Int j simul model 19 (2020) 3, 517-528
ISSN 1726-4529 Original scientific paper

A DISCRETE JOB-SHOP SCHEDULING ALGORITHM BASED ON IMPROVED GENETIC ALGORITHM

Zhang, H.[#] & Zhang, Y. Q.
School of Information Engineering, Yulin University, Yulin 719000, China
E-Mail: zhanghui@yulinu.edu.cn, zhangyaqiong@yulinu.edu.cn ([#] Corresponding author)

Abstract
In a discrete job-shop, the scheduling objectives are often conflicting and constrained, and the actual production is disturbed by many uncertainties. This paper combines bi-directional scheduling with genetic algorithm (GA) to optimize the static scheduling results. Considering the dynamicity of the various emergencies in the discrete job-shop, a multi-level dynamic scheduling model was proposed based on rolling window. The model integrates the merits of periodic rescheduling and event-driven rescheduling to reduce the scheduling cost and mitigate the impact of disturbances, without sacrificing the stability and efficiency of production. Our method was verified through discrete event simulation.
(Received in May 2020, accepted in August 2020. This paper was with the authors 2 months for 1 revision.)

Key Words: Discrete Job-Shop Scheduling Problem (DJSP), Bi-Directional Scheduling, Genetic Algorithm (GA), Rolling Window, Discrete Event Simulation

1. INTRODUCTION

In a discrete job-shop, the actual production faces many uncertain disturbances. The typical disturbances include machine failure, emergency order insertion, and order change. However, the timing and method of rescheduling are often decided empirically by job-shop schedulers. Lacking sufficient information and intelligent management, the rescheduling plan could lead to low production efficiency and high production cost. Therefore, a growing attention has been paid to the dynamic scheduling of the discrete job-shop, with the aims to effectively handle the disturbances to production, generate a robust rescheduling plan, and prevent the cost increment and confusion induced by excessive rescheduling.

This paper attempts to optimize the scheduling plan for the discrete job-shop. Firstly, a mathematical model was established for discrete job-shop scheduling, aiming to minimize the processing cycle and ensure the simultaneous arrival of jobs at the assembly process [1, 2]. Next, the genetic algorithm (GA) was improved to support the bi-directional scheduling of the discrete job-shop [3-5]. The key jobs were subject to reverse scheduling to ensure timely completion, while common jobs were subject to forward scheduling to improve machine utilization. Finally, rolling window was introduced for rescheduling under the effects of local dynamic disturbances.

2. LITERATURE REVIEW

Discrete job-shop scheduling problem (DJSP) is non-deterministic polynomial-time (NP) hard [6, 7]. In 1954, Johnson [8] proposed a polynomial time algorithm for the flow-shop scheduling problem (FSSP) with two machines, marking the start of theoretical research into job-shop scheduling problem (JSP). Subsequently, Artigues and Feillet [9] applied mathematical programming to solve the JSP. Karsiti et al. [10] introduced the priority distribution principle into job-shop scheduling and proposed a job-shop scheduling algorithm based on heuristic dispatching rules. Lin and Liao [11] probed into a two-stage hybrid flow-shop with multiple parallel machines in the first stage. Portmann et al. [12] presented a branch and bound (B&B) method for hybrid flow-shop and explained the specific steps to solve the complex hybrid FSSP.

Focusing on the multi-level assembly problem, Kubiak et al. [13] divided all jobs into several levels, and assembled each job step by step, until the completion of the assembly task. Fattahi et al. [14] established a mathematical model for a hybrid job-shop with assembly under the constraints of hybrid job-shop. Liu et al. [15] investigated the hybrid production problem in discrete job-shop and designed a systematic manufacturing execution system under hybrid production mode. Hamidian and Seyedpoor [16] developed a fuzzy inference system to select machines and balance machine load. Ye et al. [17] pointed out the research direction and implementation path of intelligent discrete job-shops.

Many efficient techniques, such as artificial intelligence (AI) and neural network (NN) have been widely applied to the JSP. Agrawal et al. [18] adopted the GA to solve the multi-objective JSP with priority constraints on parallel machines. Li et al. [19] employed simulated annealing (SA) algorithm to minimize the maximum makespan and minimize the completion time of each job. Kashan et al. [20] combined the GA with bi-directional scheduling algorithm to optimize the delivery time and total makespan of key jobs in the JSP. Sarkheyli et al. [21] applied particle swarm optimization (PSO) and shuffled frog leaping algorithm (SFFA) to the DJSP. Based on hybrid GA, Rosyid et al. [22] set up a job-shop scheduling evaluation system to optimize factors like time, machine utilization, production cost, and consumer satisfaction. Wang and Tang [23] coupled the NN with the GA to solve the JSP.

The dynamic job-shop scheduling methods can be divided into traditional methods and intelligent methods [24]. Fang and Xi [25] invented the rolling window technology, which segments a dynamic and random JSP into several static intervals. Zuo and Xue [26] explored the strategy of rescheduling cycle based on the GA and mined the job selection principle in rolling window. Chu et al. [27] studied the processing methods of various disturbances, aiming to penalize completion delays and minimize the maximum makespan. Gao et al. [28] classified the common disturbances of flexible job-shop and treated the deviation between scheduling and rescheduling plans as the evaluation index. Shahrabi et al. [29] put forward a dynamic scheduling method for flexible job-shop based on variable rescheduling intervals; at machine failure, the affected job will be processed by a machine with the same processing ability. Wang et al. [30] proposed an energy-aware dynamic scheduling method to overcome the uneven load and energy consumption increment in flexible job-shop under disturbances.

