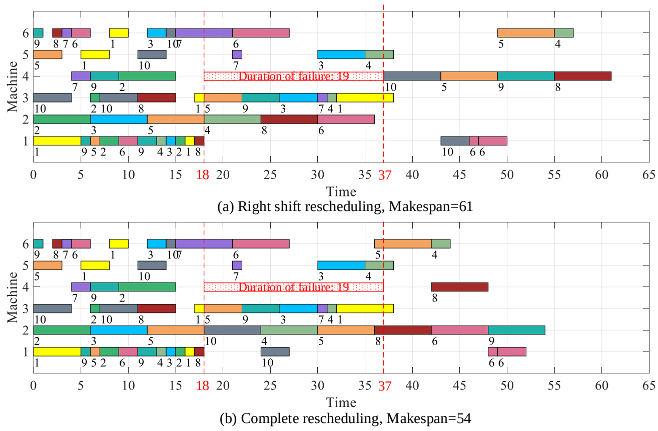


Fig. 11. Rescheduling Gantt charts of mk01-2.

In the complete rescheduling scheme shown in Fig. 12, operations $O_{10,6}$, $O_{66}$ and $O_{65}$ are respectively moved to machine $M_4$, $M_4$ and $M_2$ for processing.

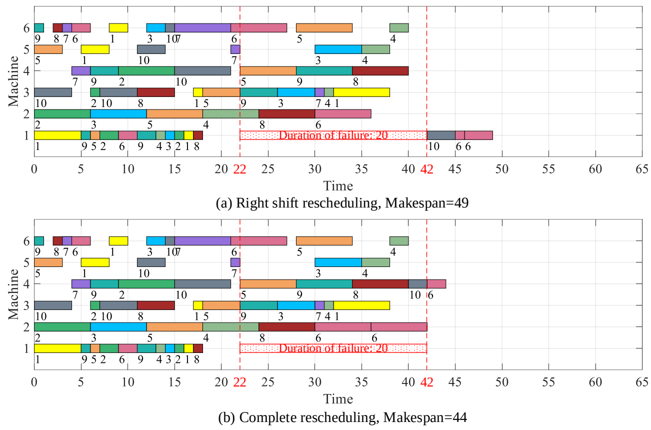


Fig. 12. Rescheduling Gantt charts of mk01-3.

In the complete rescheduling scheme shown in Fig. 13, operations $O_{95}$, $O_{34}$ and $O_{16}$ are respectively moved to machine $M_6$, $M_3$ and $M_4$ for processing.

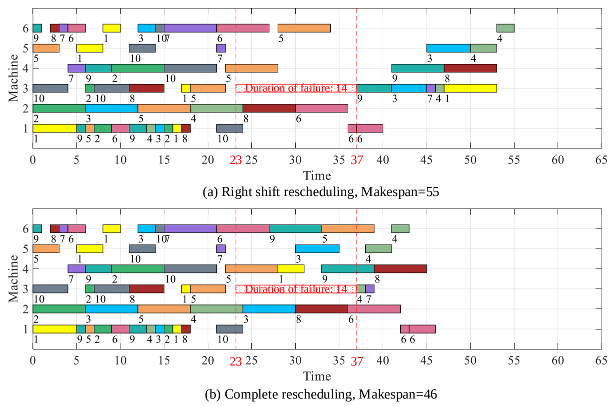


Fig. 13. Rescheduling Gantt charts of mk01-4.

In the complete rescheduling scheme shown in Fig. 14, operations $O_{10,3}$, $O_{10,5}$ and $O_{66}$ are respectively moved to machine $M_2$, $M_2$ and $M_4$ for processing.

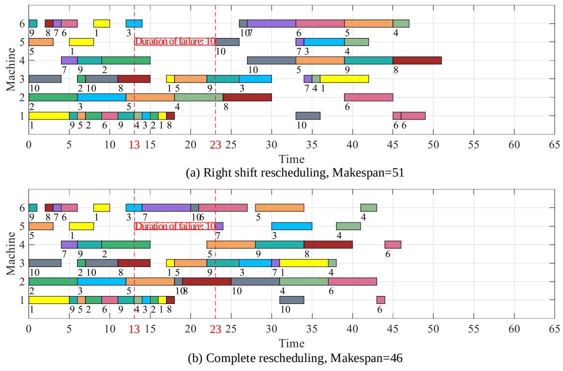

Fig. 14. Rescheduling Gantt charts of mk01-5.

It can be seen from the comparison of Gantt charts from Fig. 10 to Fig. 14 that the complete rescheduling transfers the operations originally processed in the failure maintenance time to other machines for processing, and replans the processing path of the operations in the rescheduling operation set, so as to minimize the makespan of the rescheduling scheme. The test results show that the complete rescheduling method based on the proposed method can respond effectively to the flexible job shop rescheduling problem with machine fault interference.

