Int j simul model 18 (2019) 3, 510-520
ISSN 1726-4529 Original scientific paper

A MULTI-OBJECTIVE OPTIMIZATION MODEL BASED ON NON-DOMINATED SORTING GENETIC ALGORITHM

Fu, H. C.[*,#] & Liu, P.[*,**]
[*] Hubei Jinhua Real Estate Co., Ltd., Wuhan 430040, China
[**] China Construction International Co., Ltd., Central China Group, Wuhan 430040, China
E-Mail: 22374591@qq.com ([#] Corresponding author)

Abstract
This paper attempts to solve the job-shop scheduling problem (JSP), in which machines are shared among multiple tasks. For this purpose, a multi-objective optimization model was established to minimize the total completion time and total cost. To solve the model, a scheduling strategy was proposed based on the NGSA with crowding mechanism. Compared with the GA, the improved NGSA can effectively avoid the local optimum trap and maintain population diversity in the later stage. In addition, the heuristic crossover operator was introduced to enhance the local search ability of the improved NGSA. The effectiveness of the proposed scheduling strategy was proved valid through simulation.
(Received, processed and accepted by the Chinese Representative Office.)

Key Words: Job-Shop Scheduling Problem (JSP), Genetic Algorithm (GA), Non-Dominated Sorting Genetic Algorithm (NSGA), Multi-Objective Scheduling

1. INTRODUCTION

Machine sharing refers to the unified scheduling of all machines in the job-shop. When several machines are reserved for new tasks, each incoming task should be assigned to a suitable machine in the light of its demand. With the rapid growth in the number of tasks, it is inevitable that multiple tasks issue jobs at the same time, waiting to be processed on the reserved machines. Sometimes, the type of machines suitable to process these jobs may not be available. This calls for the optimization of the job sequence based on the task demands (e.g. cost and time).

This kind of job-shop scheduling problem (JSP), in which machines are shared among multiple tasks, requires the arrangement of suitable machines for a group of jobs, each of which needs to go through one or more processes, as per task demands. The scheduling strategy is affected by multiple factors, namely, the job sequence, machine capacity, number of jobs, delivery time and cost limit. These can be considered as deterministic factors in the JSP. Meanwhile, abnormalities like job change, system failure and machine failure are random factors.

The scheduling performance can be evaluated by various indices. These evaluation indices generally fall into three categories: capacity indices, cost indices and user satisfaction indices. The capacity indices mainly include the job completion time and product qualification rate of the machines. The cost indices focus on the production cost and profit. The user satisfaction indices describe how much the parameters meet user expectations at the end of production. The scheduling objectives may vary depending on the perspective. For task issuer, the machines should operate at the lowest cost and process jobs in the shortest time. For machine owner, the utilization and load of all machines should be improved and balanced.

The common way to solve the said JSP is to transform the problem into a constrained optimization problem, establish a mathematical model, and select a suitable algorithm to determine the optimal scheduling strategy. Based on the non-dominated sorting genetic algorithm (NSGA), this paper designs a multi-objective optimization

algorithm to solve the JSP, in which machines are shared among multiple tasks. The algorithm can optimize multiple objectives simultaneously, such that each objective is optimized as far as possible. In this way, a set of non-dominated solutions were obtained. Then, the optimal solution was selected according to the demand of specific tasks.

2. LITERATURE REVIEW

In traditional JSP, the objective is to minimize the completion time by optimizing the operation sequence of each job [1]. Along with the stricter task requirements, the flexible job-shop scheduling problem (FJSP) has emerged to strike a balance between production cost and profit. In the FJSP, each operation of a job can be implemented on multiple machines, leading to a larger solution space and greater randomness [2]. To solve the problem, the scheduler must determine the operation sequence of each job and the machine sequence of each job.