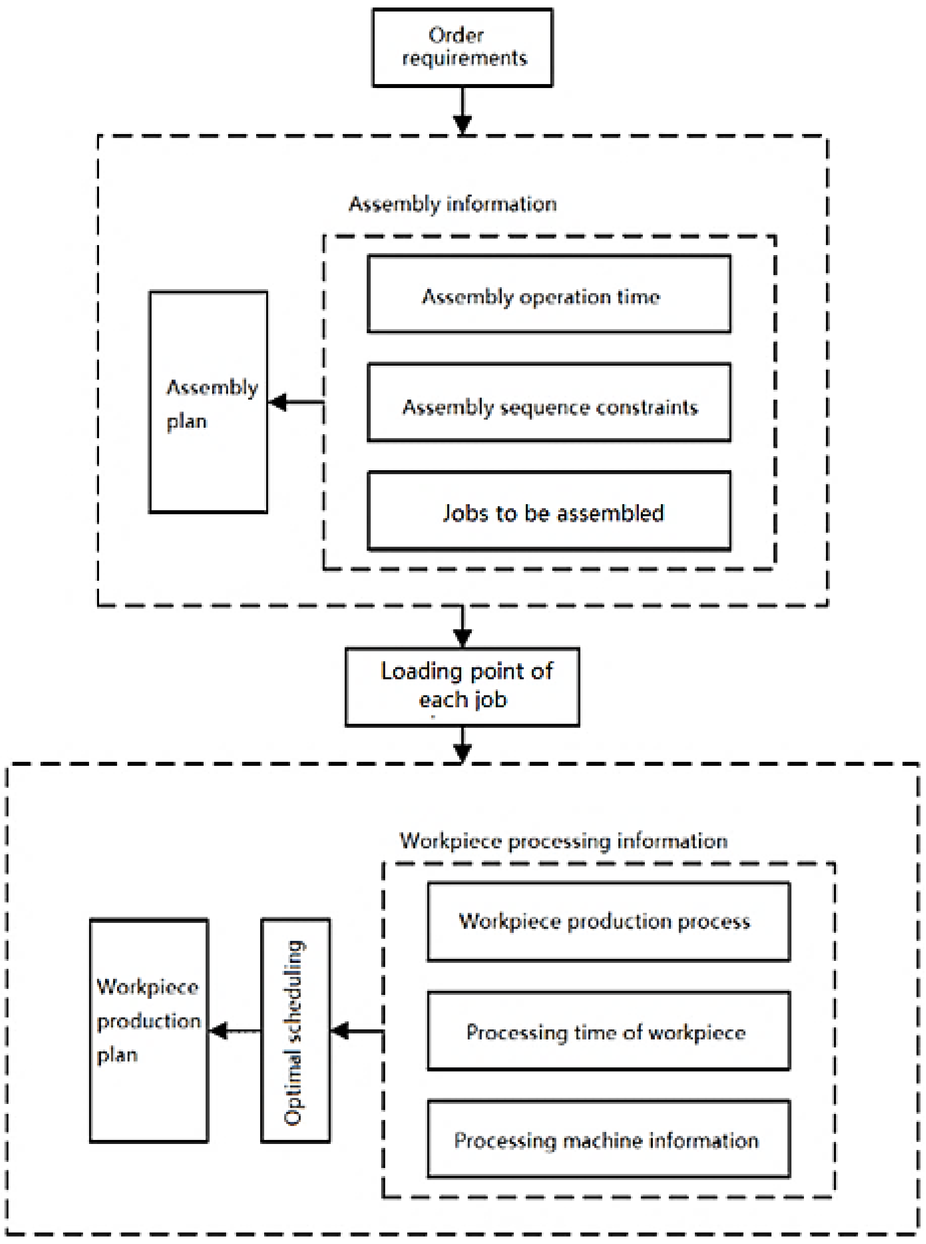
3. PROBLEM DESCRIPTION

3.1 Problem definition

For a discrete job-shop with *n* jobs and *m* machines, the DJSP aims to rationalize the operation sequence of each job and optimize the given performance index under multiple constraints; the machines and processing time of each operation are determined, and each job has a definite loading point. To reflect the actual situation of the job-shop, the scheduling of the discrete job-shop should meet multiple constraints:

- All machines are idle at time zero.
- The jobs are independent of each other, and the operation sequence of each job has a negligible effect on the production process.
- Each job must be processed strictly as per the process route, and the next operation cannot be started before the completion of the current operation.
- For the jobs in the same assembly process, the assembling cannot start until all the jobs have been processed.
- Each machine can only process one job at a time; each job can only be processed on one machine at a time; each job has at least one operation.
- The preparation time and transport time of a job are included in the processing time.

On this basis, the discrete job-shop scheduling can be illustrated by Fig. 1 below.


Order requirements
Assembly information
Assembly operation time
Assembly sequence constraints
Jobs to be assembled
Assembly plan
Loading point of each job
Workpiece processing information
Workpiece production process
Processing time of workpiece
Processing machine information
Optimal scheduling
Workpiece production plan
Figure 1: The block diagram of discrete job-shop scheduling.

