Int j simul model 19 (2020) 1, 123-133
ISSN 1726-4529 Original scientific paper

A MULTI-OBJECTIVE FLEXIBLE JOB-SHOP SCHEDULING MODEL BASED ON FUZZY THEORY AND IMMUNE GENETIC ALGORITHM

Shi, D. L.[#]; Zhang, B. B. & Li, Y.
Shandong Agriculture and Engineering University, Jinan 250100, China
E-Mail: z2013379@sdaeu.edu.cn ([#] Corresponding author)

Abstract
This paper explores flexible job-shop scheduling problem (FJSP), using the rolling window rescheduling strategy. The fuzzy delivery time was considered, which satisfies the trapezoidal delivery window of the fuzzy membership function and directly bears on consumer satisfaction. Taking machine failure as the cause of dynamic interferences, the author established a dynamic scheduling model for the FJSP with fuzzy delivery time, according to the fuzzy mathematics theory. The model has multiple objectives: minimizing energy consumption, maximum makespan and consumer dissatisfaction. Next, the immune genetic algorithm (IGA) was improved to solve the model. The established model and the improved IGA were verified through simulations, in comparison with the genetic algorithm (GA). The research results shed new light on the FJSPs in real-world scenarios.
(Received in September 2019, accepted in January 2020. This paper was with the authors 1 month for 1 revision.)

Key Words: Flexible Job-Shop Scheduling Problem (FJSP), Fuzzy Delivery Time, Immune Genetic Algorithm (IGA), Makespan

1. INTRODUCTION

The job-shop scheduling problem (JSP) aims to optimize relevant performance indices through reasonable allocation of production tasks under limited resources and certain rules. Let n and m be the number of jobs in the job-shop and the number of machines to process the jobs. Suppose each job needs to go through a number of operations, and each operation must be executed by one machine. Then, the JSP can be described as the optimization of performance indices by arranging the operation sequence of each job on the machines.

The JSP varies with the conditions of the job-shop. In general, the JSPs can be divided into single-machine JSP [1], parallel-machine JSP [2], open-shop scheduling problem (OSSP) [3], flow-shop scheduling problem (FSSP) [4] and flexible job-shop scheduling problem (FJSP) [5-7]. The details of each type of JSP are given in Table I below.

Table I: Classification of JSPs.

Type	Description
Single-machine JSP	The job-shop has only one machine to process all jobs, and each job only needs to go through one operation.
Parallel-machine JSP	The job-shop has several machines to process the jobs, and each job only needs to go through one operation.
OSSP	Each job needs to go through multiple operations, with no constraint on operation sequence (i.e. the operations can be implemented in random order).
FSSP	The job-shop has several machines with different functions, and each job needs to go through multiple operations. Each operation must be implemented on a designated machine. The jobs have the same operation sequence.
FJSP	Each operation can be implemented on multiple machines.

The FJSP is the most complex form of the JSP. Traditionally, the FJSP is treated as a deterministic problem, in which each task has a fixed processing time and delivery time. In real-world scenarios, however, both processing time and delivery time are subjected to random interferences, which arise from factors like sudden machine failure and the change of production environment. Therefore, the FJSP is actually a dynamic problem with a fuzzy delivery time.

2. LITERATURE REVIEW

Over the years, many scholars have probed deep into the dynamic JSPs. Nelson et al. [8], being the first to solve dynamic JSPs, divided the dynamic process into continuous static intervals, and handled the problem by rolling window. Considering the dynamicity of machine failure and processing time, Sabuncuoglu and Karabuk [9] proposed a rescheduling strategy for job-shop with variable intervals. Chou [10] solved the single-machine dynamic JSP with machine learning and genetic algorithm (GA). Based on multi-objective evolutionary algorithm, Shen and Yao [11] developed a dynamic scheduling optimization model, which integrates operation scheduling and machine allocation, and introduced heuristic strategy into the model to quickly adapt to new environment. From the perspective of dynamic disturbances, Kundakci and Kulak [12] constructed a model with maximal makespan, and solved the model by hybrid GA. Inspired by rule-based scheduling, Zhu [13] selected the jobs with high rule probabilities, improved the non-dominated sorting genetic algorithm (NSGA), and verified the rationality of the algorithm in the dynamic optimization of multi-objective FJSP. Wang et al. [14] proposed a dynamic rescheduling strategy with variable intervals, set up a dynamic scheduling model, and designed an improved GA to solve the model.