In the real world, the JSP is a combinatorial optimization problem of machines and jobs. The scheduler must at once fulfil the expectation of different tasks, and satisfy the constraints on job sequence and completion time. In theory, the problem is non-deterministic polynomial-time (NP) hard. Many attempts have been made to solve the JSP in a satisfactory manner. For instance, Wong and Ngan [3] constructed a JSP model for minimal completion time, and proposed a hybrid particle swarm optimization (PSO) under uncertain completion time. Li et al. [4] introduced some priority rules into genetic algorithm (GA), and re-sorted the jobs being processed on the same machine. In this way, the scheduling effect was improved, realizing reduced time gap and limited exchange. Banihani et al. [5] improved the GA to optimize the job sequence in flow shop scheduling problem (FSP) under time window constraints. Li and Wu [6] proposed a quantitative optimization model for the JSP, in which the machines are shared among concurrent tasks, and presented a heuristic algorithm based on problem decomposition. Grillo et al. [7] constructed a dynamic real-time decision-making model based on task priority, and created a shared reservation model of Markov objective function, which can be solved by dynamic programming. Akhshabi et al. [8] optimized the original set of jobs under the constraints of delivery time and quantity, thus reducing the power consumption and scheduling time of the job-shop.

The GA has been widely used in scheduling problems, thanks to its strong global search ability and implicit parallelism. For example, Zhang et al. [9] designed a priority-preserving crossover operator and a domain search mutation operator to solve the JSP, and proposed a hybrid GA with excellent global and local search abilities. Asefi et al. [10] created a process-based coding method for the GA, which expands the search space of the process domain in three steps: improving the quality of the initial population based on rich heuristic information, changing the job sequence on each machine through standardized forward coding, and adjusting the result of active decoding to the right. To improve product quality and shorten makespan, Geng et al. [11] optimized the machine allocation in the JSP, and updated the population of the GA in each iteration under the simulated annealing (SA) mechanism. Zhang et al. [12] improved the crossover and mutation operators to speed up the search for the global optimum and prevent premature convergence.

Recent years has seen a growing interest in multi-objective evolutionary scheduling algorithm (MOEA) [13], which effectively pursues multiple optimization goals. For example, Yi et al. [14] introduced the elite retention strategy to the NSGA, designed genetic operators for operation adjustment and machine scheduling, and derived the optimal solution set by analytic hierarchy process (AHP). Rohaninejad et al. [15] suggested optimizing the intensity Pareto algorithm with fuzzy C-means clustering, rather than hierarchical clustering, aiming to coordinate scheduling problems with different machine utilizations. Luo et al. [16] established an FJSP model for the maximal completion time and machine load index. Focusing on multi-

objective JSP, Lei [17] merged the child population of genetic operation with the parent population, selected the next-generation individuals for the merged population by the NSGA-III, and introduced a heuristic priority mechanism to guide the simultaneous optimization of multiple objectives. Cai et al. [18] established a multi-objective FJSP optimization model that maximizes the completion time, machine load and the total machine load.

Drawing on the existing studies, this paper attempts to solve multi-objective JSP by the following strategy: acquiring a set of multiple non-dominated solutions by multi-objective optimization algorithm, setting up a priority sequence based on the weight or importance of each objective, and outputting the best solution as the final solution. For the same problem, the objectives were given different weights.

3. MULTI-OBJECTIVE OPTIMIZATION OF THE JSP

This section describes a multi-objective JSP, and then sets up a mathematical model to minimize the completion time and cost.

3.1 The JSP

Job-shop scheduling is to allocate each operation of a job to a suitable machine at the proper time. If implemented properly, the scheduling can reduce the makespan and enhance the machine utilization. The completion time and cost are the top concerns of the scheduler. Below is the mathematical description of the JSP [19].

Let $J\{J_i,\ i = 1, 2, \dots, n\}$ be a set of n jobs, and $\mathrm{M} = \{M_1, M_2, \dots, M_t\}$ be a set of M machines. Each job J_i consists of N_i operations. The j^{th} operation of job J_i, denoted as $O_{i,j}$, $j \in \{1, 2, \dots, N_i\}$, can be implemented on a set M_p of num_p machines. The M machines can be classified into t types by function.

Assuming that operation $O_{i,j}$ is implemented on the k^{th} ($k \in M_p$) machine, the completion time of the operation on that machine can be denoted as W_{ijk}, which is the difference between the start time S_{ijk} and end time E_{ijk}. The unit time cost of the k^{th} machine, and the waiting time to implement operation $O_{i,j}$ on the k^{th} machine are respectively denoted as C_{pk} and WT_k. The expected completion time and cost are UTE_i and UCE_i, respectively.