3.2 Performance indices

The main performance indices of the JSP are introduced as follows.

(1) Time-based indices

There are two performance indices with time as the optimization objective: the maximum makespan and mean flow time. The maximum makespan refers to the completion time of the last operation when the job-shop processes multiple jobs at the same time. The shorter the maximum makespan, the better the production efficiency. Hence, the maximum makespan should be minimized:

$$f_t = \min\,(max_{i=1}^{n} CT_i) \tag{1}$$

where, CT_i is the completion time of job i; n is the number of jobs.

The mean flow time refers to the average of the time of all jobs from the start time to the end time. The smaller the mean flow time, the shorter the waiting time, and the better the production efficiency. Thus, the mean flow time should also be minimized:

$$f_f = \min\{\tfrac{1}{n}\textstyle\sum_{i=1}^{n}(CT_i - ST_i)\} \tag{2}$$

where, ST_i represents start time of job i.

(2) Delivery-based indices

The delivery date is the completion time of the order agreed by the job-shop. The scheduling performance improves as the order completion time approaches the due date. Therefore, the lead time and delay time of order completion time should both be minimized:

$$f_l = \min\{\max[(DT - CT), 0]\} \tag{3}$$

$$f_d = \min\{\max[(CT - DT), 0]\} \tag{4}$$

where, DT is the delivery date; CT is the order completion time.

(3) Machine-based indices

The performance indices based on machines include job-shop utilization and machine utilization. Job-shop utilization refers to the ratio of working time to total time of all machines in a period. The higher the job-shop utilization, the better the use of resources in the job-shop. As a result, the job-shop utilization should be maximized:

$$f_{ju} = max \textstyle\sum_{i=1}^{n}\sum_{j=1}^{n_i}\sum_{k=1}^{m}(E_{ijk} - S_{ijk})/mCT_l \tag{5}$$

where, m is the total number of machines; k is the serial number of machines; j is the serial number of operations of job i; n_i is the number of operations of job i; E_{ijk} is the completion time of operation j of job i on machine k; S_{ijk} is the start time of operation j of job i on machine k; CT_l is the completion time of the last operation.

Machine utilization refers to the utilization difference between the best utilized machine and the worst utilized machine. The value of this index is negatively correlated with the load balance of all machines in the job-shop. Hence, the machine utilization should be minimized:

$$f_{mu} = min\,(max_{k=1}^{m}\textstyle\sum_{i=1}^{n}\sum_{j=1}^{n_i}X_{ijk}T_{ijk} - min_{k=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n_i}X_{ijk}T_{ijk})/CT_l \tag{6}$$

where, X_{ijk} is a decision variable (if operation j of job i is processed on machine k, then $X_{ijk} = 1$); T_{ijk} is the processing time of operation j of job i on machine k.

(4) Cost-based index

The cost-based index reveals the impact of scheduling on the economic benefits of the job-shop. In dynamic job-shop scheduling, the most used cost-based index is the rescheduling cost. Since all job-shops are profit-seeking, the rescheduling cost should be minimized:

$$f_c = \min(\tau YT + \varphi LT + \omega N) \tag{7}$$

where, τ is coefficient of overtime pay; φ is the coefficient of inventory cost; ω is the scheduling cost of task change; YT and LT are the delay time and lead time of the order, respectively; N is the number of machines subject to task change.

(5) Loading point-based indices

To reflect the coordination effect of production and assembly plans, three performance indices were designed based on the loading point, namely, the total delay time, the total lead time, and the assembly simultaneity. The total delay time is the sum of the delays of the jobs at each loading point. Since it is negatively correlated with scheduling performance, the total delay time should be minimized:

$$f_{lp} = \min P = \min\{\textstyle\sum_{i=1}^{n}\sum_{l=1}^{L}\max[x_{il}(CT_i - d_l), 0]\} \tag{8}$$

where, P is the total delay time; l is the serial number of loading points; L is the total number of loading points; d_l is the delay time of the lth loading point; x_{il} is a decision variable (if job i is assembled at the lth loading point, then $x_{il} = 1$).

The total lead time is the sum of the leads of the jobs at each loading point. Since it is negatively correlated with scheduling performance, the total lead time should be minimized:

$$f_{le} = \min E = \min \{\sum_{i=1}^{n} \sum_{l=1}^{L} \max [x_{il}(d_l - CT_i), 0]\} \tag{9}$$

where, E is the total lead time.

The assembly simultaneity is the sum of the difference between the completion times of the last job and other jobs at the same loading point. Since it is negatively correlated with scheduling performance, the assembly simultaneity should be minimized:

$$f_a = \min (max_{i=1}^{n_l} \sum_{l=1}^{w} CT_{li}^{a} - \sum_{l=1}^{w} \sum_{i=1}^{n_l} CT_{li}^{a}) \tag{10}$$

where, CT_{li}^{a} is the completion time of job i at loading point l; w is the total number of assembly operations; n_l is the number of jobs at loading point l.