In recent years, fuzzy mathematics has permeated into the field of the JSP. For example, Ishii et al. [15] were the first to consider fuzzy delivery time in the OSSPs with a single machine or two machines. Sakawa and Kubota [16] studied the multi-objective JSP from the angle of fuzzy delivery time. Liu et al. [17] relied on distributed estimation to solve the dynamic scheduling of FJSP with fuzzy processing time, and converted the dynamic fuzzy FJSP into a static fuzzy FJSP through a series of changing programs. Gao et al. [18] verified the feasibility of two-stage artificial bee colony (ABC) algorithm in solving the FJSP, through the analysis on eight cases. Palacios et al. [19] investigated the FJSP with fuzzy processing time, created a model to minimize the maximum makespan, and combined tabu search and heuristic strategy into a GA to solve the model. Chiang and Lin [20] proposed a FJSP model with fuzzy processing time, and solved it with multi-objective evolutionary algorithm.

To sum up, the FJSP and fuzzy FJSP have both been explored extensively, bringing great progress in model construction and algorithm design. However, there are still some shortcomings in the current research. Firstly, most scholars treated delivery time as fixed rather than fuzzy, while the delivery time in actual FJSPs is not fixed. Secondly, the dynamic interferences, which suppress the production efficiency, in actual FJSPs have not been fully considered. This calls for efficient rescheduling strategies and algorithms to solve the dynamic FJSP.

3. MULTI-OBJECTIVE FJSP MODEL WITH FUZZY DELIVERY TIME

3.1 Problem description and hypotheses

The dynamic FJSP aims to process n jobs $\{J_1, J_2, \ldots, J_n\}$ on m machines $\{M_1, M_2, \ldots, M_m\}$. Each job J_i ($i = 1, 2, \ldots, n$) needs to go through o_j operations. Each operation can be implemented on one of the machines, and the processing time of an operation varies with

machines. Let O_{ij} be the j^{th} operation of the i^{th} job J_i, and M_{ij} ($M_{ij} \subseteq M$) be the set of machines available for operation O_{ij} to choose from. If machine k is selected for operation O_{ij}, then the processing time of operation O_{ij} can be expressed as T_{ijk}.

During the actual production, dynamic interferences may occur to the FJSP, making the initial scheduling strategy infeasible. The strategy must be modified to cope with these interferences. In this paper, machine failure is regarded as the cause of the dynamic interferences. Considering the impacts of dynamic interferences and fuzzy processing time, the author employed the rolling window to select the most suitable machine for each operation, and rationalize the operation sequence on each machine, aiming to achieve the optimal rescheduling effect.

In traditional FJSPs, the production environment is greatly simplified. For convenience, the static FJSPs often assume that no operation of a job is interruptible. This assumption overlooks the dynamic interferences. By contrast, the dynamic FJSP is much closer to actual production. Take machine failure for example. Once a machine fails, the job processing will be suspended. Then, the subsequent operations on the machine will be halted, making the original scheduling strategy infeasible. To restore the normal operation, it is necessary to reschedule the job processing.

Based on the above description, the following hypotheses were presented to make the dynamic FJSP more realistic:

- Hypothesis 1: A machine can only process one job at a time.
- Hypothesis 2: The machine sequence is fixed for each job, and no job is prior to the other jobs.
- Hypothesis 3: The job preparation time is included in the processing time.
- Hypothesis 4: The operation sequence is fixed for each job, i.e. the subsequent operation cannot be started before the completion of the current operation.
- Hypothesis 5: During rescheduling, the operation not in the current window cannot be started until the current operation is completed.
- Hypothesis 6: The operation of a job being implemented on the faulty machine is included in the current window, and will be restarted after the rescheduling.

3.2 Model construction

The solution to the FJSP must strike a balance between the interests and requirements of all stakeholders, namely, efficiency, cost control and timely delivery. In this paper, multiple objectives are considered to solve the dynamic FJSP. First, the maximum makespan must be minimized, which is a common objective of scheduling optimization. Next, the energy consumption of machines should be controlled, according to the actual production environment. In addition, the fuzzy delivery time ought to be considered, which satisfies the trapezoidal delivery window of the fuzzy membership function and directly bears on consumer satisfaction. To sum up, this paper attempts to establish an optimization model with such objectives as minimizing the maximum makespan, energy consumption and consumer dissatisfaction.