The JSP optimization model should determine the optimal job sequence on each machine under the given constraints, such as to minimize the completion time (including waiting time and running time) and cost of the jobs. Thus, the objective function and constraints can be expressed as:

$$Z = \begin{cases} f_1 = min \sum_{i=1}^{n} \sum_{j=1}^{N_i} \sum_{k \in M_p} [x_{ijk} \times (C_{pk} \times W_{ijk})] \\ f_2 = min \sum_{i=1}^{n} \sum_{j=1}^{N_i} \sum_{k \in M_p} [x_{ijk} \times (W_{ijk} + WT_k)] \end{cases} \tag{1}$$

s.t.

$$\sum_{i=1}^{n} \sum_{j=1}^{N_i} W_{ijk} \le H_{pk}$$

$$\sum_{j=1}^{N_i} (W_{ijk} + WT_k) \le UTE_i,\ \ i = 1,2, \dots n$$

$$S_{i(j-1)k} + W_{i(j-1)k} \ge S_{ijk},\ \ x_{ijk} = x_{i(j-1)k} = 1$$

where f_1 is the minimal cost of all jobs; f_2 is the minimal completion time of each job; x_{ijk} is a variable (if operation $O_{i,j}$ is implemented on the k^{th} machine, $x_{ijk} = 1$; otherwise, $x_{ijk} = 0$).

3.2 Multi-objective optimization model

Let $J = \{J_1, J_2, \ldots J_j\}$ be a set of j tasks and $M = \{M_1, M_2, \ldots, M_m\}$ be a set of m machines. The value of tasks j is much greater than that of m, that is, some machines should be shared among multiple tasks. The multi-objective optimization model is subjected to the following constraints:

(1) The tasks must be completed before the delivery date.

(2) The tasks cannot be interrupted during the processing.

(3) The machine exchange time is a negligible constant.

(4) Each task cannot be processed by multiple machines simultaneously, and each machine can only process one task at a time.

(5) The tasks can be completed in advance, but not delayed.

The symbols in the model are defined in Table I below.

Table I: Symbol definitions.

Name	Definition
r_j	Start time of the task
C_j	Completion time of the task
d_j	Delivery date of the task
E_j	Early completion time of the task
S_m	The set of tasks processed on machine M_m
$X_{jm} = 1$	Task J_j is not processed on machine M_m
$X_{jm} = 0$	Task J_j is processed on machine M_m

The multi-objective optimization model should satisfy the following hypotheses:

Hypothesis 1: The tasks on each machine are arranged by the start time without emission reduction, and have no time conflict.

Hypothesis 2: All tasks are completed before the delivery date.

Hypothesis 3: The problem has one or more optimal solutions with minimal early completion time.

Based on the above hypotheses, the multi-objective optimization model for the JSP, machines are shared among multiple tasks, can be finalized as:

Objective function:
$$\min\ E = \sum_{i=1}^{j} E_i \tag{2}$$

$$\text{s.t.} \begin{cases} d_j - c_j \geq 0 \\ E_j = d_j - c_j \\ c_j \leq r_{j+1},\ \ J_i, J_{i+1} \in S_m \\ \sum_{m=1}^{M} X_{im} = 1, i = (1, 2, \ldots, j),\ X_{jm} \in \{0, 1\} \\ J = \{J_1, J_2, \ldots, J_j\} \\ M = \{M_1, M_2, \ldots, M_m\} \end{cases}$$

4. MULTI-OBJECTIVE SCHEDULING MODEL BASED ON IMPROVED NSGA

4.1 Design of the NSGA

The NSGA introduces the crowding mechanism [20] to the selection process. Firstly, the high-quality individuals are selected by the crowding fitness (*CF*), which is determined based

on the parent population, and then crowded out according to their similarity with their offspring. Next, the individuals for the new population are selected by the new fitness. Finally, the individual quality is improved through local search.

