Hindawi
International Journal of Antennas and Propagation
Volume 2023, Article ID 9842357, 1 page
https://doi.org/10.1155/2023/9842357

Retraction

Retracted: A FJSSP Method Based on Dynamic Multi-Objective Squirrel Search Algorithm

International Journal of Antennas and Propagation

Received 19 December 2023; Accepted 19 December 2023; Published 20 December 2023

This article has been retracted by Hindawi following an investigation undertaken by the publisher [1]. This investigation has uncovered evidence of one or more of the following indicators of systematic manipulation of the publication process:

(1) Discrepancies in scope

(2) Discrepancies in the description of the research reported

(3) Discrepancies between the availability of data and the research described

(4) Inappropriate citations

(5) Incoherent, meaningless and/or irrelevant content included in the article

(6) Manipulated or compromised peer review

The presence of these indicators undermines our confidence in the integrity of the article's content and we cannot, therefore, vouch for its reliability. Please note that this notice is intended solely to alert readers that the content of this article is unreliable. We have not investigated whether authors were aware of or involved in the systematic manipulation of the publication process.

Wiley and Hindawi regrets that the usual quality checks did not identify these issues before publication and have since put additional measures in place to safeguard research integrity.

We wish to credit our own Research Integrity and Research Publishing teams and anonymous and named external researchers and research integrity experts for contributing to this investigation.

The corresponding author, as the representative of all authors, has been given the opportunity to register their agreement or disagreement to this retraction. We have kept a record of any response received.

References

[1] Y. Wang and J. Han, "A FJSSP Method Based on Dynamic Multi-Objective Squirrel Search Algorithm," *International Journal of Antennas and Propagation*, vol. 2021, Article ID 6062689, 19 pages, 2021.

Hindawi
International Journal of Antennas and Propagation
Volume 2021, Article ID 6062689, 19 pages
https://doi.org/10.1155/2021/6062689

Research Article

A FJSSP Method Based on Dynamic Multi-Objective Squirrel Search Algorithm

Yanjiao Wang and Jieru Han

School of Electrical Engineering, Northeast Electric Power University, Jilin 132000, China

Correspondence should be addressed to Jieru Han; jieru_han@163.com

Received 1 September 2021; Accepted 30 September 2021; Published 15 October 2021

Academic Editor: Fangqing Wen

This paper comprehensively analyzes the characteristics of flexible job shop scheduling problem (FJSSP), takes the dynamic factors in the actual scheduling process such as the arrival and departure of jobs, the breakdown and recovery of machines into account at the same time, and establishes a new dynamic multi-objective mathematical model. Take the Squirrel Search Algorithm (SSA) as the core evolution strategy, and combine the multi-objective framework and the dynamic processing technology to solve the established mathematical model. Experimental results show that the mathematical model proposed in this paper can solve the flexible job shop scheduling problem effectively. Compared with other mathematical models, the mathematical model established in this paper can keep better balance between the efficiency and stability.

1. Introduction to Flexible Job Shop Scheduling Problem

The purpose of flexible job shop scheduling problem (FJSSP) is to allocate machines rationally to all the processes that arrive at the workshop to be processed, so as to achieve one or more goals concerning operation efficiency and system stability [1]. FJSSP is a complex combinatorial optimization problem. The amount of computation increases exponentially with the increase in the numbers of machines and jobs. Intelligent evolutionary algorithms are not constrained by the constraints of search space and do not rely on other auxiliary knowledge, they only need to decide the optimization objective and corresponding fitness function of the search direction, and they are suitable for solving large-scale optimization problems, such as polarization estimation, estimation, and cooperative localization [2–6]. Many scholars have used swarm intelligence and evolutionary algorithms, including the traditional intelligent optimization algorithm, such as Genetic Algorithm [7], and new intelligent evolutionary algorithms, such as Shuffled Frog Leaping Algorithm, the Swarm Spider Optimization algorithm, and so on, to solve FJSSP large-scale combinatorial optimization problems. The completion time of scheduling scheme is minimum, which is the initial optimization goal of FJSSP, and it is the most common optimization objective [8–10]; however, the actual working environment is often dynamic, i.e., new events may occur at any time, such as the arrival of new jobs, the departure of completed jobs, and the sudden breakdown of machines. Obviously, the best scheduling scheme matching the original system state is unsuitable for the new working environment. Rescheduling the jobs in the new environment is necessary. For the scheduling scheme in the new environment, if the highest efficiency is the goal, the machine with the highest efficiency for the process to be chosen may cause the load imbalance of the system, and the new scheduling scheme may have a great difference from the historical scheduling scheme, which will affect the system stability. If the system stability is the optimization objective, the gap between new and old scheduling schemes can be reduced, but the processes needed may not be performed by the most efficient machines corresponding to them, resulting in low operation efficiency. In summary, flexible job shop scheduling should consider the efficiency and stability of the system at the same time to ensure the scheduling efficiency after changing the working environment, and a mutual

exclusion relationship exists between the two. Scholars have proposed new flexible job shop scheduling methods after environmental change.

For instance, Lei and Jensen, respectively, aimed at the shortest completion time and the best robustness, and solved the task-scheduling problem with machine failure [11, 12]. Chryssolouris and Zhang used the weighted sum of completion time and delay time as the optimization target for production-scheduling scheme [13, 14], although the operation efficiency and system stability were considered in the optimization process; when solving the problem, multiple targets could be transformed into single targets by using the weighted summation method, and the reasonable weight values could not be set in accordance with the actual situation of the environment; consequently, the most suitable scheduling scheme was difficult to generate, and the universality was weak. Huang Zhiqing et al. set up an energy consumption model for the FJSSP problem and improved the genetic algorithm with the idea of particle swarm optimization and simulated annealing algorithm. The completion time and energy consumption were regarded as optimization objectives, and the efficiency and stability were optimized simultaneously. Nevertheless, the model ignored the difference between the old and new scheduling schemes before and after machine failure, and the system stability at fault time was poor [15]. Shi et al. solved the FJSSP problem by using the discrete multi-population invasive weed optimization algorithm, adaptive mutation, and domain search strategy to improve the algorithm's ability to search and use the concept of rolling window to schedule the system operation at a certain time. In essence, the static FJSSP problem is divided into several time periods, and it is not applied to the dynamic situation of the task sequence leaving at any time [16]. Chen et al., used delay and deviation as efficiency indicators and stability indicators, respectively. They used NSGA-II to optimize the two targets simultaneously. The deviation degree was evaluated by the rescheduling scheme at the time of failure. The stability of the system is improved to a certain extent before and after machine failure, but it is not applied to the dynamic environment where the operation is arriving at any time [17].

To sum up, firstly, FJSSP has natural dynamics. When establishing flexible job shop scheduling model, work efficiency is usually measured by the completion time of the task to be processed, and the stability of the system is measured by the machine load state of the workshop. Although the existing mathematical model can cope with the change in some working environments, they still cannot deal with the random arrival of task sequence, the departure of completed tasks, sudden machine failure, and the repair of failed machines at the same time. Secondly, the efficiency and stability of flexible workshop systems are mutually exclusive; therefore, in the essence of mathematics, FJSSP is a dynamic multi-objective optimization problem [18]. Compared with the method of transforming multiple targets into a single objective through weighted summation, multi-objective evolutionary algorithm can simultaneously consider efficiency and stability, and it is the main method to solve flexible job shop scheduling. Thus, the performance of multi-objective algorithm is also the key to solve DJSSP. In order to get a scheduling scheme with both efficiency and stability in dynamic environment, it is necessary to establish a mathematical model conforming to the dynamic characteristics of DFJSSP and a dynamic multi-objective evolutionary algorithm with excellent performance.

Based on the above analysis, this paper proposes a flexible shop scheduling method based on dynamic multi-objective squirrel algorithm. The main innovations are as follows: First, in consideration of the changes in an actual workshop environment, including the random arrival of task sequences, the departure of completed tasks, sudden machine failure, and the repair of failed machines, a new dynamic multi-objective flexible shop scheduling model is proposed, with the completion time, load balance (LB), and fault rescheduling deviation as the objective functions, to address working efficiency and system stability. Second, in order to effectively solve the above established mathematical model, an improved dynamic multi-objective squirrel search algorithm based on decomposition is proposed in this paper. It adopts operation and machine coding for individual coding and improves individual evolutionary strategy of SSA according to the characteristics of FJSSP. In addition, a new dynamic processing technology including the establishment and update of the external population and the initialization of the population at the time of rescheduling is proposed, which can respond quickly to changes by regenerating initial population at rescheduling time.

The remaining sections are arranged as follows: Section 2 introduces the establishment process of the mathematical model of dynamic multi-objective flexible shop scheduling problem; the third part introduces the principle of the flexible shop scheduling method based on dynamic multi-objective squirrel search algorithm; the fourth part gives the experimental results and analysis; and the fifth part summarizes the whole paper.

2. The Establishment of a Mathematical Model for Dynamic Multi-Objective Flexible Job Shop Scheduling Problem

2.1. Description of Flexible Job Shop Scheduling Problem. The flexible job shop scheduling problem can be described as: there are n jobs $J = \{J_1, J_2, \ldots, J_n\}$, each operation J_i is composed of n_i processes, and the number of multi-objective steps per operation is n_i. The processing time t_{ij} of each process is subject to certain distribution. The purpose of the FJSSP is to assign the operation machine to each process, so that the process of arriving at the workshop is processed in a certain order, and after processing is finished, the operation machine leaves the workshop. A new mathematical model for flexible job shop scheduling problem is established to realize flexible job shop dynamic scheduling. Firstly, the variables involved in flexible job shop scheduling problem are introduced as follows:

t_1: rescheduling cycle, time $t = nt_l$ (n is a positive integer), update the state of the workshop, including new task, machine idle time, etc.

t_b: the time when the machine breaks down.

t_r: the time for recovering from machinery breakdowns.

Mtbf: the average breakdown time.

Mttr: the average recovery time.

J_i: the task *i-th* that arrives at the workshop.

N_l: the number of tasks arriving at the workshop before the time of nt_l.

n_i: the number of Ji's processes.

O (i, j): the *j-th* process of task J_i.

m: the total number of machines in workshop.

M: the set of machines in workshop, $M = \{M_1, M_2, \ldots, M_m\}$.

M (i, j): a set of all machines that can handle *O (i, j)*.

F (i, j): the number of machines contained in the alternative machine set *M (i, j)* of *O (i, j)*, flexible degree.

t (i, j, k): the processing time of *O (i, j)* on machine M_k.

S_i: the beginning time of job *i*.

C_i: the completion time of job *i*. *S (i, j)*: the beginning time of process *O (i, j)*.

C (i, j): the completion time of process *O (i, j)*.

C_{k_last}: *O (i, j)* is processed by M_k, the completion time of the last procedure of M_k processing before *O (i, j)* starts.

S (i, j) *: the beginning time of O_{ij} in rescheduling processes.

C (i, j) *: the completion time of O_{ij} in rescheduling processes.

I_k: the earliest available time of machine M_k;

L_k: the total working time of machine M_k;

Task scheduling requirements for Flexible Job Shop:

(1) At the initial time, all machines are available, i.e., $t = 0$, $I_k = 0$.