(1) Minimizing the maximum makespan

The maximum makespan refers to the longest time it takes to complete implementing all the operations of all the jobs. This parameter reflects the production efficiency of the job-shop. The maximum makespan can be defined as:

$$C_{max} = \max\left(\sum_{i=1}^{n} C_i\right) \tag{1}$$

where, C_i is the processing time of each job J_i.

(2) Minimizing energy consumption

There are four states of each machine: stop, start, no-load operation and processing. The machine consumes different amounts of energy in different states. Empirical evidence shows that a machine mainly consumes energy in processing state, no-load state and start state.

1) Energy consumption in processing state

In processing state, the energy consumed by a machine depends on two factors: the energy consumption in processing state per unit time, and the processing time. Hence, the energy consumption of all machines in processing state can be expressed as:

$$\sum E_a = \sum_{i=1}^{n} \sum_{j=1}^{h_i} \sum_{k=1}^{m} E_a^{ijk} \times P_{ijk} \tag{2}$$

where, E_a^{ijk} is the energy consumed by machine k to process job J_i per unit time; P_{ijk} is the processing time of job J_i on machine k.

2) Energy consumption in no-load state

In no-load state, the energy consumed by a machine depends on two factors: the energy consumption in no-load state per unit time, and the dead time. Hence, the energy consumption of all machines in no-load state can be expressed as:

$$\sum E_b = \sum_{k=1}^{m} E_b^k \times t_{bk} \tag{3}$$

where, E_b^k is the energy consumed by machine k in no-load state per unit time; t_{bk} is the dead time of machine k.

3) Energy consumption in start state

In start state, the energy consumed by a machine depends on two factors: the energy consumption in start state per unit time, and the start time. Hence, the energy consumption of all machines in start state can be expressed as:

$$\sum E_c = \sum_{k=1}^{m} E_c^k \times t_{ck} \tag{4}$$

where, E_c^k is the energy consumed by machine k in start state per unit time; t_{ck} is the single start time of machine k.

To sum up, the total energy consumption of all machines can be obtained by summing up the energy consumptions in the above three states:

$$E = \sum E_a + \sum E_b + \sum E_c \tag{5}$$

To minimize the energy consumption of dynamic FJSP, it is necessary to select the most suitable machine for each operation of each job and minimize the number of machines in idle period (i.e. no-load state and stop state). The idle period is inevitable in job processing. If the length of the idle period T is no longer than $E_c^k \times t_{ck}/E_b^k$, the machine is in no-load state; otherwise, the machine is in stop state.

(3) Minimizing consumer dissatisfaction

In the dynamic FJSP, both consumers and the manufacturer long for timely delivery. Early completion marks up the storage cost and incurs a penalty. The manufacturer also faces a penalty for delivery delay.

In the JSPs, the traditional practice is to set the delivery time to a fixed value. However, a fixed delivery time is impossible in actual job-shops, due to information and technical constraints. Therefore, it is more realistic to treat the time factors of dynamic FJSP as fuzzy variables.

Let c_i and d_i be the processing time and delivery time of job J_i, respectively. If $c_i > d_i$, the early completion occurs; otherwise, the delivery delay happens:

$$L_i = c_i - d_i \tag{6}$$

$$T_i = \max\{L_i, 0\} \tag{7}$$

(4) Multi-objective optimization model

Through the above analysis, three optimization objectives, namely, minimizing maximum makespan, energy consumption and consumer dissatisfaction, were selected based on the common objectives of JSPs and the actual production environment. Then, the multi-objective optimization model was established by lexicographic multi-objective programming (LMOP) [21].

The LMOP sorts multiple objectives by importance, and considers them in descending order of importance. The next objective will be solved if the optimal solution of the current objective in not unique, until there is a unique solution for every objective.