Let *PopSize* be the population size and *Gen* be the maximum number of iterations. Then, the NSGA based on crowding mechanism can be designed as:

Step 1: Initialize the population by greedy algorithm, and compute the fitness *F* of each individual.

Step 2: Select the first $CM = 1/CF$ chromosomes as crowding members.

Step 3: Select a set of high-quality individuals.

Step 4: Perform heuristic crossover on the selected individuals. After that, replace the relatively poor individuals with new individuals.

Step 5: Perform mutation on the population obtained through crossover.

Step 6: Perform crowding operations. Firstly, merge the child chromosomes obtained in Step 5 with the crowding members selected in Step 2 into a pool of $CM + PopSize$ individuals. After that, calculate the distance between two randomly selected individuals x_i and x_j in the pool: $d_{ij} = \sum_{i=1}^{CM+PopSize-1} \sum_{j=i+1}^{CM+PopSize} (x_i - x_j)^2$. Compare the fitness the two individuals, if their distance is smaller than the threshold *L*. With the elapse of time, the probability for the remaining individuals to be selected continues to decrease. Finally, rank the $CM + PopSize$ individuals in ascending order of new fitness, and take the first *PopSize* individuals as candidates for the pool.

Step 7: Perform local retrospective search on the selected individuals to find the local optimal solution for each individual, and add it to the next-generation population.

Step 8: Judge whether $T = Gen$ is satisfied or not. If yes, output the optimal solution and terminate the algorithm; Otherwise, $T = t + 1$, and return to Step 4.

4.2 Fitness function

For the GA, the best individual must be selected in each iteration. In this paper, the multi-objective optimization model is transformed into a constrained single objective function through normalization, in the light of preference information. A penalty term was applied on the individuals failing to meet the constraints, forming an evaluation function. This function was adopted to assess the quality of each individual.

The final solution of the JSP, in which machines are shared among multiple tasks, should cater to the unique demand of each task through the following steps:

Step 1: Assuming that job J_i is processed on the p^{th} machine, the mean completion time AT_j and mean cost EC_j of a job on the machine set M_p can be respectively computed by:

$$AT_j = \frac{\sum_{k=1}^{num_p} (W_{ijk} + WT_k)}{num_p} \tag{3}$$

$$EC_j = \frac{\sum_{k=1}^{num_p} (C_{pk} \times W_{ijk})}{num_p} \tag{4}$$

Step 2: Calculate AT_j / UTE_j, the ratio of AT_j to the expected completion time of job J_i, and EC_j / UCE_j, the ratio of EC_j to the expected cost of job J_i.

Step 3: Calculate the mean AT_j / UTE_j and mean EC_j / UCE_j of each task.

Step 4: Take the number (a_1) of tasks whose EC_j / UCE_j is smaller than the mean EC_j / UCE_j as the weighting factor of objective function f_1 in the objective function *Z*, and take $(n - a_1)$ as the weighting factor of objective function f_2. To eliminate the differences in unit and quantity, normalize the two objective functions by:

$$f_i' = \frac{f_i - f_{low}}{f_{up} - f_{low}} \tag{5}$$

where f_{up} and f_{low} are the upper and lower bounds of objective function f_i, respectively.

For completion time f_1,

$$f_{up} = \sum_{i=1}^{n} (\max \sum_{k=1}^{num_p} (W_{ijk} + WT_k)) \tag{6}$$

$$f_{low} = \sum_{i=1}^{n} min \sum_{k=1}^{num_p} W_{ijk} \tag{7}$$

For cost f_2,

$$f_{up} = \sum_{i=1}^{n} (\max \sum_{k=1}^{num_p} (C_{pk} \times W_{ijk})) \tag{8}$$

$$f_{low} = \sum_{i=1}^{n} (min \sum_{k=1}^{num_p} (C_{pk} \times W_{ijk})) \tag{9}$$

Taking the constraints as penalty terms, an integrated fitness function can be established as:

$$F = \sum_{i=1}^{r} w_i f_i'(r = 1, 2) + \lambda * \Delta Q \tag{10}$$

where $w_1 = a_1$; $w_2 = n - a_1$; λ is a penalty term; ΔQ is the degree of constraint conflict.