(2) A process can only be processed by one machine at a time, that is, *O (i, j)* can only be processed by one machine in *M (i, j)* at a time.

(3) One machine can handle only one process at a time.

(4) There is no precedence constraint between different tasks.

(5) The process in the same task has no priority, that is, *O (i, j-1)* can only deal with *O (i, j)* after processing.

(6) All processes have no priority for the same machine, that is, the machine must deal with the current process before it can handle the next process.

(7) When the machine is not out of order, the operation system is scheduled at a fixed cycle according to the completion and arrival of the operation. When the machine breaks down, no matter whether the fault machine is executing the procedure or whether the procedure is to be executed soon, in the rescheduling plan, all the remaining procedures corresponding to the fault machine are arranged to be executed by the nonfault machine. After the fault is fixed, the repaired machine can continue to perform job scheduling.

For the periodic rescheduling time nt_l, the rescheduling time t_b and the task to be processed at the recovery time of rescheduling t_r, if the procedure *O (i, j)* is processed on machine M_k, then the beginning time *S (i, j)* and the ending time *C (i, j)* of *O (i, j)* are, respectively, shown in formulas (1) and (2). The beginning time Si and the ending time C_i of J_i are shown in formulas (3) and (4), respectively.

$$S_{ij} = \begin{cases} \max\{nt_l, I_k\}, & j = 1, \\ \max\{C(i, j-1), I_k\}, & j > 1. \end{cases} \quad (1)$$

Among them: $I_k = \max\{nt_l, t_b, t_r, C_{k\_last}\}$,

$$C(i, j) = S(i, j) + t(i, j, k), \quad (2)$$

$$S_i = \max\{S(i, 1), \ldots, S(i, n_i)\}, \quad (3)$$

$$C_i = \max\{C(i, 1), \ldots, C(i, n_i)\}. \quad (4)$$

The scheduling example in [19] is illustrated to have an enhanced understanding of the rules of flexible job shop scheduling. As shown in Table 1, "/" means that the process cannot be executed on the corresponding machine. Figure 1 is a Gantt chart corresponding to a specific scheduling plan for the data in Table 1.

As shown in Figure 1, when $t = 0$, O_{31}, O_{41}, and O_{21} are treated using M_1, M_3, and M_5, respectively. Given that O_{31}, O_{41}, and O_{21} are the first processes of J_3, J_4, and J_2, respectively, all machines are idle when $t = 0$; thus, $S_{31} = S_{41} = S_{21} = 0$. O_{11} is selected to be treated using M_1. Although O_{11} is the first step of J_1, given no priority constraint among different tasks, M_1 has started processing O_{31} when $t = 0$. O_{11} can start after O_{31} ends; therefore, $S_{11} = I_1 = C_{31} = S_{31} + t_{311} = 5$. O_{12} is processed using M_4. Given that the process in the same task has no priority, O_{12} can only start after O_{11} ends, and O_{11} ends at $t = C_{11} = S_{11} + t_{111} = 7$. However, when $t = 7$, M_4 handles O_{32}, and M_4 is idle after O_{32} is finished, i.e., $I_4 = C_{32} = S_{32} + t_{324} = 8$; consequently, $S_{12} = I_4 = 8$. O_{13} is processed using M_2. Although M_2 has been idle at $t = 5$ when O_{22} ends, O_{12} has not started yet when $t = 5$, and the process in the same task has no priority. Accordingly, $S_{13} = C_{12} = S_{12} + t_{124} = 10$.

2.2. Characteristic Analysis of FJSSP. For the scheduling problem of flexible job workshops, even for exactly the same scheduling tasks, multiple scheduling schemes often exist. Figure 2 is a Gantt chart corresponding to Table 1 and another scheduling scheme different from Figure 1.

For FJSSP, the efficiency and stability of the system are usually considered. On this basis, we compare and analyze the scheduling schemes in Figures 1 and 2 from these two perspectives. For the tasks in Table 1, the final completion time of the two scheduling schemes is 17, indicating that the efficiency of the two scheduling schemes is equivalent. However, in the scheduling scheme corresponding to Figure 1, all machines participate in the scheduling process, and the machine load is

Table 1: Six task scheduling examples.

Job number	Process number	Machine number and corresponding processing time of each process					
		1	2	3	4	5	6
1	1	2	3	4	—	—	—
1	2	—	3	—	2	4	—
1	3	1	4	5	—	—	—
2	1	3	—	5	—	2	—
2	2	4	3	—	—	6	—
2	3	—	—	4	—	7	11
3	1	5	6	—	—	—	—
3	2	—	4	—	3	5	—
3	3	—	—	13	—	9	12
4	1	9	—	7	9	—	—
4	2	—	6	—	4	—	5
4	3	1	—	3	—	—	3

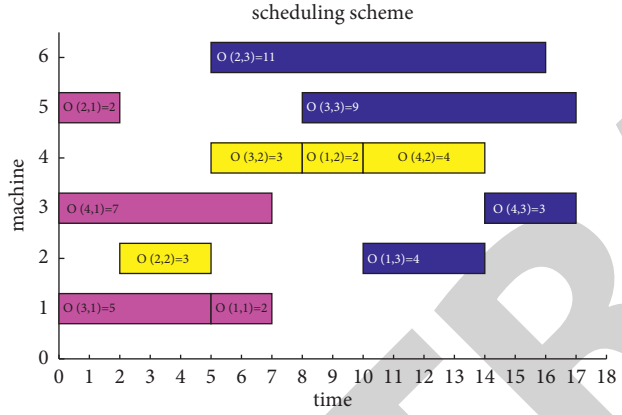


Figure 1: Scheduling plan 1.

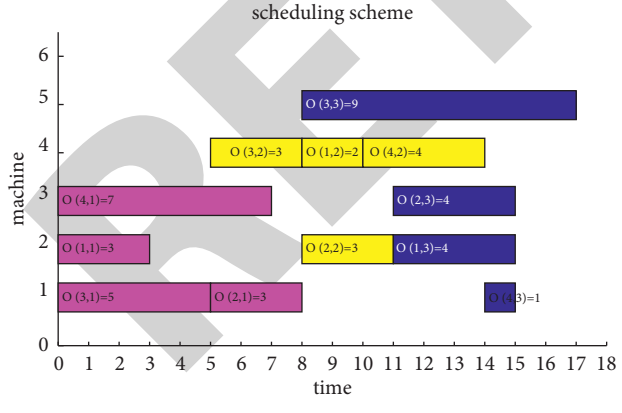


Figure 2: Scheduling plan 2.

balanced. For the scheduling scheme in Figure 2, M_6 is not involved in the scheduling process. All processes are completed using M_1–M_5, and the LB of the machine is poor. The unbalanced load of machines in the workshop may cause some machines to be idle and others to be in working state for a long time. The machines in working state for a long time are prone to failure, thus affecting the system stability. Therefore, compared with the scheduling scheme in Figure 2, the scheduling scheme in Figure 1 is better while ensuring efficiency and considering system stability.

The machine failure mentioned above is also one of the important factors that affect the system stability, and the specific analysis is as follows. When a machine in the workshop breaks down, the process in the same task has no priority, except for the process that is being implemented or has not been implemented on the failed machine. O (i, j) can only be processed after O (i, j-1) is processed, and the processes arranged on no-fault machines in the original scheduling plan will also be affected. Figure 2 is regarded as an example, i.e., if M_4 breaks down at $t = 9$. Process O (1, 2) being handled on M_4 and process O (4, 2) that has not yet been handled on M_4 need to be rescheduled. O (1, 3) and O (4, 3) cannot be executed in accordance with the original scheduling plan due to the changes in O (1, 2) and O (4, 2). Although O (2, 3) in the original scheduling scheme is unaffected by the failed machine, the machine failure changes the system state, and the scheduling of O (2, 3) will also have a new impact on the system state. Nonetheless, the workshop still maintains an efficient and stable operation at the time of failure. Except for the processes that have been started on no-fault machines and the process that has been completed on the fault machine, all the other processes need to be rescheduled. At the time of recovery, all the remaining untreated processes need to be rescheduled, except for the processes that have been completed and started. The smaller the deviation between the rescheduling scheme and the original scheduling scheme is, the smaller the system changes, and the better the stability will be. Conversely, the greater the difference between the new and old scheduling schemes is, the worse the system stability will be.

To sum up, if we want to ensure the operation efficiency, the most efficient machines in all processes can lead to some machines being idle for a long time, while some machines are in a state of work for a long time, resulting in unbalanced load. If we want to ensure the LB of machines, some jobs may not be processed by the most efficient machines, resulting in low efficiency. Some processes do not match the most efficient machines, and the efficiency of the machines may still be reduced. If the work efficiency is to be guaranteed, the new scheduling scheme may be larger than the historical scheduling scheme, but the system stability will be worse.

2.3. Dynamic Multi-Objective Mathematical Model of FJSSP. Subsection 2.2 indicates that operation efficiency and system stability are conflicted. To consider the efficiency and stability of the workshop at the same time, this paper regards the completion time, LB, and rescheduling deviation as the objective functions and establishes a multi-objective optimization mathematical model. The completion time is used to evaluate the workshop efficiency, while LB and fault rescheduling deviation are used to measure the system stability. The details are as follows:

2.3.1. Work Efficiency. A short completion time means high work efficiency. To ensure the workshop work efficiency, this paper regards the shortest completion time as the

optimization objective, and the objective function is shown in Formula (5).

$$\min CT = \max\{C_i, i = 1, 2, \ldots, N_l\}. \tag{5}$$

2.3.2. System Stability. This study measures the system stability from two aspects: the machine LB and rescheduling deviation. LB is measured using the variance of machine utilization shown in Formula (6); the smaller the variance is, the more balanced the machine load and the better the system stability will be. When machines break down or recover, the average deviation (AD) between the new and old scheduling schemes shown in Formula (8) indicates the rescheduling deviation. Small AD indicates a small difference between the rescheduling and original scheduling schemes, and improved stability.

$$\min LB = \frac{1}{m}\sum_{k=1}^{m}\left[\frac{L_k}{\max\{L_1, \ldots, L_m\}} - \frac{1}{m}\sum_{k=1}^{m}\frac{L_k}{\max\{L_1, \ldots, L_m\}}\right]^2. \tag{6}$$

In (6),

$$L_k = \sum_{i=1}^{N_l}\sum_{j=1}^{n_i} x(i, j, k) * t(i, j, k),$$
$$x(i, j, k) = \begin{cases} 1, & O(i, j) \text{is handle using } M_k, \\ 0, & \text{other.} \end{cases} \tag{7}$$

The smaller the LB is, the more balanced the system will be.

$$\min AD = \frac{\sum_{i=1}^{N_l}\sum_{j=1}^{n_i}\left(\left|S(i, j) - S^*(i, j)\right| + \left|C(i, j) - C^*(i, j)\right|\right)}{n_{\text{res}}}. \tag{8}$$

In (8), n_{res} is the total number of operations to be rescheduled; for the operations that have started or been completed on the working machines, $S(i, j) = S^*(i, j)$ and $C(i, j) = C^*(i, j)$.