In this way, the most, second and third important objectives of our problem were determined as minimizing energy consumption, O_1, minimizing consumer dissatisfaction, O_2, and minimizing maximum makespan, O_3. Hence, the multi-objective optimization model for the dynamic FJSP can be constructed as:

$$\min E = \left(\sum_{i=1}^{n} \sum_{j=1}^{h_i} \sum_{k=1}^{m} E_a^{ijk} \times P_{ijk} + \sum_{k=1}^{m} E_b^k \times t_{bk} + \sum_{k=1}^{m} E_c^k \times t_{ck} \right) \tag{8}$$

$$\max \ F = \max \left[\sum_{i=1}^{n} \alpha(C_i) \right] \tag{9}$$

$$\min \ C_{max} = \min \left[max \sum_{i=1}^{n} C_i \right] \tag{10}$$

$$\text{s.t.} \ \ O_1 \gg O_2 \gg O_3 \tag{11}$$

$$U_{ijk} + Y_{ijk} \times W_{ijk} \leq T_{ijk} \tag{12}$$

$$T_{ijk} \leq U_{i(j+)k} \tag{13}$$

$$C_i \leq C_{max} \tag{14}$$

$$U_{ijk} + W_{ijk} \leq U_{abk} + L \times \left(1 - R_{ijabk}\right) \tag{15}$$

$$\sum_{k=1}^{m_{ij}} Y_{ijk} = 1 \tag{16}$$

$$U_{ijk} \geq 0 \, , T_{ijk} \geq 0 \tag{17}$$

where, W_{ijk} and U_{ijk} are the processing time and preparation time of job J_i on machine k, respectively. W_{ijk} is a decision variable: if $W_{ijk} = 1$, operation O_{ij} is implemented on machine k; if $W_{ijk} = 0$, operation O_{ij} is not implemented on machine k. R_{ijabk} is also a decision variable: if $R_{ijabk} = 1$, operation O_{ij} is implemented before operation O_{ab} on machine k; if $R_{ijabk} = 0$, operation O_{ij} is implemented after operation O_{ab} on machine k.

Eq. (8) represents the objective to minimize energy consumption, including the energy consumed in processing state, no-load state and start state; Eq. (9) represents the objective to

minimize consumer dissatisfaction; Eq. (10) represents the objective to minimize the maximum makespan; Eq. (11) specifies the priority of each objective; Eqs. (12) and (13) maintain the operation sequence of each job, i.e. the next operation cannot be implemented before the current operation of the job is completed; Eq. (14) requires that the makespan of a job cannot exceed the maximum makespan; Eq. (15) indicates that the same machine can only execute one operation at a time; Eq. (16) shows the same operation can only be implemented on one machine; Eq. (17) is the nonnegative constraint of variables.

4. IMPROVED IMMUNE GENETIC ALGORITHM

4.1 Immune algorithm

Inspired by the immune system, the immune algorithm (IA) provides a desirable tool for combinatorial optimization problems, thanks to its stochasticity and fuzziness. The IA mainly performs three functions: antigen recognition, antibody evaluation and immunization.

The recognition of antigen is achieved by generating an initial population of antibodies. If relevant memories are available in the memory cells, the antibodies will be generated by these cells; otherwise, the antibodies will be initialized in a random manner. The antibodies are then evaluated through individual selection based on fitness. The immunization is carried out in the light of the immune operator.

Mimicking the immune system, the IA can converge to the optimal solution quickly. The similarity between individuals is adjusted to maintain the diversity of the population, and avoid the local optimum trap.

As shown in Fig. 1, the IA mainly covers seven steps: antigen recognition, population initialization, affinity calculation, differentiation between memory cells, promotion/inhibition of antibody growth, population update, and termination judgement [22].