4.3 Heuristic crossover

The crossover operation swaps some genes in two parent chromosomes at a preset probability *pc*. After the swap, the genes are recombined into new child chromosomes. This operation increases the probability of finding the optimal solution, and guides the population evolution to a better level. In this paper, the heuristic knowledge about the research problem is integrated into the crossover operator, and the heuristic crossover is carried out according to the genetic loci.

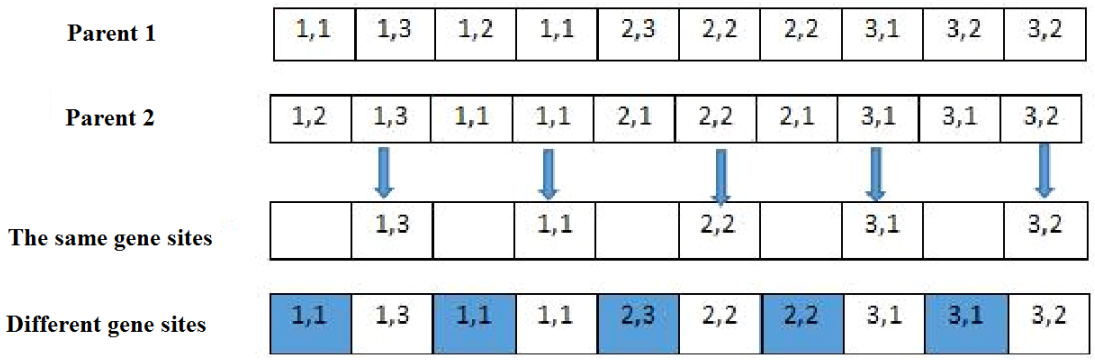

Figure 1: Example of heuristic crossover.

Firstly, the basic patterns of both parent chromosomes were retained in the child chromosomes. Next, the high-quality genes that provide good individual traits were kept in the next generation at a relatively high probability, so that the child chromosomes can have a good genetic structure. Fig. 1 illustrates the heuristic crossover when *parent*1 has better fitness than *parent*2.

Next, a local search operator was designed based on the research problem, the coding structure and the representative solutions. After the crossover, the local search

was performed on the chromosomes to identify the potential high-quality genes hidden near the selected genes, and pass them to the next generation. This operation helps to enhance solution quality and speed up convergence.

The local search is implemented in the following steps:

Step 1: From left to right, take each genetic locus of each chromosome as the centre, and perform neighbourhood search in a stepwise manner.

Step 2: Compare the fitness of all genes in the searched domain, and extract the optimal genes.

Step 3: Compare the selected best genes with the existing genes in the current neighbourhood of the chromosome. Replace the existing genes with the selected genes if the former is worse than the latter. Otherwise, keep the existing genes unchanged.

Step 4: Repeat the above steps until the neighbourhood no longer produces better individuals.

5. SIMULATION AND RESULTS ANALYSIS

The proposed model was verified through simulation. It is assumed that 15 machines are reserved for incoming tasks. The machines fall into three types: A, B and C. There are 2 type A machines, 2 type B machines and 4 type C machines. The information on jobs and machines are given in Tables II and III, respectively.

Table II: Information of the jobs.

Job	**Time of completing the process**									***UTE***	***UTC***
	$M_{A,1}$	$M_{A,2}$	$M_{A,3}$	$M_{B,1}$	$M_{B,2}$	$M_{C,1}$	$M_{C,2}$	$M_{C,3}$	$M_{C,4}$		
J_1	2	3	1.5	-	-	-	-	-	-	1.6	30
J_2	1	2	1	-	-	-	-	-	-	-	25
J_3	2.5	4	1.5	-	-	-	-	-	-	18	20
J_4	3.5	6	2	-	-	-	-	-	-	6	-
J_5	3	5	2	-	-	-	-	-	-	-	-
J_6	-	-	-	1.5	4	-	-	-	-	-	45
J_7	-	-	-	2	3	-	-	-	-	2.6	-
J_8	-	-	-	0.5	1.5	-	-	-	-	-	-
J_9	-	-	-	1.5	2	-	-	-	-	-	30
J_{10}	-	-	-	-	-	3.6	7	3	2.5	-	-
J_{11}	-	-	-	-	-	2	3	1.5	1	0.8	-
J_{12}	-	-	-	-	-	5	7	2.5	2	-	80
J_{13}	-	-	-	-	-	4.5	8	3.5	4	3	150
J_{14}	-	-	-	-	-	1.5	4	1	0.5	1	-
J_{15}	-	-	-	-	-	2.5	5	2	1.5	-	90

Table III: Information of the machines.