In summary, to consider the efficiency and stability of the system, a dynamic multi-objective mathematical model of FJSSP is established:

$$\begin{cases} \min CT = \max\{C_i, i = 1, 2, \ldots, N_l\}, \\ \min LB = \dfrac{1}{m}\displaystyle\sum_{k=1}^{m}\left[\frac{L_k}{\max\{L_1, \ldots, L_m\}} - \frac{1}{m}\sum_{m}^{1}\frac{L_k}{\max\{L_1, \ldots, L_m\}}\right]^2, \\ \min AD = \dfrac{\left(\sum_{i=1}^{N_l}\sum_{j=1}^{n_i}\left(\left|S(i, j) - S^*(i, j)\right| + \left|C(i, j) - C^*(i, j)\right|\right)\right)}{n_{\text{res}}}. \end{cases} \tag{9}$$

In the mathematical model above, the first objective function guarantees the shortest completion time of all tasks arriving at time nt_1; the second objective function guarantees that all machines will be loaded as evenly as possible until nt_1; the third objective function ensures that when machines break down or get back to work, the deviation between the rescheduling scheme and the original scheduling scheme is the smallest. When no machine breaks down or a machine is not recovered, only the objective functions one and two are considered, and the mathematical model is considered a two-objective optimization problem. If a machine breaks down or the faulty machine is recovered, the three objectives are all needed to be optimized, and the mathematical model is regarded as a three-objective optimization problem.

Compared with existing mathematical models of FJSSP, the mathematical model established in this paper considers the dynamic characteristics of the departure of completed tasks, the arrival of new tasks, the failure of workshop machines, and the repair of machine faults. The mathematical model adjusts the optimization objective automatically in accordance with the machine state, which will shorten the working time and balance the load of the machine when no machine breaks down or a machine is not recovered. Meanwhile, the established mathematical model minimizes the deviation of the new and old scheduling schemes when a machine breaks down or the faulty machine is recovered, which gives consideration to the working efficiency and system stability in real time.

3. Dynamic Multi-Objective SSA For Solving FJSSP

When solving a dynamic multi-objective optimization problem, the dynamic environment is divided into several transient environments; a dynamic processing technology is used to deal with environmental changes, and a multi-objective algorithm is used to solve static multi-objective optimization problems in a transient environment. Squirrel Search Algorithm (SSA) is a new swarm intelligence evolutionary algorithm [20]. Compared with traditional evolutionary algorithms, SSA has higher convergence speed and accuracy. Wang et al. propose an Improved Squirrel Search Algorithm for global function optimization (ISSA), which improves the convergence of SSA. Besides, Zhang et al.

proposed MOEA/D, which can solve the multi-objective optimization problem in a transient environment and has high solving speed and efficiency [21]. Therefore, to expand the application of SSA in the engineering field, this paper regards ISSA as the core evolutionary strategy, considers the multi-objective framework of MOEA/D, and integrates a dynamic processing technology to solve FJSSP. Figure 3 is the flowchart of the dynamic multi-objective SSA for solving FJSSP.

Figure 3 illustrates that at the end of each transient environment, the final scheduling scheme under the current environment in the nondominated solution set should be selected, and the scheduling under the new environment is based on the selected scheduling scheme. In this study, the TOPSIS method [22] is used to determine the final scheduling scheme for each transient environment. In addition, considering that FJSSP is a discrete optimization problem, this study adopts the individual coding method [23]. In accordance with the characteristics of FJSSP, the dynamic processing technology and evolution strategy for FJSSP are designed as follows.

3.1. Individual Coding Method. The individual coding method includes two parts: operation and machine coding. The operation coding is determined using the job number to be processed at the rescheduling time. Operation *O (i, j)* is represented by its task number J_i. The first occurrence of J_i corresponds to operation *O (i, 1)*, the second occurrence of J_i corresponds to operation *O (i, 2)*, and so on. The *j-th* occurrence of J_i corresponds to operation *O (i, j)*; therefore, the frequency of occurrence of J_i is the same as the total operation number of J_i. Each operation code corresponds to one machine code. A machine is selected from the candidate machine set of *O (i, j)* randomly, and the number of the selected machine is the machine code corresponding to *O (i, j)*.

The task data in Table 1 are regarded as an example. The complete code corresponding to the scheduling scheme shown in Figure 1 is 342133221144/135145264243, i.e., $O_{31} \longrightarrow M_1$, $O_{41} \longrightarrow M_3$, $O_{21} \longrightarrow M_5$, $O_{11} \longrightarrow M_1$, $O_{32} \longrightarrow M_4$, $O_{33} \longrightarrow M_5$, $O_{22} \longrightarrow M_2$, $O_{23} \longrightarrow M_6$, $O_{12} \longrightarrow M_4$, $O_{13} \longrightarrow M_2$, $O_{42} \longrightarrow M_4$, $O_{43} \longrightarrow M_3$.

3.2. Dynamic Processing Technology

3.2.1. Establishment and Renewal of External Population. The solution principle of MOEA/D indicates that when offspring individual y' cannot replace any individual in the neighbourhood, y' is abandoned directly. Thus, this study establishes an external population *Ey* for each individual to make the abandoned y' participate in the evolution. The specific way is as follows. An external population with the same size is set as the neighbourhood *T*. num_alt is used to record the number of individuals in the neighbourhood replaced with y'. y', whose num_alt is equal to 0, enters the external population.

When the scale of *Ey* is smaller than the upper limit *T*, if num_alt = 0, the corresponding y' enters *Ey*; if $0 <$ num_alt $\leq T/2$, *Ey* remains unchanged; if num_alt $> T/2$, *Ey* is emptied because the current subproblem and its neighbour subproblems are far from the optimal solution.

When the scale of *Ey* is up to the upper limit *T*, if num_alt = *0*, the population has evolved to a certain extent, and the current individual is close to its neighbour individual. To maintain the population diversity, the individual with the smallest Euclidean distance to the current individual in *Ey* is deleted, and y' enters *Ey*.

3.2.2. Generation of Initial Population at Rescheduling Time. Two cases needing to reschedule tasks are considered in this paper. The first one is related to workshop working time, which is called periodic rescheduling. The second one is related to machine state, including fault and fault recovery rescheduling. In consideration of the two situations above, two different dynamic processing technologies are designed, and the details are as follows.

At the time of periodic rescheduling nt_l, the new tasks are recorded, the machine state is updated, and the initial population under the new period is generated in accordance with the coding method in Subsection 3.1.

If a machine breaks down or operates again, i.e., $t = t_b$ or $t = t_r$, the new population of t_b or t_r is generated in accordance with the implementation of the original scheduling scheme. The details are as follows. If $t = t_b$, in the current scheduling scheme, operations to be rescheduled contain all the unfinished operations arranged on the broken machine and all the operations that have not started until t_b. If $t = t_r$, in the current scheduling scheme, operations to be rescheduled contain all the operations that have not started until t_r. The set of all operations to be rescheduled is named O_{re}. When $t = t_b$ or $t = t_r$, for each individual in the original population, if $Ey = \varnothing$, all operations in O_{re} should be reinitialized as the new individual in the new environment in accordance with the coding method above; if $Ey \neq \varnothing$, for each scheduling scheme in *Ey*, the set of operations that need to be rescheduled is recorded and named O_{ey}; if memory individuals with exactly the same O_{ey} and O_{re} exist, one is selected randomly, the scheduling order of all operations is kept in O_{ey}, and the corresponding machine code is mutated and regarded as the initial individual in the new environment. Given the high similarity between the scheduling scheme in memory population and the currently executing scheduling scheme, the new individuals generated using memory individuals are more suitable for the current working state, which is conducive to the system stability.

3.3. Core Evolutionary Strategy. In the scheduling process of each transient environment, the core evolutionary strategy SSA is used to update the population. An individual contains operation and machine codes, but the machine code is determined by the operation code. Therefore, the updating strategy of the population is aimed only at the operation code, rather than the machine code. The specific updating process is described below.

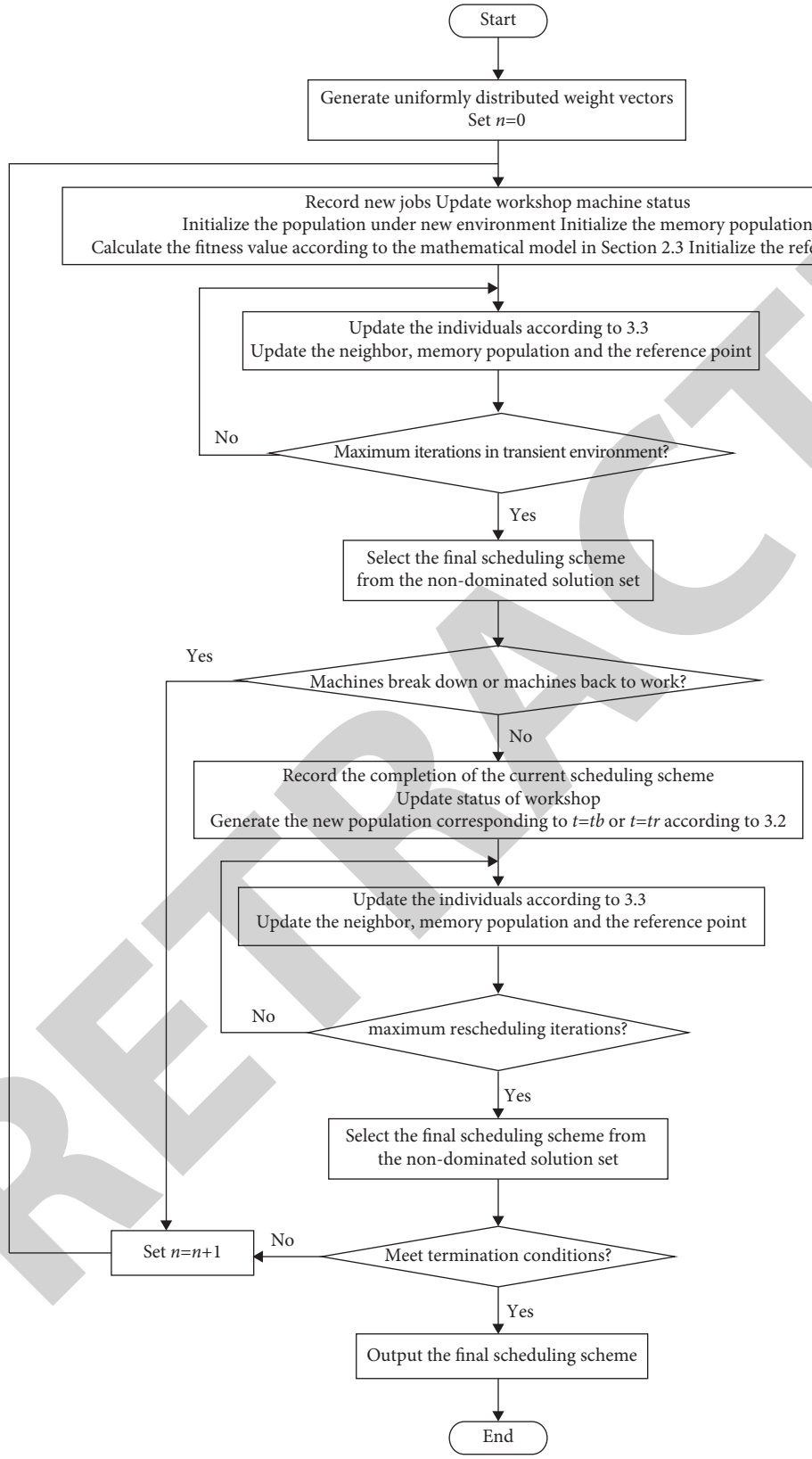


Figure 3: Flowchart for solving FJSSP.