4. BI-DIRECTIONAL DISCRETE JOB-SHOP SCHEDULING BASED ON IMPROVED GENETIC ALGORITHM

There are generally two kinds of scheduling methods for the DJSP: forward scheduling for early delivery of all jobs, and reverse scheduling to meet the delivery date or loading point. In practice, different scheduling methods should be applied to different jobs. For key jobs, the reverse scheduling was adopted to complete them on time for the assembly at the loading point(s). For common jobs, forward scheduling was implemented to complete the order as soon as possible, without affecting the production of key jobs. The combination of forward scheduling and reverse scheduling is called bi-directional scheduling:

$$f = \min \{max_{i=1}^{n} CT_i + \gamma \times \sum_{i=1}^{n^b} \left|ET_i^b - d_i\right| + \delta \times (max_{i=1}^{w_l} \sum_{l=1}^{w} CT_{li}^{a} - \sum_{l=1}^{w} \sum_{i=1}^{n_l} CT_{li}^{a})\} \tag{11}$$

where, ET_i^b is the completion time of key job i; d_i is the loading point of key job i; n^b is the number of key jobs; γ is the weight coefficient of the difference between completion time of key jobs and loading point (hereinafter referred to as the loading point difference); δ is the weight coefficient of assembly simultaneity of key jobs. The bi-directional scheduling needs to meet the following constraints:

$$\sum_{k=1}^{m} X_{ijk} = 1 \quad \forall i, j \tag{12}$$

$$\begin{gathered} ST_{ij} - PT_{uvk} \geq ST_{uv} \\ Y_{ijuvk} = 1, X_{ijk} = 1, X_{uvk} = 1 \end{gathered} \tag{13}$$

$$ST_{i(j+1)} - ST_{ij} \geq PT_{ijk} \;\; j + 1 \leq p_k, X_{ijk} = 1 \tag{14}$$

where, m is the number of machines; p_i is the number of operations of job i; Y_{ijuvk} is a discriminator of different operations on the same machine k (if operation j of job i starts after the completion of operation v of job u, then $Y_{ijuvk} = 1$); ST_{ij} is the start time of operation j of job i; PT_{ijk} is the processing time for operation j of job i on machine k.

Facing the NP-hard problem of DJSP, it is very difficult to find the optimal solution accurately. Fortunately, the GA [31], which transforms the candidate solutions to the problem into chromosomes through encoding, has obvious advantages in solving this kind of problems.

The scheduling of the discrete job-shop is equivalent to sorting the operations of jobs and determining the processing time of each operation. As a result, the operations were treated as the basic units of chromosome encoding. Specifically, the sum of operations of a job was taken as the total length of the corresponding chromosome, in which each gene is represented by an integer. Then, the operations of the same job were assigned the same integer, such that each integer represents the serial number of a job.

Considering the features of bidirectional scheduling, the original GA was improved to contain the key jobs and common jobs. The decoding of the improved GA is as follows:

Step 1. Prioritizing each type of jobs: Let t^b and t^c be the set of key jobs, and the set of common jobs, respectively. Because the key jobs should be reversely scheduled before the forward scheduling of common jobs, set the priority of the operations of the first key job to 1-1, the priority of the operations of the second key job to 1-2, …, and the priority of the gene of the first common job to 2-1.

Step 2. Renumbering operations of key jobs: Renumber the operations of each key jobs in reverse order to the process route.

Step 3. Reversely scheduling key jobs

(1) Taking the loading point as the start point, compute the start time of each key job: set the start time of the last job at the loading point to 0, and compute the start time of other jobs by:

$$st_i^b = max_{i=1}^{n_l} d_i - d_i \tag{15}$$

where, st_i^b is the start time of key job i.

(2) Calculate the start time ST_{ij} and completion time CT_{ij} of operation O_{ij}^b of the key job with the highest priority by:

$$ST_{ij} = st_i^b \tag{16}$$

$$\begin{gathered} CT_{ij} = ST_{ij} + T_{ijk} \\ X_{ijk} = 1 \end{gathered} \tag{17}$$

(3) Check if machine k is working from ST_{ij} to CT_{ij}. If yes, assume that the operation on the machine in that period is O_{mn}, and set $ST_{ij} = CT_{mn}$ and $CT_{ij} = ST_{ij} + T_{ijk}$. Repeat until operation O_{ij}^b is assigned to the idle period of machine k.

(4) If $CT_{ij} > d_i$, it is impossible to find a reasonable start time for job i through reverse scheduling. Then, move job i to the job set t for forward scheduling, and remove it from the priority list of reverse scheduling. After that, restart reverse scheduling from Step (1). If $CT_{ij} \leq d_i$, proceed to the next step.

(5) Update the start time of each job by:

$$st_i^b = CT_{ij} \tag{18}$$

(6) Repeat Steps (2) to (5) until the scheduling of key jobs is completed.

(7) Calculate the actual situation of scheduling:

$$ST_{ij} = max_{i=1}^{n_l} d_i - CT_{ij} \tag{19}$$

$$CT_{ij} = max_{i=1}^{n_l} d_i - ST_{ij} \tag{20}$$

Step 4. Forwardly scheduling common jobs

(1) According to the priority setting rule for common jobs, adjust the priorities of the operations of the key jobs requiring forward scheduling, such that they are higher than those of common jobs.

(2) Set the earliest available time of each machine Mt_k= 0, and the earliest start time of job i of forward scheduling to $r_i = 0, i \in t$.