Equipment number	$M_{A,1}$	$M_{A,2}$	$M_{A,3}$	$M_{B,1}$	$M_{B,2}$	$M_{C,1}$	$M_{C,2}$	$M_{C,3}$	$M_{C,4}$
Equipment type	A type			B type		C type			
Unit time cost of equipment	10	20	15	50	15	40	15	45	60

During simulation, the multi-objective JSP was encoded by a double-layer coding method. The upper layer encodes the machine selection problem, and the lower layer encodes the operation sequence. Based on the job sequence (Table IV), the jobs and operations were coded (Fig. 2).

Table IV: The job sequence.

Workpiece	Processes	Processing time				
		M_1	M_2	M_3	M_4	M_5
$J_1=1$	O_{11}	3	2	5	-	7
	O_{12}	3	-	5	-	3
	O_{13}	5	-	-	2	3
$J_2=1$	O_{21}	2	-	4	-	6
	O_{22}	3	3	5	-	-

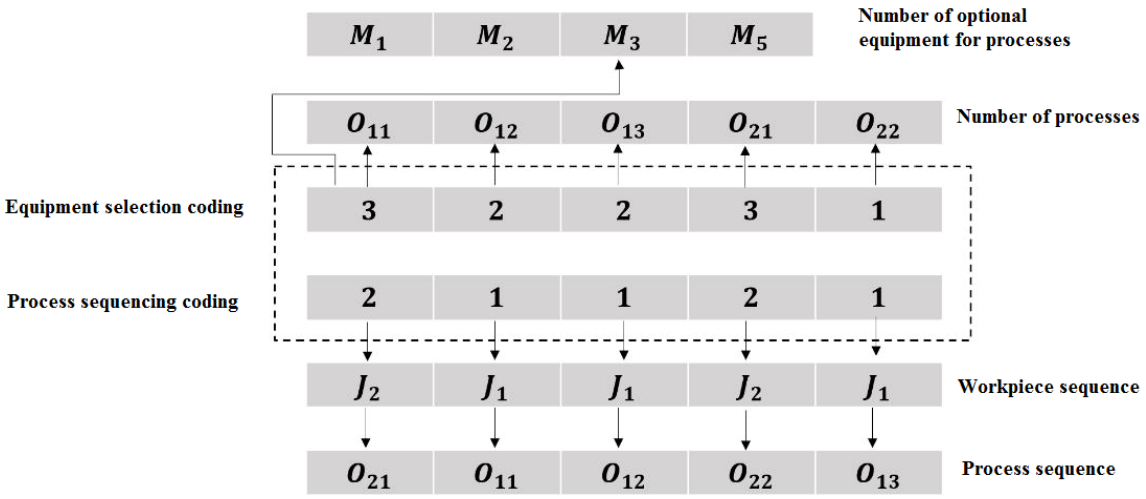

Figure 2: Coded jobs and operations.

The simulation parameters were initialized as follows: crossover probability $pc = 0.5$, population size $PopSize = 50$, the maximum number of iterations $Gen = 200$, and the simulation runs $Run = 100$. The final results of the simulation are listed in Table V.

The final results show that each task can have various machine sequences, and the result varies with the scheduling objectives.

Furthermore, an extreme scheduling scenario with six machines was simulated. The production was performed according to the proposed algorithm. However, the fifth machine suddenly failed when the production reached the 10th time point. Then, the schedule was redesigned by our algorithm as Fig. 3.