## 5. Conclusion

In this paper, we studied a rescheduling method based on genetic algorithm for FJSP with machine failure, thus aiding the operational reliability of robust shop floor production systems. To be more precise: the mathematical model of FJSP, the detailed process of solving FJSP by genetic algorithm and the event-driven rescheduling policy in dynamic environments are established; and then, the rescheduling decision method based on genetic algorithm for FJSP with machine fault interference is proposed and verified by extended instances. The test solution data show that the average delay ratio of right-shift rescheduling is 49.70%, and the average delay ratio of complete rescheduling is 27.29%, which leads to the conclusion: complete rescheduling is superior than right-shift rescheduling, and the proposed complete rescheduling decision method can effectively respond to the rescheduling solution of FJSP with machine fault interference.

As the machine ages, production system machine failures and maintenance problems become com-mon. Combining the event-triggered rescheduling theory in this paper with the condition-based preventive maintenance is an effective means to ensure the stability of the production system. The future work is to predict the machine maintenance according to the running state of the machine, and combine the re-scheduling method in this paper to improve the maintenance function module of the production and manufacturing system.

## References


1. An Y, Chen X, Gao K, Zhang L, Li Y, Zhao Z. Integrated optimization of real-time order acceptance and flexible job-shop rescheduling with multi-level imperfect maintenance constraints. Swarm and Evolutionary Computation 2023; 77: 101243, https://doi.org/10.1016/j.swevo.2023.101243.
2. An Y, Chen X, Gao K, Zhang L, Li Y, Zhao Z. A hybrid multi-objective evolutionary algorithm for solving an adaptive flexible job-shop rescheduling problem with real-time order acceptance and condition-based preventive maintenance. Expert Systems with Applications 2023; 212: 118711, https://doi.org/10.1016/j.eswa.2022.118711.
3. An Y, Chen X, Hu J, Zhang L, Li Y, Jiang J. Joint optimization of preventive maintenance and production rescheduling with new machine insertion and processing speed selection. Reliability Engineering & System Safety 2022; 220: 108269, https://doi.org/10.1016/j.ress.2021.108269.
4. An Y, Chen X, Hu J, Zhang L, Zhao Z. Integrated optimization of condition-based preventive maintenance and production rescheduling with multi-phase processing speed selection and old machine scrap. Reliability Engineering & System Safety 2023; 238: 109399, https://doi.org/10.1016/j.ress.2023.109399.
5. Azimpoor S, Taghipour S, Farmanesh B, Sharifi M. Joint planning of production and inspection of parallel Machines with two-phase of failure. Reliability Engineering & System Safety 2022; 217: 108097, https://doi.org/10.1016/j.ress.2021.108097.
6. Brandimarte P. Routing and scheduling in a flexible job shop by tabu search. Annals of Operations Research 1993; 41(3): 157-183, https://doi.org/10.1007/BF02023073.
7. Chrétienne P. Maximizing the number of jobs scheduled at their baseline starting times in case of machine failures. Journal of Scheduling 2020; 23(1): 135-143, https://doi.org/10.1007/s10951-019-00621-x.