(3) Calculate the start time ST_{ij} and completion time CT_{ij} of operation O_{ij} of the job with the highest priority by:

$$ST_{ij} = r_i \tag{21}$$

$$CT_{ij} = ST_{ij} + T_{ijk} \quad X_{ijk} = 1 \tag{22}$$

(4) Check if machine k is working from ST_{ij} to CT_{ij}. If yes, assume that the operation on the machine in that period is O_{mn}, and set $ST_{ij} = CT_{mn}$ and $CT_{ij} = ST_{ij} + T_{ijk}$.

(5) Repeat Step (4) until operation O_{ij} is assigned a reasonable period for processing.

(6) Set $r_i = CT_{ij}$.

(7) Repeat Steps (3) to (6) until the scheduling of all jobs is completed.

5. MULTI-LEVEL DYNAMIC SCHEDULING BASED ON ROLLING WINDOW

The dynamic DJSP can be described as: the job-shop has m machines, each of which can process some operations, and n jobs to be processed at the same time. The jobs fall into two categories: key jobs and common jobs. The key jobs should subject to reverse scheduling, while the common jobs to forward scheduling. The operation sequence of each job remains unchanged, but the number of jobs to be scheduled may change with the start of rescheduling. Each job has a loading point, and each loading point contains multiple jobs. During the production, disturbances will occur from time to time. The task of scheduling is to allocate the operations of each job to suitable machines, and optimize the scheduling objectives under the following assumptions:

- The operation sequence of each job remains unchanged.
- The machine and processing time of each operation are determined.
- During rescheduling, the operation underway will not be affected, unless there is a machine failure, that is, the operation will continue until it is completed.
- Each machine can only process one operation at a time.

Considering the effect and stability of the scheduling process, the maximum makespan, the total loading point difference, and the assembly simultaneity of key jobs were selected as the optimization objectives of scheduling and rescheduling:

$$f = \min \{max_{i=1}^{n} CT_i + \gamma \times \sum_{i=1}^{n^b} \left| ET_i^b - d_i \right| + \delta \times (max_{i=1}^{w_l} \sum_{l=1}^{w} CT_{li}^a - \sum_{l=1}^{w} \sum_{i=1}^{n_l} CT_{li}^a)\} \quad (23)$$

Rolling window decomposes a dynamic scheduling problem into a series of static scheduling intervals called rolling intervals. Initially, some jobs are selected by certain rules to enter the processing window, generating the initial scheduling plan. The initial plan needs to be adjusted in the event of any disturbance: the completed jobs are moved out of the window, and new jobs are selected by the said rules to enter the window. Then, the window is scheduled statically to produce a new scheduling plan.

To deal with disturbances pertinently, the first step is to classify the impact of disturbances into multiple levels and select the proper rescheduling strategy for each level. In this paper, the disturbances are divided into three levels based on their impacts. A third-level disturbance only influences the time of operation. If a third-level disturbance occurs, the impact can be mitigated through right-shift scheduling.

A second-level disturbance can be solved through local rescheduling, which keeps the system stable and saves the scheduling cost. During local rescheduling, the forward scheduling is performed first to utilize the idle time of the machines. If the result is undesirable and if the completion time of the job to be processed exceeds the deadline (LT_k), the sequence of the affected operations on the machine should be adjusted.

A first-level disturbance has a significant impact on the scheduling plan. If a first-level disturbance occurs, local rescheduling is not enough to satisfy the local assembly needs. Instead, the scheduling plan should be updated by selecting the jobs to enter the window and reschedule all the jobs in the window. Fig. 2 presents the workflow of the multi-level dynamic scheduling strategy.

6. DISCRETE EVENT SIMULATION AND RESULTS ANALYSIS

6.1 Simulation on bi-directional scheduling algorithm

To verify its feasibility, the bi-directional scheduling algorithm, proposed based on improved GA, was simulated on a production task (Tables I and II) with six jobs, each of which has six operations, in a job-shop with six machines. Among them, jobs 1-4 are key jobs, and jobs 5 and 6 are common jobs. The loading point of each job is given in Table III.