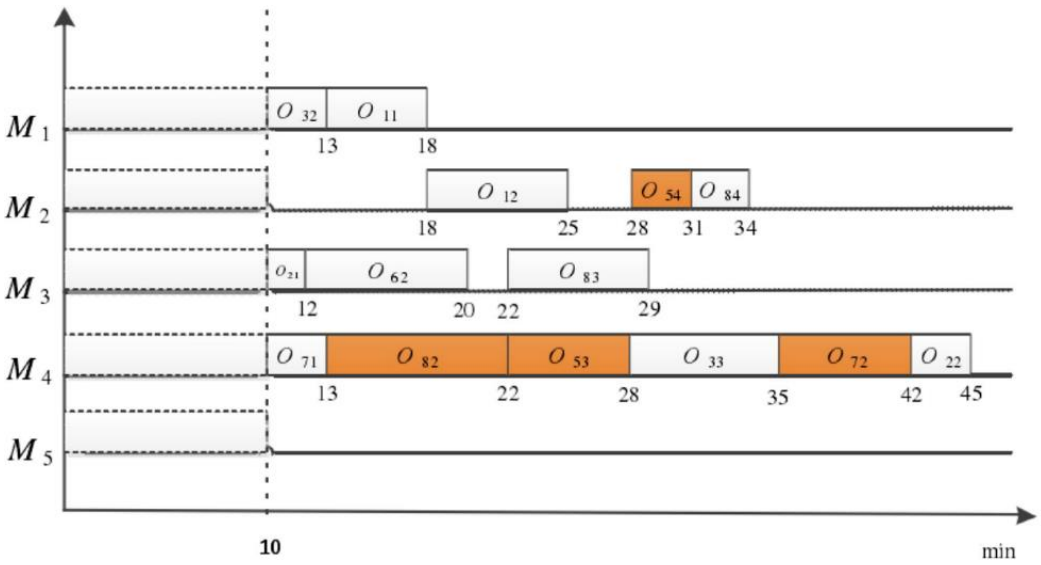

Figure 3: The new schedule after machine failure.

Table V: The optimal schedule.

Workpiece	Processes	Selected equipment	Processing time
J_1	O_{11}	$M_{A,2}$	2
	O_{12}	$M_{C,2}$	3
	O_{13}	$M_{A,3}$	3
J_2	O_{21}	$M_{B,1}$	1
	O_{22}	$M_{A,1}$	2
	O_{23}	$M_{C,4}$	3
J_3	O_{31}	$M_{A,3}$	2
	O_{32}	$M_{C,1}$	4
	O_{33}	$M_{C,3}$	2
J_4	O_{41}	$M_{C,3}$	3
	O_{42}	$M_{B,2}$	3
	O_{43}	$M_{A,1}$	2
J_5	O_{51}	$M_{B,2}$	4
	O_{52}	$M_{A,1}$	2
	O_{53}	$M_{C,3}$	4
J_6	O_{61}	$M_{C,4}$	3
	O_{62}	$M_{A,2}$	1
	O_{63}	$M_{B,2}$	2
J_7	O_{71}	$M_{C,2}$	2
	O_{72}	$M_{B,2}$	3
	O_{73}	$M_{B,1}$	1
J_8	O_{81}	$M_{A,3}$	5
	O_{82}	$M_{C,4}$	2
	O_{83}	$M_{B,2}$	2
J_9	O_{91}	$M_{C,2}$	3
	O_{92}	$M_{C,4}$	1
	O_{93}	$M_{B,1}$	3
J_{10}	O_{101}	$M_{C,2}$	3
	O_{102}	$M_{B,2}$	2
	O_{103}	$M_{A,1}$	2
J_{11}	O_{111}	$M_{C,1}$	4
	O_{112}	$M_{B,2}$	2
	O_{113}	$M_{B,1}$	1
J_{12}	O_{121}	$M_{B,1}$	3
	O_{122}	$M_{C,2}$	3
	O_{123}	$M_{C,3}$	3
J_{13}	O_{131}	$M_{C,1}$	2
	O_{132}	$M_{A,3}$	1
	O_{133}	$M_{C,4}$	2
J_{14}	O_{141}	$M_{C,1}$	2
	O_{142}	$M_{C,2}$	3
	O_{143}	$M_{A,2}$	4
J_{15}	O_{151}	$M_{A,3}$	3
	O_{152}	$M_{C,1}$	2
	O_{153}	$M_{C,2}$	1

6. CONCLUSIONS

This paper puts forward a multi-objective optimization model for the JSP, in which machines are shared among multiple tasks. To solve the model, a scheduling strategy was proposed

based on the NGSA with crowding mechanism. Compared with the GA, the improved NGSA can effectively avoid the local optimum trap and maintain population diversity in the later stage. In addition, the heuristic crossover operator was introduced to enhance the local search ability of the improved NGSA. The effectiveness of the proposed scheduling strategy was proved valid through simulation.