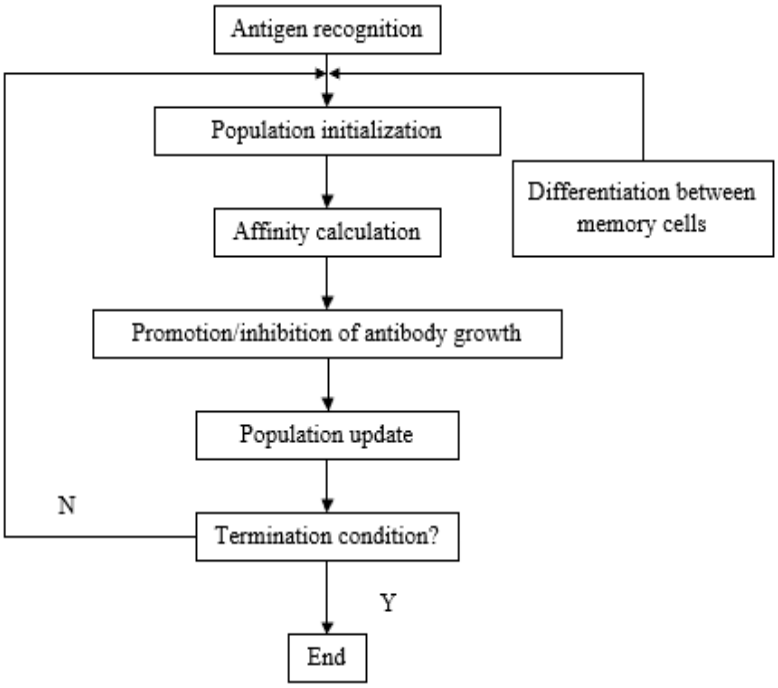

Figure 1: Flow chart of the IA.

4.2 Immune genetic algorithm

Immune genetic algorithm (IGA) is a combination between the GA and the IA, which have similarities and differences in solving JSPs.

The IA is introduced to make the random iterative search of the GA more targeted, especially in the extraction of vaccines. In the IGA, the immune process produces antibodies to foreign antigens, and converts the memory of the antibody to the antigen into a vaccine operator to be optimized. In this way, the IGA can converge quickly to the optimal solution, without falling to the local optimum.

The key step of the IGA lies in vaccination and antibody selection, which overcomes the defects of the GA, such as the randomness of crossover and mutation and the degradation of individuals.

4.3 Improved immune genetic algorithm

This paper further improves the IGA for the multi-objective optimization of dynamic FJSP, including adaptively adjusting the crossover and mutation probabilities, and initializing the population of machines with relatively high efficiency. As shown in Fig. 2, the improved IGA (IIGA) is implemented in the following steps:

Step 1. Coding

This step aims to sort the operations and allocate the machines. Hence, a two-layer coding method was adopted. Each operation was encoded by the serial number of the corresponding job and machine. The operation sequence of a job was taken as a chromosome. Each machine is encoded by its serial number.

Step 2. Antigen/antibody representation

For the dynamic FJSP, the objective functions and constraints were expressed as antigens, while the scheduling solutions as antibodies.

Step 3. Population initialization

To initialize the antibody population, the machines were selected in descending order of efficiency, i.e. in ascending order of processing time. Firstly, the processing time of each operation was obtained based on the schedule and minimum makespan T_{min}. Then, the probability of selecting a machine for that operation was computed by:

$$p = \frac{\frac{1}{T_{mini}}}{\sum_{i}^{apq} \frac{1}{T_{mini}}} \tag{18}$$

The most probable machine was then selected for that operation. The above procedure was repeated until all machines were encoded.

Step 4. Crossover and mutation

The crossover was implemented using the POX operator. Firstly, the jobs were randomly divided into two sets, A and B. Then, a chromosome of zeros and ones was generated. The jobs in the two sets corresponding to the positions of ones were kept unchanged, and those corresponding to the positions of zeros were exchanged.

The mutation was carried out using the swap operator. Two chromosomes were randomly created. The genes in each chromosome were swapped based on machine variation.

Step 5. Calculation of affinity and adaptive parameters

The affinity between antibody and antigen was calculated by high-frequency mutations, and expressed as information entropy. To ensure population diversity, the antibodies with low concentration and affinity were retained.

The adaptive parameters were calculated to adjust the crossover and mutation probabilities, thereby enhancing the solution quality and speeding up global convergence.

Step 6. Vaccine extraction and immune memorization

Vaccine extraction was performed to eliminate the antibodies with low affinity, leaving only those with high affinity.

To facilitate memorization, the selected antibodies were converted into effector cells and cloned. Based on affinity, the memory cells are adjusted both actively and passively. Active adjustment is to introduce new cells to replace the weak ones, and passive adjustment is to convert high-affinity effector cells into memory cells, and use them replace the memory cells with the lowest affinity.

Step 7. Vaccination and antibody selection

The extracted antibodies were added into the population, enhancing the productivity. To obtain a feasible solution and ensure population diversity, the sequential replacement was conducted to inoculate each allele: (1) Find allele i of vaccine $k_i \neq 0$ one by one; (2) Find k_i from the next position; (3) Swap the allelic values at the two positions. Next, the antibodies with high affinity and low concentration were added to the new population.

Step 8. Population update

The antibody population was updated by elite retention. If the population is sufficiently large, the new population would be randomly generated.