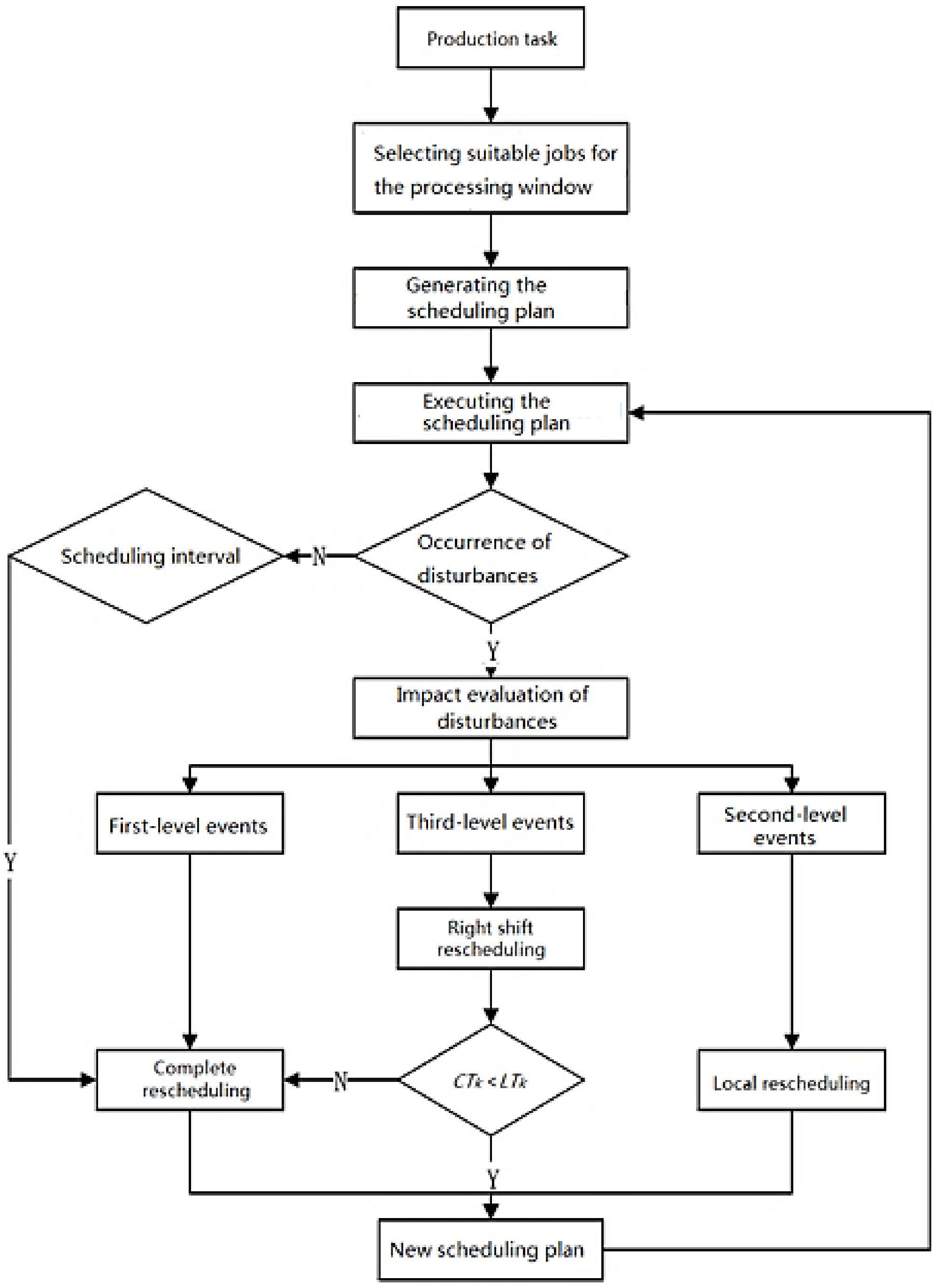


Figure 2: The workflow of multi-level dynamic scheduling strategy.

Table I: The processing time of each operation.

Job \ Operation	P_1	P_2	P_3	P_4	P_5	P_6
W_1	1	3	7	8	4	5
W_2	2	3	8	9	3	2
W_3	7	6	9	8	9	3
W_4	3	4	5	5	7	9
W_5	8	3	6	4	3	2
W_6	5	4	7	8	2	6

Table II: The machines for each job.

Job \ Machine	M_1	M_2	M_3	M_4	M_5	M_6
W_1	2	1	3	4	5	6
W_2	1	4	5	2	6	3
W_3	2	4	6	5	1	3
W_4	3	2	5	4	1	6
W_5	3	4	5	6	1	2
W_6	2	4	6	1	5	3

Table III: The loading point of each job.

Job	W_1	W_2	W_3	W_4	W_5	W_6
Loading point	60	60	65	65	60	65

During the simulation, the bi-directional scheduling algorithm was applied to optimize the scheduling plan. The GA parameters were configured as follows: population size 200, crossover probability 0.75, mutation probability 0.05, and the maximum number of iterations 200. The optimal scheduling result is that the key jobs 1-4 are all completed on time at their loading points, and the processing cycle is 63.

For comparison, other GA-based scheduling methods were adopted to schedule the same example. The GA-based forward scheduling method had a shorter processing cycle than the proposed algorithm (59 vs. 63) but delayed the completion of key job 3. Assuming that all jobs are common, the GA-based reverse scheduling method ended up with a processing cycle of 67 and delayed the completion of job 6. Obviously, the proposed bi-directional scheduling method outperforms the forward and reverse scheduling methods in feasibility and rationality. It is capable of completing all key jobs on time at the loading points, without significantly extending the processing cycle.

6.2 Simulation on multi-level dynamic scheduling model based on rolling window

The above example was still adopted for simulation. The deadline of each job is shown in Table IV.

Table IV: The deadline of each job.

Job	W_1	W_2	W_3	W_4	W_5	W_6
Deadline	70	70	75	75	80	80

In this simulation, machine failure was taken as a typical three-level disturbance. Once the failure occurs, the job on the machine and the subsequent operations will be affected. In this case, it is impossible to eliminate the impact of the failure using the idle time of the machines. Then, the processing time of all affected jobs needs to be adjusted through reverse rescheduling. In total, 8 operations were adjusted, and the total change time was 32, indicating that the right shift rescheduling is reasonable and efficient.