3.3.1. Related Variables

F_d: Learning target in accordance with the standard SSA; the details have been introduced in Reference [20].

n_same: When the operation codes of FS_i and F_d are matched one by one, n_same is the total number of the same operation codes.

n_j_l: Total number of new jobs at t_l.

Fr_1, Fr_2: If $Ey = \varnothing$, Fr_1 and Fr_2 are two individuals different from F_d and FS_i randomly selected in their neighbor; if $Ey \neq \varnothing$, Fr_1 is any individual in its neighborhood different from FS_i, and Fr_2 is randomly selected in Ey.

S_c^g and $S_{\min}$ are calculated according to Reference [20], and the winter search strategy is selected when $S_c^g > S_{\min}$.

(1) Winter search strategy

① No predators exist when $r > P_{dp}$.
The jump search method is selected when $n\_same > n\_j_l/2$, and the progressive search method is decided when $n\_same \leq n\_j_l/2$.
The jump search method: FS_i and F_d have high similarity when $n\_same > n\_j_l/2$, and the individual is updated using Formula (10).

$$FS_i^{g+1} = FS_i^g + \left[d_g \times G_c \times \left(F_d^t - FS_i^t\right)\right]. \quad (10)$$

In the formula above, "[.]" means to carry out rounding operation, which ensures that the individual code of each dimension is a positive integer. d_g and G_c have been introduced in Reference [20].
The progressive search method: FS_i and F_d have low similarity when $n\_same \leq n\_j_l/2$. To maintain population diversity, r_{ds} dimensions are selected randomly in accordance with Formula (11) and then rearranged also randomly.

$$r_{ds} = \left[r \times \left(n\_j_l - n\_same\right)\right]. \quad (11)$$

In the formula above, r is a random number between 0 and 1; the lower the similarity between FS_i and F_d is, the more dimensions can be rearranged, and the search space can be developed more sufficiently.

② A predator occurs when $r \leq P_{dp}$.
FS_i is considered dead, and the offspring is generated via F_{r1} and F_{r2} in accordance with the POX introduced in Reference [24].

(2) Summer search strategy

① No predators exist when $r > P_{dp}$.
The jump search method is selected when $n\_same > n\_j_l/2$, and the progressive search method is chosen when $n\_same \leq n\_j_l/2$.
The jump search method: FS_i and F_d have high similarity when $n\_same > n\_j_l/2$, and the individual is updated using Formula (12).

$$FS_i^{t+1} = F_d^t + \left[d_g \times G_c \times \left(FS_i^t - F_d^t\right)\right]. \quad (12)$$

Formula (12) regards F_d as the basic vector, improves the convergence speed at the beginning of environmental change, and deeply develops the search space after the population evolves to a certain extent.
The progressive search method is the same as that in the winter search method. If $n\_same \leq n\_j_l/2$, r_{ds} dimensions are selected randomly in accordance with Formula (10) and then rearranged also randomly.

② A predator occurs when $r \leq P_{dp}$.

FS_i is considered dead, and the offspring is generated using F_{r1} and F_{r2} in accordance with the POX introduced in Reference [24].

FJSSP requires that the value of each dimension is the job number corresponding to the operation, and the total occurrence time of each job number should be the same as its total operations. Therefore, the operation code of the offspring obtained using the updating strategy above should be modified. The task in Table 1 is regarded as an example to introduce the modification process, and the details are as follows.

The redundant job numbers are determined and recorded in vector "more." If number i of job J_i occurs $n_i + c$ times, then i occurs in "more" c times. Figure 4 is considered an example. For the problem in Table 1, the operation code of the offspring is 134312214131, more = {1, 1}.

The missing job numbers are identified and recorded in vector "*less.*" If number i of job J_i occurs n_i-c times, then i occurs in "less" c times. Figure 4 is regarded as an example. For the problem in Table 1, the operation code of the offspring is 134312214131, less = {2, 4}.

The redundant job numbers occurring in "more" in the offspring are selected randomly to replace the missing job numbers in "less." All elements in "less" are arranged randomly, and the modification is completed. Figure 4 is regarded as an example, the 5-th dimension of the offspring operation code changes from 1 to 4, and the 10th dimension changes from 1 to 2.

An operation machine is selected randomly in the alternative machine set after the operation code of each offspring is determined, and the operation code constitutes a complete offspring scheduling scheme.

4. Experimental Simulation and Result Analysis

To verify the effectiveness of the dynamic multi-objective SSA for solving the FJSSP mathematical model established in this paper, the FJSSP model and the dynamic multi-objective SSA are applied to solve an actual FJSSP. All the tests are implemented on a computer CPU: Intel (R) Core (TM) i5-7200U, 4 G memory, and 2.70 GHz main frequency. The program is implemented in MATLAB R2016a. The experiment includes two parts: effectiveness validation and advanced verification.

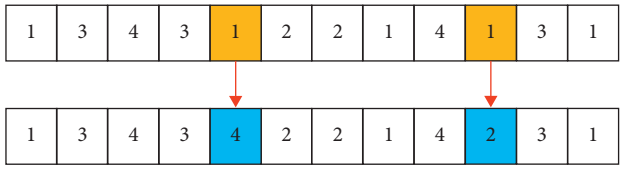


Figure 4: Modification of the operation code.

4.1. Effectiveness Validation. To investigate the FJSSP mathematical model established in this paper and the effect of the designed scheduling method, the following FJSSPs are tested: the number of machines is $m = 6$; the rescheduling cycle is $t_l = 1$; the time intervals between two adjacent tasks arrive at an exponential distribution of mean 0.5; the number of jobs per task n_j obeys the uniform distribution between $[m/2, m]$; the flexibility of each process is subject to a uniform distribution between $[m/2, m]$; the processing time of each process obeys the exponential distribution with a mean value of 1; *MTBF* obeys the exponential distribution of mean *mtb*, and *mtb* obeys the uniform distribution between [20, 40]; *MTTR* obeys the exponential distribution of mean *MTR*, and *MTR* obeys the uniform distribution of [5, 10] and randomly selects a fault machine. The task to be processed at the initial time is shown in Table 2. The algorithm parameters are set as follows: SSA's $N_{fs} = 2$; population size $N = 300$; the upper limit of neighbourhood size and positive memory population size is 20; the maximum iteration number of each transient environment is $g_{\max} = 100$; POX cross mode cross probability $p_c = 1$; mutation probability $p_m = 1$; when $N_{l=}100$, the total number of scheduling tasks is terminated.

Table 3 is a part of the nondominated solutions obtained from the fault free time $t = 5$, the first failure time $t = 15.7633$, and the first fault repair time $t = 23.7424$ or the rescheduling task optimization. Figures 5–7 are, respectively, the nondominated graphs obtained from the task to be processed or the rescheduling task optimization of t = 52.1802 at the time of failure free t = 30, the second failure time t = 43.6123, and the second fault repair time.

From the above chart, the mathematical model proposed in this paper is a two-objective optimization problem when all the machines in the workshop are free of failure. The nondominated solution is distributed in the same plane. When the fault occurs or restores, the mathematical model is a three-objective optimization problem, and the nondominated solution is dispersed in a three-dimensional space.

Figures 8 and 9 are the Gantt charts of the shop scheduling plan during $t = 10$–30 and $t = -55$, respectively. J_i is the task number. Purple, yellow, blue, orange, cyan, and gray represent the 1–6 processes of each task, respectively. Red represents the rescheduling process after a machine fails, and green represents the rescheduling process after the machine fails to recover. When machine M_2 breaks down at $t = 15.7633$–23.7424, the fault machine does not perform any operation. When machine M breaks down at $t = 43.6123$–52.1802, the fault machine does not perform any operation. After the machine breaks down, the fault machine is in working state again.

In order to further verify the FJSSP mathematical model established in this paper and the effect of the designed scheduling method, we carry out the following experiments. Set the number of machines to 8, and the rescheduling period $t_l = 15$; the time interval between two adjacent tasks follows an exponential distribution with a mean of 0.625. The parameter settings and algorithm settings of other scheduling environments are consistent with those mentioned above. The tasks to be processed at the initial time are shown in Table 4. Figures 10 and 11 are the Gantt charts of the shop scheduling plan during $t = 15$–25 and $t = 30$–40, respectively.

It can be seen from Figures 10 and 11, when machine M_1 breaks down at $t = 15.5810$–$t = 17.7670$, the fault machine does not perform any operation. When machine M_2 breaks down at $t = 33.9251$–$t = 39.2939$, the fault machine does not perform any operation. After the machine breaks down, the fault machine is in working state again.

To sum up, the FJSSP mathematical model and the scheduling method established in this paper can effectively cope with the changes in an actual workshop environment, including the task of reaching or leaving at any time and the occurrence of faults. This condition can effectively solve FJSSP.

4.2. Advanced Verification. The dynamic multi-objective SSA is used to optimize the mathematical models established in this paper and Reference [23] to solve FJSSP by satisfying the following conditions: the number of machines is $m = 6$; rescheduling cycle $t_l = 30$; the time intervals between two adjacent tasks arrive at an exponential distribution of mean 3; the number of jobs per task n_j obeys the uniform distribution between $[m/2, m]$; the degree of flexibility of each process obeys the uniform distribution between $[m/2, m]$; the processing time of each process obeys the exponential distribution of mean 5; *MTBF* obeys the exponential distribution of mean *mtb*, and *mtb* obeys the uniform distribution of [100, 200]; *MTTR* obeys the exponential distribution of mean *mtr*, and *mtr* obeys the uniform distribution of [30, 60] and randomly selects a fault machine at the time of failure. The rest of the parameter settings are consistent with the previous ones. The task to be processed at the initial time is shown in Table 5.

For a dynamic scheduling environment, the scheduling scheme is different, and the workshop state at the end of each transient environment is diverse. That is, the initial state of the workshop at the beginning of the next new environment is different. Comparing the optimization capability of different mathematical models for FJSSP requires the same initial workshop state and the same task to be optimized. The following parts are tested in this study.

(1) The mathematical model in Reference [23] is used to optimize the dynamic environment continuously. The workshop state at the beginning of each new scheduling cycle is recorded. Several different initial states and transient scheduling cycles are obtained. The new task to be processed in each scheduling cycle is recorded, and the mathematical model built in this

Table 2: Tasks to be processed at $t = 0$.