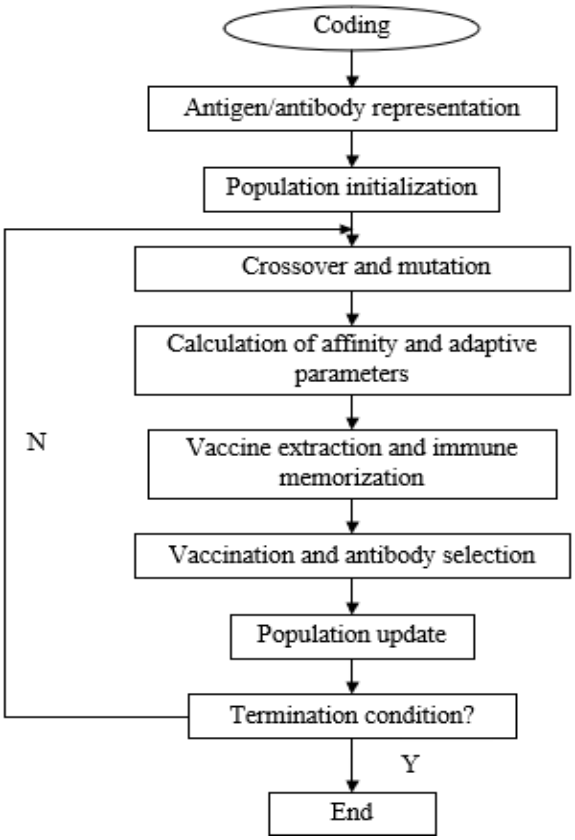

Figure 2: Flow chart of the IIGA.

5. SIMULATION AND RESULTS ANALYSIS

To verify its performance, the IIGA was applied to the multi-objective optimization of a classical example of dynamic FJSP with fuzzy delivery time. As shown in Table II, the example FJSP involves 5 jobs and 10 machines. Each job has 5 operations. For each job, two machines could be chosen from for each of its operation.

The main parameters of the IIGA were configured as follows: population size, $N = 90$; maximum number of iterations, $MI = 200$; crossover probability, $k = 1.2$; mutation probability, $m = 0.2$; memory capacity, $M = 20$; clone coefficient, $\delta = 0.65$.

It is assumed that machine 2 failed at time zero, and the unprocessed job on that machine was transferred to the other machine available for that job. The rescheduling was conducted at time 20. Then, the Gantt chart after rescheduling is shown in Fig. 3.

Table II: An example of the FJSP.

	Jobs	**1**	**2**	**3**	**4**	**5**
Machine constraint	Operation 1	(2, 10)	(1, 8)	(2, 8)	(3, 6)	(4, 3)
	Operation 2	(1, 6)	(3, 2)	(4, 7)	(2, 9)	(3, 8)
	Operation 3	(3, 7)	(5, 9)	(6, 10)	(2, 8)	(5, 8)
	Operation 4	(5, 7)	(2, 7)	(1, 10)	(4, 6)	(5, 7)
	Operation 5	(3, 8)	(2, 10)	(4, 6)	(2, 6)	(2, 6)
Processing time	Operation 1	(3, 6)	(2, 4)	(3, 6)	(6, 7)	(3, 5)
	Operation 2	(10, 8)	(1, 7)	(4, 5)	(9, 11)	(10, 12)
	Operation 3	(7, 3)	(1, 3)	(5, 7)	(6, 9)	(9, 6)
	Operation 4	(2, 2)	(6, 4)	(8, 1)	(2, 2)	(5, 3)
	Operation 5	(10, 3)	(2, 2)	(1, 8)	(2, 6)	(9, 7)

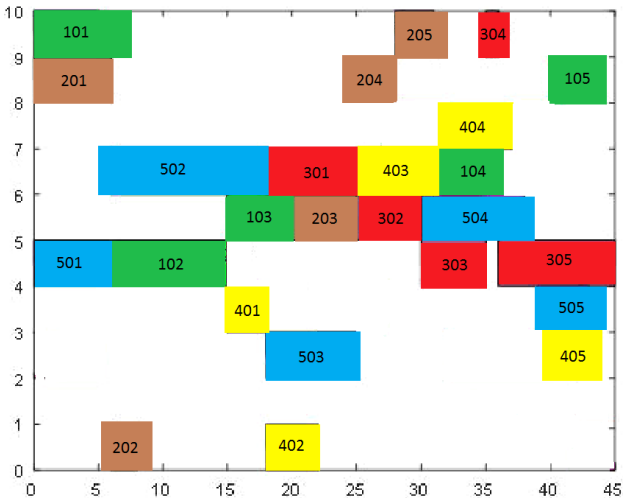

Figure 3: The Gantt chart after rescheduling.