Then, the scrapping of some jobs was considered as a typical second-level disturbance, which calls for local rescheduling. Since no job has been completed, 7 jobs need to be rescheduled. After the scheduling, all job could be completed before the deadlines, indicating that the local rescheduling meets the requirements of the assembly plan. Compared with the original scheduling plan, the total change time was 36, and the change is small. Thus, the scheduling cost is very small, revealing the rationality of local rescheduling.

Finally, the change of assembly plan at time 30 was viewed as a typical first-level disturbance. After the change, job 6 became a key job with the loading point of 65, and job 5 was removed from the production task. Then, a complete rescheduling was carried out through simulation. The simulation results show that, with rolling window technology, the dynamic DJSP was transformed into several static scheduling intervals. By solving each static scheduling interval, the scheduling plan could adapt to the complex and continuously changing job-shop environment, and the disturbances could be handled quickly.

The above simulations show that the proposed multi-level dynamic scheduling method can effectively optimize the production and assembly of the DJSP.

7. CONCLUSIONS

This paper mainly explores the static and dynamic scheduling of the discrete job-shop. In view of the features of the discrete job-shop, a mathematical model was established for the bi-directional scheduling of the DJSP. Then, the GA was improved to create a multi-level event-driven rescheduling method. The proposed model and method were proved effective and reasonable through simulations. The future research will solve the multi-objective JSP with other versions of the GA, namely, the fast non-dominated genetic algorithm (NGSA), making the objectives more in line with actual production needs.

ACKNOWLEDGEMENTS

This work is supported by Research and Development for Science and Technology Program of Shaanxi Province (No. 2020NY-163, No. 2020NY-170); Research and Development for Science and Technology Program of Yulin (No. 2019-77-2); Young Elite Scientist Sponsorship Program by the Yulin Association for Science and Technology (No. 20190127).

REFERENCES

[1] Chen, W.; Hao, Y. F. (2018). Genetic algorithm-based design and simulation of manufacturing flow shop scheduling, *International Journal of Simulation Modelling*, Vol. 17, No. 4, 702-711, doi:10.2507/IJSIMM17(4)CO17

[2] Nazifa, T. H.; Ramachandran, K. K. (2018). Exploring the role of information sharing in supply chain management: a case study, *Journal of System and Management Sciences*, Vol. 8, No. 4, 13-37

[3] Fu, H. C.; Liu, P. (2019). A multi-objective optimization model based on non-dominated sorting genetic algorithm, *International Journal of Simulation Modelling*, Vol. 18, No. 3, 510-520, doi:10.2507/IJSIMM18(3)CO12

[4] Jemilda, G.; Baulkani, S. (2018). Moving object detection and tracking using genetic algorithm enabled extreme learning machine, *International Journal of Computers Communications & Control*, Vol. 13, No. 2, 162-174, doi:10.15837/ijccc.2018.2.3064

[5] Li, Y.; Shi, D.; Bu, F. (2019). Automatic recognition of rock images based on convolutional neural network and discrete cosine transform, *Traitement du Signal*, Vol. 36, No. 5, 463-469, doi:10.18280/ts.360512

[6] Hanen, C. (1994). Study of a NP-hard cyclic scheduling problem: the recurrent job-shop, *European Journal of Operational Research*, Vol. 72, No. 1, 82-101, doi:10.1016/0377-2217(94)90332-8

[7] Wang, C.; Zeng, L. (2019). Optimization of multi-objective job-shop scheduling under uncertain environment, *Journal Européen des Systèmes Automatisés*, Vol. 52, No. 2, 179-183, doi:10.18280/jesa.520210

[8] Johnson, S. M. (1954). Optimal two-and three-stage production schedules with setup times included, *Naval Research Logistics Quarterly*, Vol. 1, No. 1, 61-68, doi:10.1002/nav.3800010110

[9] Artigues, C.; Feillet, D. (2008). A branch and bound method for the job-shop problem with sequence-dependent setup times, *Annals of Operations Research*, Vol. 159, No. 1, 135-159, doi:10.1007/s10479-007-0283-0

[10] Karsiti, M. N.; Cruz Jr, J. B.; Mulligan Jr, J. H. (1992). Simulation studies of multilevel dynamic job shop scheduling using heuristic dispatching rules, *Journal of Manufacturing Systems*, Vol. 11, No. 5, 346-358, doi:10.1016/0278-6125(92)90063-L

[11] Lin, H.-T.; Liao, C.-J. (2003). A case study in a two-stage hybrid flow shop with setup time and dedicated machines, *International Journal of Production Economics*, Vol. 86, No. 2, 133-143, doi:10.1016/S0925-5273(03)00011-2

[12] Portmann, M.-C.; Vignier, A.; Dardilhac, D.; Dezalay, D. (1998). Branch and bound crossed with GA to solve hybrid flowshops, *European Journal of Operational Research*, Vol. 107, No. 2, 389-400, doi:10.1016/S0377-2217(97)00333-0

[13] Kubiak, W.; Steiner, G.; Yeomans, J. S. (1997). Optimal level schedules for mixed-model, multi-level just-in-time assembly systems, *Annals of Operations Research*, Vol. 69, 241-259, doi:10.1023/A:1018985029260