Job number	Process number	Machine number and corresponding processing time of each process					
		1	2	3	4	5	6
1	1	1.5278	0.5901	1.1614	0.7246	0.4811	0.4691
1	2	0.3956	0.4539	1.0810	1.0675	0.6425	0.5351
1	3	0.5129	0.6446	0.3990	—	0.3562	—
2	1	0.3668	0.1866	3.7728	0.5068	1.4320	1.0902
2	2	1.2446	0.6315	—	0.2521	—	0.2403
2	3	0.4178	0.6454	—	0.5130	0.4797	0.1741
3	1	1.0332	0.4689	1.0538	0.5362	0.5307	1.0737
3	2	1.4766	0.6708	0.6258	0.5990	3.4804	—
3	3	—	0.5051	0.5074	0.4874	0.4436	1.9361
3	4	0.5011	3.0418	0.3736	—	0.5299	1.4914
3	5	0.5034	1.0831	0.4777	0.3409	—	—
4	1	—	—	0.6324	2.4931	0.6410	0.4427
4	2	0.4982	0.5326	0.5844	0.0977	1.7782	1.0170
4	3	1.0407	0.3432	2.2131	2.3234	—	0.3157
4	4	0.4482	0.5209	2.5250	6.4491	—	0.6933
5	1	0.4508	—	0.4902	0.7867	0.4561	—
5	2	—	—	0.3664	—	0.6246	0.5435
5	3	—	0.4329	0.3010	—	0.7159	—
6	1	0.4204	0.5577	1.0348	2.8571	—	0.7716
6	2	1.7587	1.5530	0.6008	—	0.7694	—
6	3	0.5188	0.3955	—	0.6630	2.4961	1.07434
6	4	1.5050	1.6941	0.6725	—	0.7614	0.5686
7	1	2.5187	3.7462	0.2872	0.4990	—	0.5650
7	2	3.7646	—	1.7037	—	0.7952	3.1828
7	3	1.0688	0.3720	0.6191	—	—	0.2297
8	1	0.6037	0.4592	—	1.0678	0.7454	0.4188
8	2	0.4884	0.4996	0.4361	—	1.8487	0.4254
8	3	0.4996	0.4550	—	—	0.1078	3.6344
8	4	0.3247	0.5305	—	0.6690	—	0.3700
9	1	0.5065	0.5234	0.7427	—	2.3589	0.3925
9	2	0.4606	0.5201	0.2577	0.6951	0.5649	1.4906
9	3	1.1232	—	0.5104	1.3293	1.0829	0.3600
9	4	—	—	—	1.8475	2.4910	0.7087
10	1	1.0394	—	0.5291	0.4944	0.3455	—
10	2	0.5875	—	0.4925	0.5101	—	0.7072
10	3	—	—	0.3822	3.9409	0.3874	0.5002
10	4	0.6164	0.2521	0.6785	—	—	—
10	5	2.9608	0.4550	0.6386	0.4971	2.3437	2.0466
10	6	0.4888	—	0.4568	2.4233	1.7440	0.6625

Table 3: Nondominated solutions retained at different times.

$t = 5$			$t = 15.7633$			$t = 24.7424$		
CT	LB	AD	CT	LB	AD	CT	LB	AD
9.7185	0.0547	0	20.3301	0.0495	7.2194	27.1586	0.0539	1.5871
10.5680	0.0529	0	20.5133	0.0401	7.2194	27.3802	0.0539	0.9745
10.7969	0.0511	0	21.5258	0.0367	7.2194	27.3130	0.0537	1.4474
11.2463	0.0453	0	22.4842	0.0541	4.8224	27.3263	0.0548	0.9963
11.4612	0.0435	0	25.0291	0.0503	4.8224	27.6919	0.0537	0.9963
12.1650	0.0398	0	26.6129	0.0411	4.7628	27.7270	0.0528	1.1974
13.2902	0.0307	0	27.8853	0.0270	5.7375	27.4832	0.0559	0.9328
14.8153	0.0243	0	29.8500	0.0232	3.7171	27.5886	0.0531	1.0079

paper is used to optimize the corresponding workshop initial state. The completion time and maximum machine load of two scheduling indexes in Reference [23] are compared. Figure 12 is a comparison chart of the maximum scheduling time obtained from the optimization of the two models. Figure 13 is a comparison diagram of the maximum machine load obtained from the two models.

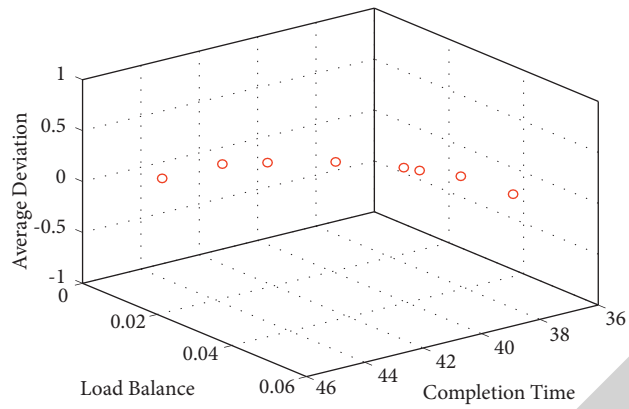


Figure 5: Nondominated graph for the optimization of tasks to be processed at $t = 30$.

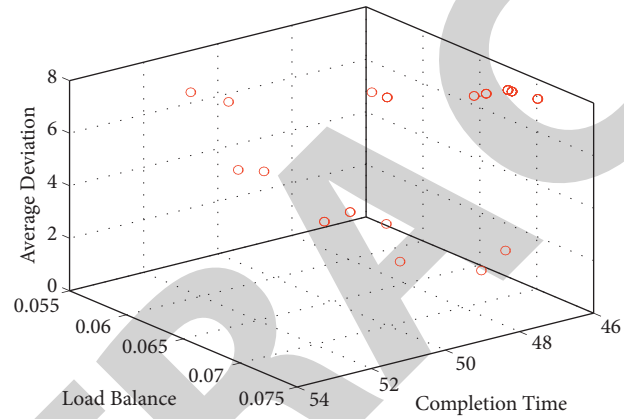


Figure 6: Nondominated graph optimization for rescheduling tasks at $t = 43.6123$.

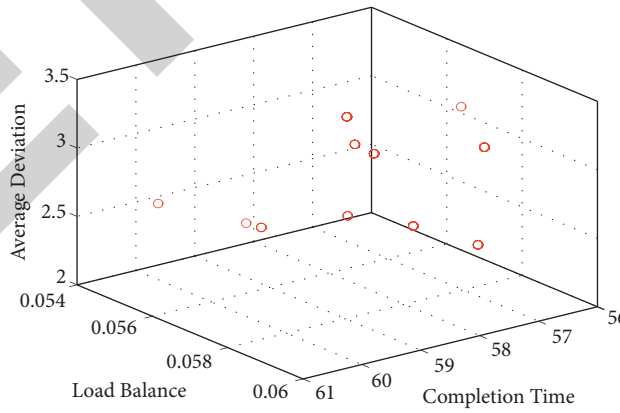


Figure 7: Nondominated graph optimization for rescheduling tasks at $t = 52.1802$.

(2) Figure 12 shows that when the mathematical model in Reference [23] is used to optimize FJSSP continuously, the completion time increases with the increase in the number of tasks to be processed. At the initial time, the mathematical model in Reference [23] achieves shorter completion time at the fourth scheduling cycle, but a minimal difference exists between the two mathematical models at the ninth scheduling cycle. In the rest of the scheduling cycles, the mathematical models in this paper achieve shorter completion time. Therefore, compared with the mathematical model in Reference [23], the mathematical model in this paper is more efficient in solving FJSSP. In Figure 13, the maximum machine

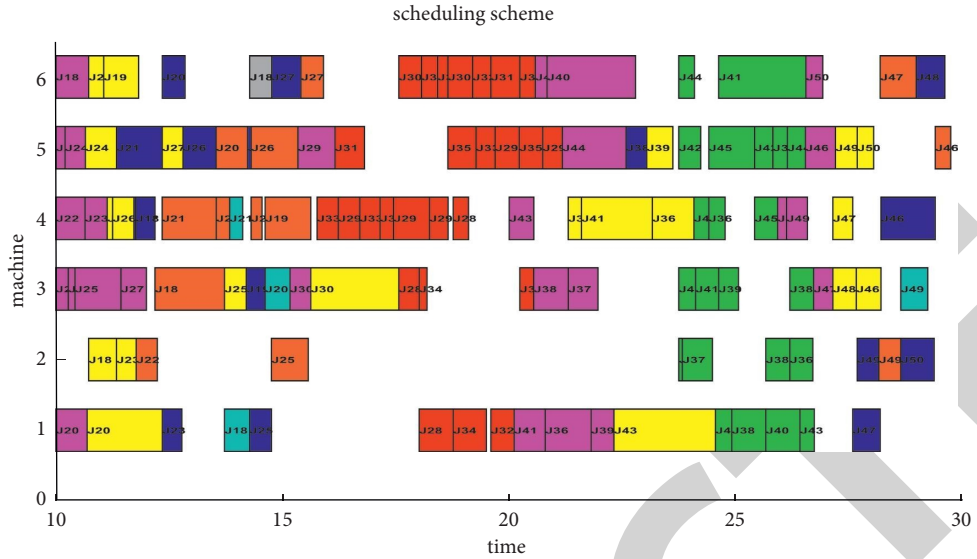


Figure 8: Gantt chart of $t = 10–30$ scheduling scheme.

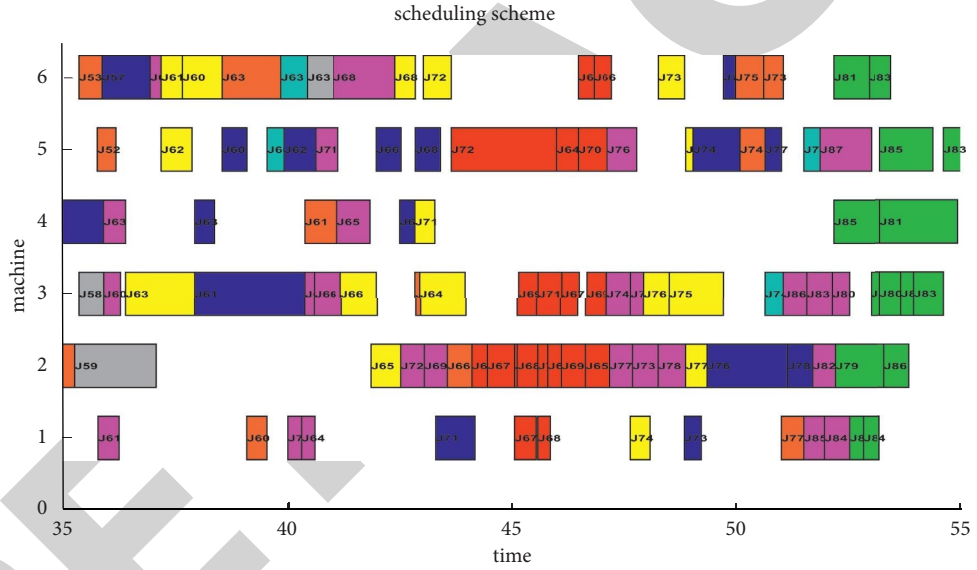


Figure 9: Gantt chart of $t = 35–55$ scheduling scheme.

Table 4: Tasks to be processed at $t = 0$.