8. De Giovanni L, Pezzella F. An improved genetic algorithm for the distributed and flexible job-shop scheduling problem. European Journal of Operational Research 2010; 200(2): 395-408, https://doi.org/10.1016/j.ejor.2009.01.008.
9. Denkena B, Schinkel F, Pirnay J, Wilmsmeier S. Quantum algorithms for process parallel flexible job shop scheduling. Cirp Journal of Manufacturing Science and Technology 2021; 33: 100-114, https://doi.org/10.1016/j.cirpj.2021.03.006.
10. Driss I, Mouss KN, Laggoun A. A new genetic algorithm for flexible job-shop scheduling problems. Journal of Mechanical Science and Technology 2015; 29(3): 1273-1281, https://doi.org/10.1007/s12206-015-0242-7.
11. Ghaleb M, Taghipour S. Dynamic shop-floor scheduling using real-time information: A case study from the thermoplastic industry. Computers & Operations Research 2023; 152: 106134, https://doi.org/10.1016/j.cor.2022.106134.
12. Ghaleb M, Taghipour S, Sharifi M, Zolfagharinia H. Integrated production and maintenance scheduling for a single degrading machine with deterioration-based failures. Computers & Industrial Engineering 2020; 143: 106432, https://doi.org/10.1016/j.cie.2020.106432.
13. Gui Y, Tang D, Zhu H, Zhang Y, Zhang Z. Dynamic scheduling for flexible job shop using a deep reinforcement learning approach. Computers & Industrial Engineering 2023; 180: 109255, https://doi.org/10.1016/j.cie.2023.109255.
14. Guo K, Yang M, Zhu H. Application research of improved genetic algorithm based on machine learning in production scheduling. Neural Computing and Applications 2020; 32(7): 1857-1868, https://doi.org/10.1007/s00521-019-04571-5.
15. Holland JH. Adaptation in natural and artificial systems. Ann Arbor: The University of Michigan Press; 1975.
16. Huang J, Wang L, Jiang Z. A method combining rules with genetic algorithm for minimizing makespan on a batch processing machine with preventive maintenance. International Journal of Production Research 2020; 58(13): 4086-4102, https://doi.org/10.1080/00207543.2019.1641643.
17. Huo L, Wang JY. Flexible job shop scheduling based on digital twin and improved bacterial foraging. International Journal of Simulation Modelling 2022; 21(3): 525-536, https://doi.org/10.2507/IJSIMM21-3-CO14.
18. Kacem I, Hammadi S, Borne P. Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems. IEEE Transactions on Systems Man and Cybernetics Part C (Applications and Reviews) 2002; 32(1): 1-13, https://doi.org/10.1109/tsmcc.2002.1009117.
19. Kowalski M, Izdebski M, Żak J, Gołda P, Manerowski J. Planning and management of aircraft maintenance using a genetic algorithm. Eksploatacja I Niezawodność – Maintenance and Reliability 2021; 23(1): 143-153, https://doi.org/10.17531/ein.2021.1.15.
20. Li D, Gao F. Improved slime mould algorithm based on Gompertz dynamic probability and Cauchy mutation with application in FJSP. Journal of Intelligent & Fuzzy Systems 2023; 44(6): 10397-10415, https://doi.org/10.3233/JIFS-223827.
21. Li M, Lei D. An imperialist competitive algorithm with feedback for energy-efficient flexible job shop scheduling with transportation and sequence-dependent setup times. Engineering Applications of Artificial Intelligence 2021; 103: 104307, https://doi.org/10.1016/j.engappai.2021.104307.
22. Li X, Gao L. An effective hybrid genetic algorithm and tabu search for flexible job shop scheduling problem. International Journal of Production Economics 2016; 174: 93-110, https://doi.org/10.1016/j.ijpe.2016.01.016.
23. Li Y, Liang X, Dong S. Reliability optimization design method based on multi-level surrogate model. Eksploatacja I Niezawodność – Maintenance and Reliability 2020; 22(4): 638-650, https://doi.org/10.17531/ein.2020.4.7.
24. Liang Z, Liu M, Zhong P, Zhang C. Application research of a new neighbourhood structure with adaptive genetic algorithm for job shop scheduling problem. International Journal of Production Research 2023; 61(2): 362-381, https://doi.org/10.1080/00207543.2021.2007310.
25. Ning T, Huang M, Liang X, Jin H. A novel dynamic scheduling strategy for solving flexible job-shop problems. Journal of Ambient Intelligence and Humanized Computing 2016; 7(5): 721-729, https://doi.org/10.1007/s12652-016-0370-7.
26. Oreski S, Oreski G. Genetic algorithm-based heuristic for feature selection in credit risk assessment. Expert Systems with Applications 2014; 41(4): 2052-2064, https://doi.org/10.1016/j.eswa.2013.09.004.
27. Sharifi M, Taghipour S. Optimal production and maintenance scheduling for a degrading multi-failure modes single-machine production environment. Applied Soft Computing 2021; 106: 107312, https://doi.org/10.1016/j.asoc.2021.107312.
28. Sun L, Lin L, Gen M, Li H. A hybrid cooperative coevolution algorithm for fuzzy flexible job shop scheduling. IEEE Transactions on Fuzzy Systems 2019; 27(5): 1008-1022, https://doi.org/10.1109/TFUZZ.2019.2895562.
29. Tubilla F, Gershwin SB. Dynamic scheduling in make-to-stock production systems with setup times and random breakdowns: performance

analysis and improved policies. International Journal of Production Research 2022; 60(10): 3263-3281, https://doi.org/10.1080/00207543.2021.1917013.

30. Wang Z, Pang CK, Ng TS. Robust scheduling optimization for flexible manufacturing systems with replenishment under uncertain machine failure disruptions. Control Engineering Practice 2019; 92: 104094, https://doi.org/10.1016/j.conengprac.2019.07.012.
31. Xing L, Chen Y, Yang K. Multi-population interactive coevolutionary algorithm for flexible job shop scheduling problems. Computational Optimization and Applications 2011; 48(1): 139-155, https://doi.org/10.1007/s10589-009-9244-7.
32. Xu Y, Li K, Hu J, Li K. A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Information Sciences 2014; 270: 255-287, https://doi.org/10.1016/j.ins.2014.02.122.
33. Zeng R, Wang Y. A chaotic simulated annealing and particle swarm improved artificial immune algorithm for flexible job shop scheduling problem. Eurasip Journal on Wireless Communications and Networking 2018; 2018: 101, https://doi.org/10.1186/s13638-018-1109-2.
34. Zhang G, Gao L, Shi Y. An effective genetic algorithm for the flexible job-shop scheduling problem. Expert Systems with Applications 2011; 38(4): 3563-3573, https://doi.org/10.1016/j.eswa.2010.08.145.
35. Ziaee M. A heuristic algorithm for solving flexible job shop scheduling problem. The International Journal of Advanced Manufacturing Technology 2014; 71(1-4): 519-528, https://doi.org/10.1007/s00170-013-5510-z.