[14] Fattahi, P.; Rad, N. B.; Daneshamooz, F.; Ahmadi, S. (2020). A new hybrid particle swarm optimization and parallel variable neighborhood search algorithm for flexible job shop scheduling with assembly process, *Assembly Automation*, Vol. 40, No. 3, 419-432, doi:10.1108/AA-11-2018-0178

[15] Liu, L. L.; Liu, X. W.; Wang, S.; Zhou, W.; Zhao, G. P. (2015). Multi-objective optimization algorithm for job shop scheduling problem in discrete manufacturing enterprise, *Applied Mechanics and Materials*, Vol. 741, 860-864, doi:10.4028/www.scientific.net/AMM.741.860

[16] Hamidian, D.; Seyedpoor, S. M. (2010). Shape optimal design of arch dams using an adaptive neuro-fuzzy inference system and improved particle swarm optimization, *Applied Mathematical Modelling*, Vol. 34, No. 6, 1574-1585, doi:10.1016/j.apm.2009.09.001

[17] Ye, Z.; Chen, Q.; Zhang, Y.; Zou, J.; Zheng, Y. (2019). Identification of vortex structures in flow field images based on convolutional neural network and dynamic mode decomposition, *Traitement du Signal*, Vol. 36, No. 6, 501-506, doi:10.18280/ts.360604

[18] Agrawal, R.; Pattanaik, L. N.; Kumar, S. (2012). Scheduling of a flexible job-shop using a multi-objective genetic algorithm, *Journal of Advances in Management Research*, Vol. 9, No. 2, 178-188, doi:10.1108/09727981211271922

[19] Li, K.; Yang, S.-L.; Ma, H.-W. (2011). A simulated annealing approach to minimize the maximum lateness on uniform parallel machines, *Mathematical and Computer Modelling*, Vol. 53, No. 5-6, 854-860, doi:10.1016/j.mcm.2010.10.022

[20] Kashan, A. H.; Karimi, B.; Jolai, F. (2010). An effective hybrid multi-objective genetic algorithm for bi-criteria scheduling on a single batch processing machine with non-identical job sizes, *Engineering Applications of Artificial Intelligence*, Vol. 23, No. 6, 911-922, doi:10.1016/j.engappai.2010.01.031

[21] Sarkheyli, A.; Zain, A. M.; Sharif, S. (2015). The role of basic, modified and hybrid shuffled frog leaping algorithm on optimization problems: a review, *Soft Computing*, Vol. 19, No. 7, 2011-2038, doi:10.1007/s00500-014-1388-4

[22] Rosyid, A.; El-Khasawneh, B.; Alazzam, A. (2018). Genetic and hybrid algorithms for optimization of non-singular 3PRR planar parallel kinematics mechanism for machining application, *Robotica*, Vol. 36, No. 6, 839-864, doi:10.1017/S0263574718000152

[23] Wang, L.; Tang, D.-B. (2011). An improved adaptive genetic algorithm based on hormone modulation mechanism for job-shop scheduling problem, *Expert Systems with Applications*, in press, doi:10.1016/j.eswa.2011.11.051

[24] Sun, G. X.; Bin, S. (2017). Router-level internet topology evolution model based on multi-subnet composited complex network model, *Journal of Internet Technology*, Vol. 18, No. 6, 1275-1283, doi:10.6138/JIT.2017.18.6.20140617

[25] Fang, J.; Xi, Y. (1997). A rolling horizon job shop rescheduling strategy in the dynamic environment, *The International Journal of Advanced Manufacturing Technology*, Vol. 13, No. 3, 227-232, doi:10.1007/BF01305874

[26] Zuo, Y.; Xue, A. (2013). A iterative decomposition procedure with global performance for job shop scheduling problems, *IFAC Proceedings Volumes*, Vol. 46, No. 13, 456-461, doi:10.3182/20130708-3-CN-2036.00104

[27] Chu, X.; Zhong, Q.-Y.; Khokhar, S. G. (2015). Triage scheduling optimization for mass casualty and disaster response, *Asia-Pacific Journal of Operational Research*, Vol. 32, No. 6, Paper 1550041, 20 pages, doi:10.1142/S0217595915500414

[28] Gao, K. Z.; Suganthan, P. N.; Pan, Q. K.; Tasgetiren, M. F. (2015). An effective discrete harmony search algorithm for flexible job shop scheduling problem with fuzzy processing time,

International Journal of Production Research, Vol. 53, No. 19, 5896-5911, doi:10.1080/00207543.2015.1020174

[29] Shahrabi, J.; Adibi, M. A.; Mahootchi, M. (2017). A reinforcement learning approach to parameter estimation in dynamic job shop scheduling, *Computers & Industrial Engineering*, Vol. 110, 75-82, doi:10.1016/j.cie.2017.05.026

[30] Wang, H.; Jiang, Z.; Wang, Y.; Zhang, H.; Wang, Y. (2018). A two-stage optimization method for energy-saving flexible job-shop scheduling based on energy dynamic characterization, *Journal of Cleaner Production*, Vol. 188, 575-588, doi:10.1016/j.jclepro.2018.03.254

[31] Liu, Y.; Yang, H.; Sun, G. X.; Bin, S. (2020). Collaborative filtering recommendation algorithm based on multi-relationship social network, *Ingénierie des Systèmes d'Information*, Vol. 25, No. 3, 359-364, doi:10.18280/isi.250310