Job number	Process number	Machine number and corresponding processing time of each process							
		1	2	3	4	5	6	7	8
1	1	—	0.6382	1.2681	1.0710	—	1.3042	0.8963	—
1	2	0.5019	0.8400	1.3880	1.0160	0.3989	0.8541	0.4708	0.3358
1	3	1.6475	—	0.8298	—	1.2152	0.2717	1.5794	1.5971
1	4	2.8928	1.4613	—	1.1014	0.9968	2.1039	—	—
1	5	1.1864	1.3696	1.2559	0.1734	—	—	—	—
1	6	1.2120	0.6545	0.6413	—	1.2581	1.0654	—	0.7189
2	1	0.6815	0.8164	0.9751	—	1.0529	2.3129	—	0.2768
2	2	0.4422	—	0.4240	—	1.5143	1.5095	2.0199	0.8215
2	3	2.7416	2.6740	—	1.7388	1.3254	0.0500	2.1685	0.3425
2	4	—	0.9891	2.0880	2.2006	0.4174	0.5254	1.9515	0.4033
2	5	2.3676	0.3323	0.7549	1.8217	0.1319	0.5349	1.8295	0.3132
3	1	2.0371	0.7917	1.5249	0.5102	1.7024	1.4674	0.5818	0.5744
3	2	2.1183	—	0.5471	0.9859	0.5768	1.0352	0.6740	1.9032
3	3	0.9638	0.6347	1.1081	0.9293	1.1082	1.4623	0.8659	0.4380

Table 4: Continued.

Job number	Process number	Machine number and corresponding processing time of each process							
		1	2	3	4	5	6	7	8
3	4	—	—	1.4176	0.2305	—	—	—	2.2456
4	1	2.0491	1.6558	—	1.0707	1.5561	0.3166	1.3802	0.7998
4	2	0.3704	0.7575	1.2038	—	—	0.5326	—	—
4	3	—	1.2657	—	—	0.8511	0.8744	2.9220	1.4142
4	4	1.3512	1.4050	0.4743	2.6791	2.8502	1.1182	2.9756	0.3806
4	5	0.9679	—	1.3229	1.2626	1.8965	0.4182	1.3583	0.5312
4	6	—	0.3225	1.9250	—	1.6030	0.9702	2.5834	—
5	1	0.4691	2.1852	0.7140	0.9937	1.2777	2.7375	3.0324	0.4996
5	2	2.9772	—	1.8705	—	—	—	—	1.1175
5	3	—	1.8562	0.1373	1.0024	0.7281	—	—	0.8232
5	4	1.3402	0.6634	0.0999	0.6004	0.2650	—	—	—
5	5	0.9894	1.9170	2.1850	2.9171	—	0.3045	0.5929	0.3364
5	6	0.6517	0.7840	—	0.2839	0.9946	0.2948	—	0.4172
6	1	1.1747	—	—	—	—	0.2649	0.7974	1.7007
6	2	1.0199	0.6806	0.4107	1.6766	—	0.7231	1.2838	2.3242
6	3	—	—	—	1.6511	0.5025	—	—	1.3843
6	4	0.2826	—	—	1.8508	—	—	0.5031	—
7	1	0.8562	0.4408	1.2425	—	—	0.9039	1.7611	2.9910
7	2	0.5652	0.8330	1.2936	0.6021	—	0.9480	1.9891	2.9164
7	3	—	0.5842	1.4921	—	0.5617	—	0.6148	—
7	4	—	—	0.7785	—	0.6716	1.6041	1.9949	0.0982
7	5	—	0.5235	1.4898	—	1.3141	0.0204	—	—
8	1	—	0.5735	—	—	0.1725	—	1.3313	0.4930
8	2	0.3275	—	—	—	0.9262	—	—	0.6194
8	3	—	2.2066	0.8222	0.9665	0.6131	0.2283	1.1873	1.7934
8	4	0.5493	0.6385	0.5536	—	0.6374	0.5601	1.1408	—
8	5	0.9098	—	—	—	1.4946	0.6431	—	0.7129
8	6	1.6085	3.3016	3.1581	0.3536	1.1893	0.4541	0.3692	—
9	1	1.5629	0.3463	1.0989	1.6513	—	—	—	—
9	2	0.2841	0.7069	2.4057	1.3018	1.1407	0.4920	1.9512	2.9612
9	3	0.0736	—	1.7815	1.3833	0.1368	1.5720	2.1434	0.4936
9	4	—	1.6847	3.0446	—	—	3.0226	0.5515	—
9	5	0.4957	1.7989	—	1.2870	0.4484	2.8027	—	0.0536
10	1	0.4631	2.9016	—	0.8857	0.1663	—	—	—
10	2	0.1600	—	1.0278	—	0.4837	—	—	—
10	3	—	2.0967	1.3427	—	—	0.3534	0.1632	1.3010
10	4	0.5864	1.3044	2.0412	0.6325	1.8661	0.4146	—	—
10	5	1.8594	0.5863	0.6037	0.2186	0.6127	0.1225	3.1598	—
10	6	2.2898	1.4587	—	0.1752	0.4326	0.3804	1.9046	0.9512

load increases with the increase in the number of tasks to be processed. The first scheduling cycle and the seventh scheduling period in the literature yield a smaller maximum machine load. The optimization results of the two mathematical models at the fifth scheduling cycle are similar to each other. In the rest of the scheduling cycles, the mathematical models proposed in this paper achieve a smaller maximum machine load. Hence, compared with the mathematical model in Reference [23], the mathematical model in this paper has better stability when solving FJSSP.

(3) The mathematical model established in this paper is used to optimize the above dynamic environment continuously. The workshop state at the beginning of each new scheduling cycle is recorded. Several different initial states and transient scheduling cycles are obtained. The new task to be processed in each scheduling cycle is recorded, and the mathematical model in Reference [23] is optimized in the initial state of the corresponding workshop. The completion time, LB, and deviation degree of the three scheduling indexes optimized in this paper are compared. Figure 14 shows a comparison of the maximum scheduling time obtained from the optimization of the two models, and Figure 15 depicts a comparison of the LB obtained from the optimization of the two models. Figure 14 shows the two models at the time of failure or recovery (M_2 breaks down at $t = 85.7705$, 108.0616; M_5 breaks down at $t = 137.4366$ and recovers at $t = 196.8504$; M_6 breaks down at $t = 253.2210$ and recovers at $t = 261.4715$). Figure 16 is a Gantt chart of the final scheduling scheme obtained by optimizing the mathematical model established in this paper.

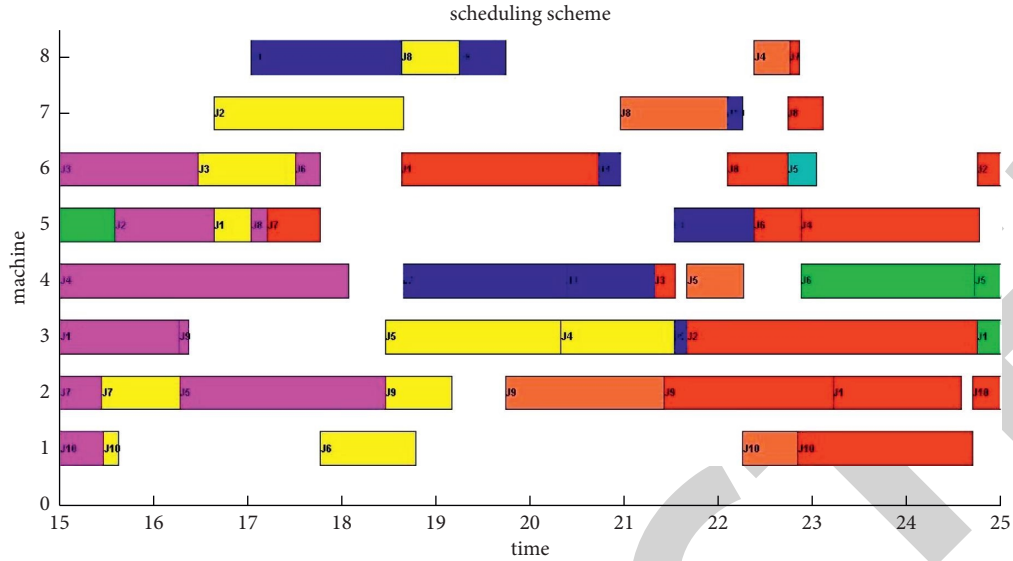


Figure 10: Gantt chart of $t = 15 \sim t = 25$ scheduling scheme.

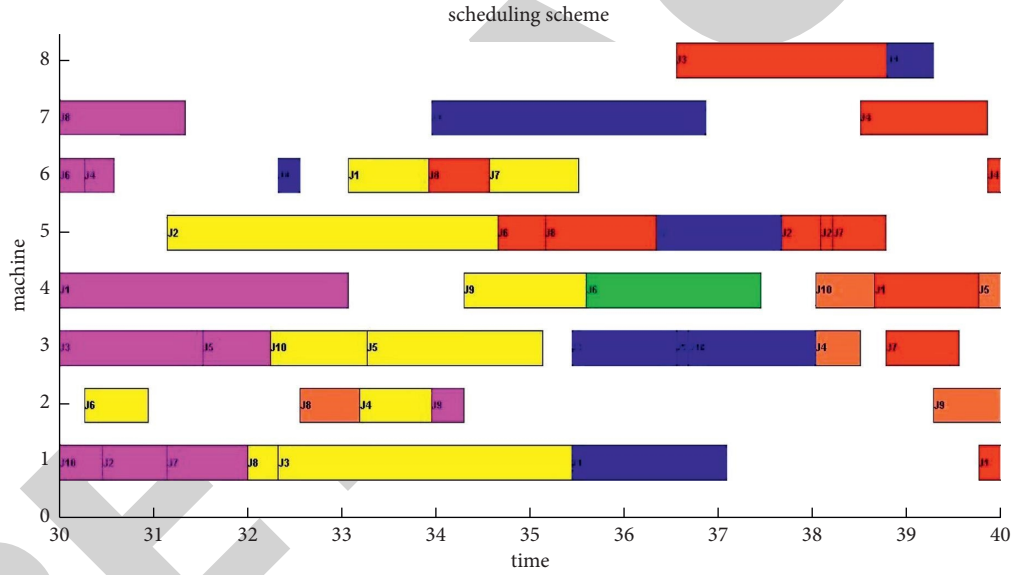


Figure 11: Gantt chart of $t = 30 \sim t = 40$ scheduling scheme.

Table 5: Tasks to be processed at $t = 0$.

Job number	Process number	Machine number and corresponding processing time of each process					
		1	2	3	4	5	6
1	1	—	—	9.8238	5.6066	2.3884	—
1	2	—	8.7614	0.1616	—	4.5111	3.6049
1	3	2.2260	—	1.0555	5.9327	—	—
1	4	6.8426	5.3029	—	—	0.3179	2.4924
2	1	—	—	8.7605	3.2139	0.8977	2.2408
2	2	1.8753	27.4468	6.7044	5.2220	—	0.1501
2	3	1.0321	0.4354	0.4033	3.8629	0.2432	0.7356
2	4	4.1349	3.6649	—	4.2966	0.3085	0.7078
2	5	5.1805	—	6.7070	0.0218	6.3798	1.4679
3	1	7.4515	—	8.7112	11.4423	0.5015	0.4604
3	2	3.4696	—	0.1401	3.4382	—	0.8415

Table 5: Continued.