As shown in Fig. 3, no job was processed on the other machine after rescheduling. The maximum makespan, energy consumption and consumer satisfaction were 45, 466.15 and 4.22, respectively.

The IIGA was adopted to simulate the rescheduling of 10 fault machines. Each fault machine was simulated 20 times. The mean results of the three objectives are listed in Table III.

Table III: Mean test results of the three objectives.

Machine no.	**Energy consumption**	**Customer satisfaction**	**Maximum makespan**
1	452.16	4.19	45
2	463.70	4.22	45
3	446.32	5.52	39
4	498.02	4.27	45
5	478.91	3.97	50
6	501.24	4.86	45
7	449.28	5.89	40
8	457.93	5.71	40
9	482.94	4.32	45
10	503.99	5.58	39

As shown in Table III, the energy consumption, customer satisfaction and maximum makespan varied with the fault machines. Specifically, the energy consumption was relatively high when machines 4, 5, 6 and 9 failed, indicating that these machines consume lots of energy in job processing. The consumers were fairly satisfied when machines 3, 7, 8 and 10 failed, indicating that the failures are tolerated by consumers. The maximum makespan was relatively short when machines 3, 7, 8 and 10 failed, indicating that these machines have high efficiency.

Overall, energy consumption, consumer dissatisfaction and maximum makespan were all minimized, when machines 3, 7, 8 and 10 failed. Hence, these four machines are more important than the remaining six machines. In FJSP, the failure of these four machines should be avoided.

Under the failure of machine 2, the GA and the IIGA were both applied to solve the FJSP with fuzzy delivery time. The optimal solutions of both algorithms (Table IV) were obtained after ten repeated solving processes.

Table IV: Comparison of optimal solutions.

Objectives	IIGA	GA
Energy consumption	446.32	497.65
Customer satisfaction	5.79	5.52
Maximum makespan	41	45

From Table IV, it can be seen that the IIGA outperformed the GA in optimization ability. The optimal solutions of the IIGA were better than those of the GA.

6. CONCLUSIONS

Considering energy consumption, this paper mainly explores the dynamic optimization of the FJSP with fuzzy delivery time. The author established a dynamic optimization model for the problem with three objectives: minimizing the maximum makespan, energy consumption and consumer dissatisfaction, and improved the IGA to solve the model. Finally, the model and algorithm were proved valid through simulations.

REFERENCES

[1] Mauguière, P.; Billaut, J.-C.; Bouquard, J.-L. (2005). New single machine and job-shop scheduling problems with availability constraints, *Journal of Scheduling*, Vol. 8, No. 3, 211-231, doi:10.1007/s10951-005-6812-2

[2] Liu, S. Q.; Kozan, E. (2012). A hybrid shifting bottleneck procedure algorithm for the parallel-machine job-shop scheduling problem, *Journal of the Operational Research Society*, Vol. 63, No. 2, 168-182, doi:10.1057/jors.2011.4

[3] Koshimura, M.; Nabeshima, H.; Fujita, H.; Hasegawa, R. (2010). Solving open job-shop scheduling problems by SAT encoding, *IEICE Transactions on Information and Systems*, Vol. 93, No. 8, 2316-2318, doi:10.1587/transinf.E93.D.2316

[4] Engin, O.; Doyen, A. (2004). A new approach to solve hybrid flow shop scheduling problems by artificial immune system, *Future Generation Computer Systems*, Vol. 20, No. 6, 1083-1095, doi:10.1016/j.future.2004.03.014

[5] Mu, H. (2019). Disruption management of flexible job shop scheduling considering behavior perception and machine fault based on improved NSGA-II algorithm, *Journal Européen des Systèmes Automatisés*, Vol. 52, No. 2, 149-156, doi:10.18280/jesa.520206

[6] Ding, H. (2018). Identification of job-shop bottlenecks based on network structure features and simulation of resource allocation optimization under disturbances, *Academic Journal of Manufacturing Engineering*, Vol. 16, No. 2, 37-45