Job number	Process number	Machine number and corresponding processing time of each process					
		1	2	3	4	5	6
3	3	0.8281	—	12.5823	0.9464	—	14.4992
3	4	5.4673	—	3.5510	—	3.7151	4.3297
3	5	—	2.3391	—	2.3792	6.5705	0.4136
4	1	1.0062	3.9873	0.2861	6.3271	2.9692	0.7711
4	2	3.1558	0.3706	3.3700	—	4.1528	0.5581
4	3	15.1522	6.1692	—	5.3464	0.2628	—
5	1	1.3501	2.3151	6.9586	6.3153	3.9388	0.5668
5	2	—	7.8409	0.4757	—	7.1685	11.6505
5	3	1.5864	1.6647	2.4072	1.1779	—	0.3065
5	4	1.6864	4.8608	—	2.6172	—	4.6677
6	1	6.3858	—	9.5675	8.1202	—	1.0911
6	2	10.0866	4.2997	—	14.2298	2.1821	—
6	3	—	3.9033	0–.6444	8.5418	3.8565	—
6	4	0.9869	—	0.7095	2.5105	11.3434	21.0465
6	5	2.1080	—	5.2743	2.0611	1.2416	6.9164
7	1	1.2192	2.0500	4.6593	—	—	2.3203
7	2	13.4036	0.3065	6.4845	4.5084	0.1403	0.7197
7	3	12.0271	7.5283	2.3252	8.6630	2.1088	4.8157
7	4	3.4707	2.7486	—	6.7813	2.5525	10.2010
8	1	10.9951	9.7286	2.7084	8.0656	—	4.2686
8	2	6.5468	3.4921	1.6932	4.2582	2.3376	9.5210
8	3	1.2771	7.0129	—	—	0.0317	9.7530
8	4	10.1690	0.3012	4.4865	3.8201	3.8540	3.9938
9	1	—	0.8426	4.9192	—	16.1560	18.1553
9	2	3.9587	1.8530	—	—	0.8106	—
9	3	0.6760	—	—	2.2566	—	7.2321
9	4	0.3468	11.8284	0.2563	2.6967	—	—
10	1	0.6787	2.9536	1.5828	4.1907	0.8061	1.9341
10	2	11.2861	0.9612	3.3480	4.2695	—	5.6762
10	3	—	3.3068	5.3467	8.9140	—	1.1600
10	4	1.9494	—	—	3.0254	0.2133	—
10	5	9.7772	0.2504	1.4523	3.7225	11.8949	0.9377
10	6	4.8633	1.2156	10.1277	2.1841	4.6520	0.2748

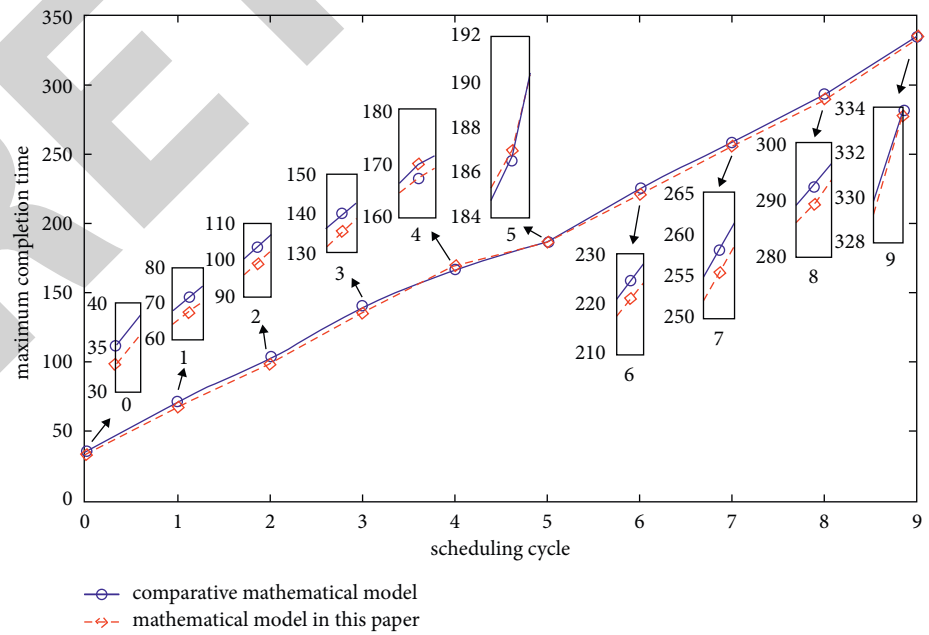


Figure 12: Comparison of the maximum completion time of each scheduling cycle.

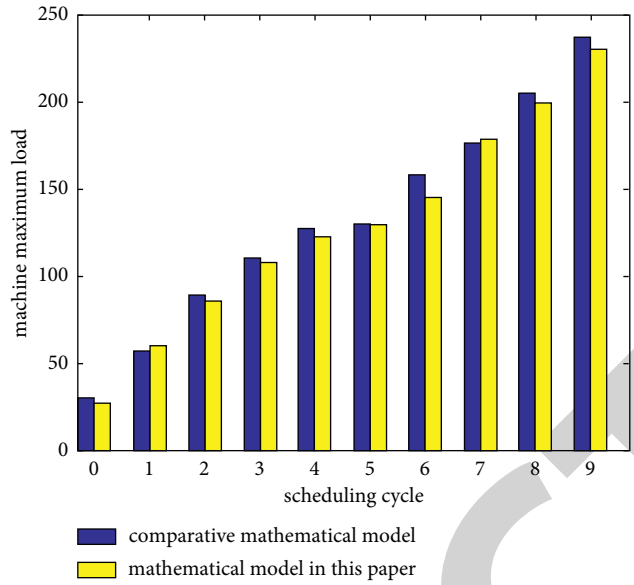


Figure 13: Comparison of the maximum load of machines in each scheduling cycle.

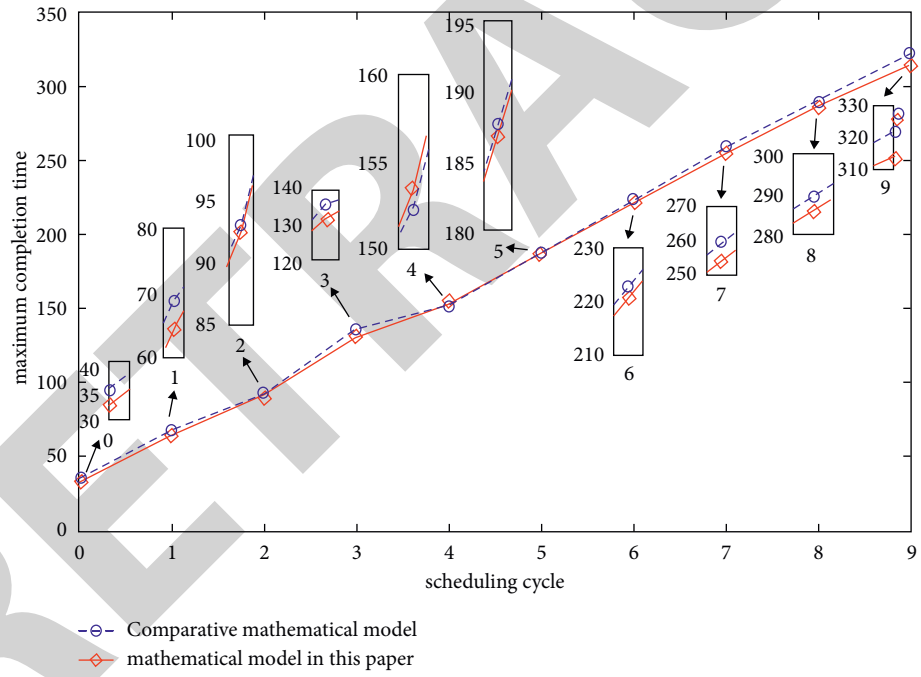


Figure 14: Comparison of the maximum completion time of each scheduling cycle.

Figure 14 demonstrates that when the mathematical model established in this paper continuously optimizes FJSSP, the completion time increases with the number of tasks to be processed. Except at the fourth scheduling cycle, the mathematical model in this paper achieves a smaller completion time in the rest of the scheduling cycles. Thus, compared with the mathematical model in Reference [23], the mathematical model in this paper is more efficient in solving FJSSP. In Figure 15, with the progress of the scheduling process, the stability of the workshop is enhanced. Except at the sixth scheduling cycle, the mathematical model in this paper achieves a smaller LB in the rest of the scheduling cycles. Accordingly, compared with the mathematical model in Reference [23], the current model achieves better system stability when solving FJSSP. Figure 14 illustrates that at the time of failure and the time of recovery, the mathematical model established in this paper can achieve smaller deviations of the old and new schemes. Therefore, the mathematical model proposed in this paper can maintain better stability in the rescheduling process after the workshop machine failure or machine failure recovery. A comprehensive observation of Figures 14–17 shows that machine M_5 fails at the fourth

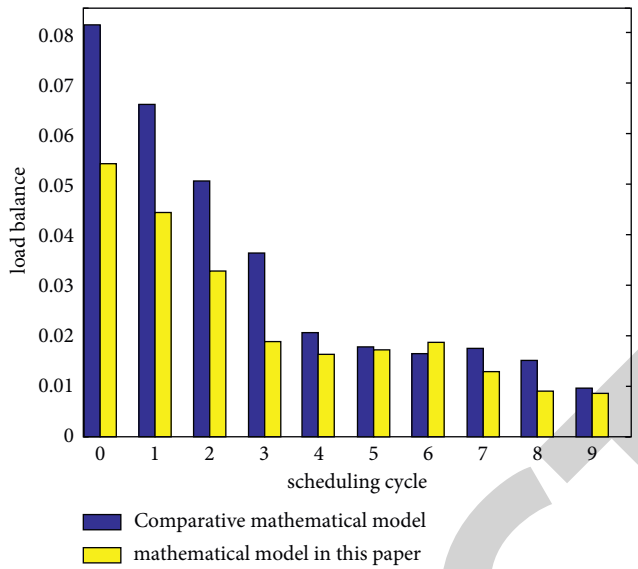
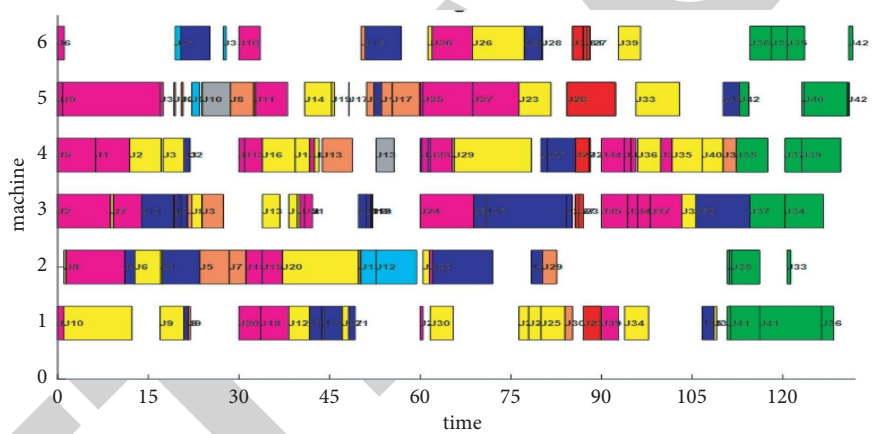
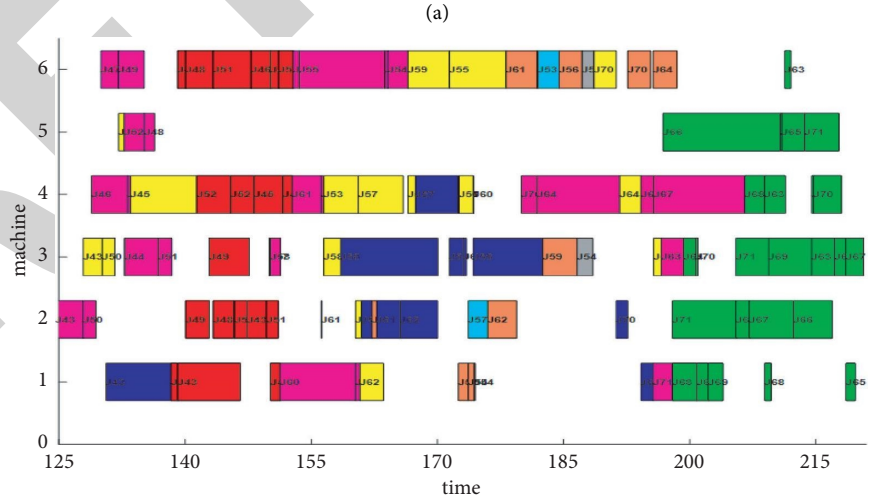


Figure 15: Comparison of LB for each scheduling cycle.

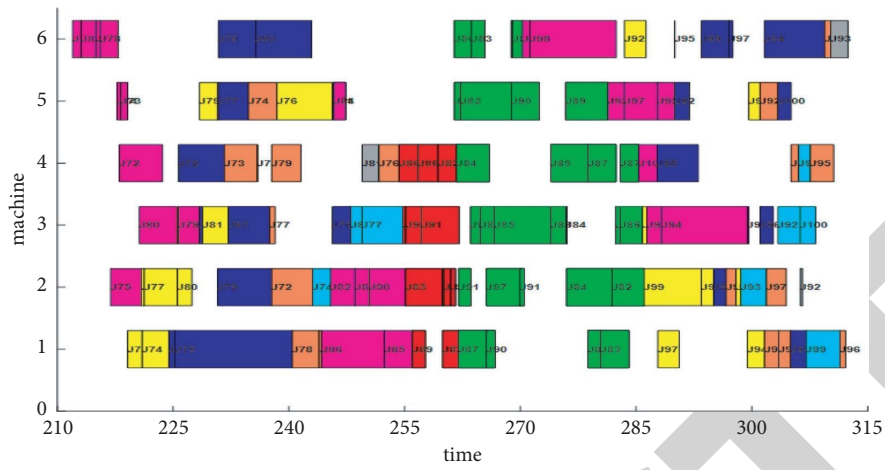


(a)



(b)

Figure 16: Continued.



(c)

Figure 16: Final scheduling scheme in this paper. (a) $t = 0$–$3t_1$. (b) $t = 4t_1$–$6t_1$. (c) $t = 7t_1$–$9t_1$.

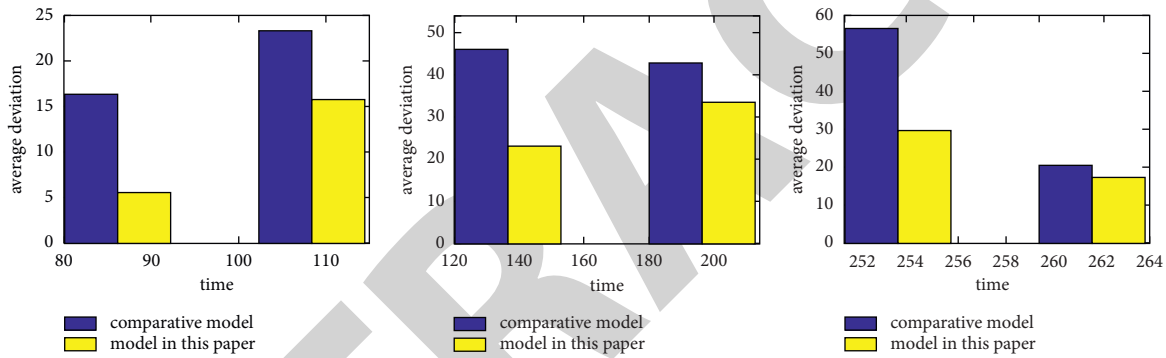


Figure 17: Comparison of deviations between old and new scheduling schemes during machine failure and failure recovery. (a) M2 failure. (b) M5 failure. (c) M6 failure.

scheduling cycle. Although the final scheduling scheme optimized by the mathematical model established in this paper is longer, it maintains better system stability, especially when the machine fault rescheduling achieves a smaller deviation; *M5* is recovered at the sixth scheduling cycle. Although the LB of the final scheduling scheme optimized using the mathematical model established in this paper is poor, the completion time is shorter, and a smaller deviation degree is achieved during machine fault rescheduling.

In summary, the mathematical model established in this paper has higher efficiency and better system stability when solving FJSSP. Moreover, in the workshop machine failure or failure recovery, it can pay attention to efficiency and stability at the same time. It has certain advantages for FJSSP optimization.

5. Conclusions

In this study, the characteristics and current research status of FJSSP are comprehensively analysed. A new mathematical model is established on the basis of three objectives: completion time, LB, and AD between new and old scheduling schemes. A dynamic multi-objective SSA is used to optimize the established model. The experimental results show that the dynamic multi-objective SSA can be used to solve FJSSP. The mathematical model established in this paper has improved efficiency and stability in solving dynamic multi-objective FJSSP. Especially when a workshop machine breaks down or gets back to work, the established model can keep enhanced balance between efficiency and stability. In the future, convergence rate of flexible job shop scheduling algorithms should be improved.

Data Availability

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

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This work was supported in part by Technological Innovation Development of Jilin under Grant no. 20210103090.

References

[1] X. Wu and S. Wu, "An elitist quantum-inspired evolutionary algorithm for the flexible job-shop scheduling problem," *Journal of Intelligent Manufacturing*, vol. 28, no. 6, pp. 1441–1457, 2017.

[2] L. Wan, K. Liu, Y. C. Liang, and T. Zhu, "DOA and polarization estimation for non-circular signals in 3-D millimetre wave polarized massive MIMO systems," *IEEE Wireless Communications*, vol. 20, no. 5, pp. 3152–3167, 2021.

[3] X. Wang, L. T. Yang, D. Meng, M. Dong, K. Ota, and H. Wang, "Multi-UAV cooperative localization for marine targets based on weighted subspace fitting in SAGIN environment," *IEEE Internet of Things Journal*, p. 1, 2021.

[4] J. Li, Y. He, X. Zhang, and Q. Wu, "Simultaneous localization of multiple unknown emitters based on UAV monitoring big data," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 9, pp. 6303–6313, 2021.

[5] T. Shu, J. He, and L. Li, "Near-field passive localization and gain-phase compensation with partly calibrated arrays," *IEEE Transactions on Aerospace and Electronic Systems*, p. 1, 2021.

[6] F. Wen, J. Shi, and Z. Zhang, "Closed-form estimation algorithm for EMVS-MIMO radar with arbitrary sensor geometry," *Signal Processing*, vol. 186, Article ID 108117, 2021.

[7] X. Huang and L. Yang, "A hybrid genetic algorithm for multi-objective flexible job shop scheduling problem considering transportation time," *International Journal of Intelligent Computing & Cybernetics*, vol. 2, no. 12, pp. 154–174, 2019.

[8] L. Liu and Y. Xi, "A hybrid genetic algorithm for job shop scheduling problem to minimize makespan," in *Proceedings of the 2006 6th World Congress on Intelligent Control and Automation*, pp. 3709–3713, Institute of Electrical and Electronics Engineers Inc, Dalian, China, June 2006.

[9] T. Meng, J. Duan, and Q. Pan, "An enhanced fruit fly optimization for the flexible job shop scheduling problem with lot streaming," in *Proceedings of the 2018 Chinese Control Conference*, pp. 2345–2349, IEEE Computer Society, Wuhan, China, July 2018.

[10] S. Kavitha, P. Venkumar, and N. Rajini, "An efficient social spider optimization for flexible job shop scheduling problem," *Journal of Advanced Manufacturing Systems*, vol. 17, no. 2, pp. 181–196, 2018.

[11] D. Lei, "Scheduling stochastic job shop subject to random breakdown to minimize make span," *International Journal of Advanced Manufacturing Technology*, vol. 55, no. 9-12, pp. 1183–1192, 2011.

[12] M. T. Jensen, "Generating robust and flexible job shop schedules using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 275–288, 2003.

[13] G. Chryssolouris and V. Subramaniam, "Dynamic scheduling of manufacturing job shops using genetic algorithms," *Journal of Intelligent Manufacturing*, vol. 12, no. 3, pp. 281–293, 2001.

[14] L. Zhang, L. Gao, and X. Li, "A hybrid genetic algorithm and tabu search for a multi-objective dynamic job shop scheduling problem," *International Journal of Production Research*, vol. 51, no. 12, pp. 3516–3531, 2013.

[15] Z. Huang, D. Tang, and M. Dai, "Bi objective optimization model of process planning and shop scheduling based on improved algorithm," *Journal of Nanjing University of Aeronautics & Astronautics*, vol. 47, no. 1, pp. 85–95, 2015.

[16] X. Shi, Y. Shi, and X. Yuan, "Discrete multi species swarm optimization algorithm for solving flexible job shop scheduling problem," *Information and Control*, vol. 44, no. 2, pp. 238–243, 2015.

[17] C. Chen, Z. Ji, and Y. Wang, "NSGA-II applied to dynamic flexible job shop scheduling problems with machine breakdown," *Modern Physics Letters B*, vol. 32, pp. 34–36, 2018.

[18] P. Fattahi and A. Fallahi, "Dynamic scheduling in flexible job shop systems by considering simultaneously efficiency and stability," *CIRP Journal of Manufacturing Science and Technology*, vol. 2, no. 2, pp. 144–123, 2010.

[19] X. Yang, "Who built the tide. genetic algorithm for solving flexible job shop scheduling problem," *Control and Decision*, vol. 10, pp. 1197–1200, 2004.

[20] M. Jain, V. Singh, and A. Rani, "A novel nature-inspired algorithm for optimization: squirrel search algorithm," *Swarm and Evolutionary Computation*, vol. 44, pp. 148–175, 2019.

[21] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[22] M. Tavana, Z. Li, M. Mobin, M. Komaki, and E. Teymourian, "Multi-objective control chart design optimization using NSGA-III and MOPSO enhanced with DEA and TOPSIS," *Expert Systems with Applications*, vol. 50, no. 5, pp. 17–39, 2016.

[23] C. P. Li, H. Y. Cui, and G. C. Wang, "The optimization of flexible job-shop scheduling problem based on NSGA-II," *Advanced Materials Research*, vol. 651, pp. 684–687, 2013.

[24] C. Zhang, Y. Rao, and X. Liu, "Based on POX crossover genetic algorithm to solve job-shop scheduling problem," *Chinese mechanical engineering*, vol. 23, pp. 83–87, 2004.