[7] Gocken, T.; Dosdogru, A. T.; Boru, A.; Gocken, M. (2019). Integrating process plan and part routing using optimization via simulation approach, *International Journal of Simulation Modelling*, Vol. 18, No. 2, 254-266, doi:10.2507/IJSIMM18(2)470

[8] Nelson, R. T.; Holloway, C. A.; Wong, R. M.-L. (1977). Centralized scheduling and priority implementation heuristics for a dynamic job shop model, *AIIE Transactions*, Vol. 9, No. 1, 95-102, doi:10.1080/05695557708975127

[9] Sabuncuoglu, I.; Karabuk, S. (1999). Rescheduling frequency in an FMS with uncertain processing times and unreliable machines, *Journal of Manufacturing Systems*, Vol. 18, No. 4, 268-283, doi:10.1016/S0278-6125(00)86630-3

[10] Chou, F.-D. (2009). An experienced learning genetic algorithm to solve the single machine total weighted tardiness scheduling problem, *Expert Systems with Applications*, Vol. 36, No. 2, 3857-3865, doi:10.1016/j.eswa.2008.02.040

[11] Shen, X.-N.; Yao, X. (2015). Mathematical modeling and multi-objective evolutionary algorithms applied to dynamic flexible job shop scheduling problems, *Information Sciences*, Vol. 298, 198-224, doi:10.1016/j.ins.2014.11.036

[12] Kundakci, N.; Kulak, O. (2016). Hybrid genetic algorithms for minimizing makespan in dynamic job shop scheduling problem, *Computers & Industrial Engineering*, Vol. 96, 31-51, doi:10.1016/j.cie.2016.03.011

[13] Zhu, W. (2017). Multi-objective dynamic scheduling algorithm for flexible job-shop problem based on rule orientation, *Systems Engineering – Theory & Practice*, Vol. 37, No. 10, 2690-2699, doi:10.12011/1000-6788(2017)10-2690-10

[14] Wang, W.-L.; Wang, L.; Wang, H.-Y.; Xu, X.-L.; Zhao, Y.-W. (2012). Dynamic job shop scheduling based on hybrid differential evolution algorithm, C*omputer Integrated Manufacturing Systems*, Vol. 18, No. 3, 531-539

[15] Ishii, H.; Tada, M.; Masuda, T. (1992). Two scheduling problems with fuzzy due-dates, *Fuzzy Sets and Systems*, Vol. 46, No. 3, 339-347, doi:10.1016/0165-0114(92)90372-b

[16] Sakawa, M.; Kubota, R. (2000). Fuzzy programming for multiobjective job shop scheduling with fuzzy processing time and fuzzy duedate through genetic algorithms, *European Journal of Operational Research*, Vol. 120, No. 2, 393-407, doi:10.1016/s0377-2217(99)00094-6

[17] Liu, B.; Fan, Y.; Liu, Y. (2015). A fast estimation of distribution algorithm for dynamic fuzzy flexible job-shop scheduling problem, *Computers & Industrial Engineering*, Vol. 87, 193-201, doi:10.1016/J.CIE.2015.04.029

[18] Gao, K.-Z.; Suganthan, P. N.; Pan, Q. K.; Tasgetiren, M. F.; Sadollah, A. (2016). Artificial bee colony algorithm for scheduling and rescheduling fuzzy flexible job shop problem with new job insertion, *Knowledge-Based Systems*, Vol. 109, 1-16, doi:10.1016/j.knosys.2016.06.014

[19] Palacios, J. J.; González, M. A.; Vela, C. R.; González-Rodríguez, I.; Puente, J. (2015). Genetic tabu search for the fuzzy flexible job shop problem, *Computers & Operations Research*, Vol. 54, 74-89, doi:10.1016/j.cor.2014.08.023

[20] Chiang, T.-C.; Lin, H.-J. (2013). A simple and effective evolutionary algorithm for multiobjective flexible job shop scheduling, *International Journal of Production Economics*, Vol. 141, No. 1, 87-98, doi:10.1016/j.ijpe.2012.03.034

[21] Sun, G.; 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

[22] Sun, G.; Bin, S. (2017). A new opinion leaders detecting algorithm in multi-relationship online social networks, *Multimedia Tools and Applications*, Vol. 77, No. 4, 4295-4307, doi:10.1007/s11042-017-4766-y