REFERENCES

[1] Leu, J.-S.; Chen, C.-F.; Hsu, K.-C. (2014). Improving heterogeneous SOA-based IoT message stability by shortest processing time scheduling, *IEEE Transactions on Services Computing*, Vol. 7, No. 4, 575-585, doi:10.1109/TSC.2013.30

[2] Yang, X. P.; Gao, X. L. (2018). Optimization of dynamic and multi-objective flexible job-shop scheduling based on parallel hybrid algorithm, *International Journal of Simulation Modelling*, Vol. 17, No. 4, 724-733, doi:10.2507/IJSIMM17(4)CO19

[3] Wong, T. C.; Ngan, S. C. (2013). A comparison of hybrid genetic algorithm and hybrid particle swarm optimization to minimize makespan for assembly job shop, *Applied Soft Computing*, Vol. 13, No. 3, 1391-1399, doi:10.1016/j.asoc.2012.04.007

[4] Li, B.-Z.; Wu, S.-S.; Yang, J.-G.; Zhou, Y.-Q.; Du, M. (2012). A three-fold approach for job shop problems: A divide-and-integrate strategy with immune algorithm, *Journal of Manufacturing Systems*, Vol. 31, No. 2, 195-203, doi:10.1016/j.jmsy.2011.05.005

[5] Banihani, S.; Al-Widyan, K.; Al-Jarrah, A.; Ababneh, M. (2013). A genetic algorithm based lookup table approach for optimal stepping sequence of open-loop stepper motor systems, *Journal of Control Theory and Applications*, Vol. 11, No. 1, 35-41, doi:10.1007/s11768-013-1165-4

[6] Li, F.-Q.; Wu, N.-Q. (2014). Modeling and solution of lane reservation problem with transportation task merging, *Systems Engineering – Theory & Practice*, Vol. 34, No. 6, 1599-1606, doi:10.12011/1000-6788(2014)6-1599

[7] Grillo, S.; Pievatolo, A.; Tironi, E. (2016). Optimal storage scheduling using Markov decision processes, *IEEE Transactions on Sustainable Energy*, Vol. 7, No. 2, 755-764, doi:10.1109/TSTE.2015.2497718

[8] Akhshabi, M.; Tavakkoli-Moghaddam, R.; Rahnamay-Roodposhti, F. (2014). A hybrid particle swarm optimization algorithm for a no-wait flow shop scheduling problem with the total flow time, *The International Journal of Advanced Manufacturing Technology*, Vol. 70, No. 5-8, 1181-1188, doi:10.1007/s00170-013-5351-9

[9] Zhang, L.; Gao, L.; Li, X. (2013). 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, 3516-3531, doi:10.1080/00207543.2012.751509

[10] Asefi, H.; Jolai, F.; Rabiee, M.; Tayebi Araghi, M. E. (2014). A hybrid NSGA-II and VNS for solving a bi-objective no-wait flexible flowshop scheduling problem, *The International Journal of Advanced Manufacturing Technology*, Vol. 75, No. 5-8, 1017-1033, doi:10.1007/s00170-014-6177-9

[11] Geng, J.-C.; Cui, Z.; Gu, X.-S. (2016). Scatter search based particle swarm optimization algorithm for earliness/tardiness flowshop scheduling with uncertainty, *International Journal of Automation and Computing*, Vol. 13, No. 3, 285-295, doi:10.1007/s11633-016-0964-8

[12] Zhang, H.; Collart-Dutilleul, S.; Mesghouni, K. (2015). Cyclic scheduling of flexible job-shop with time window constraints and resource capacity constraints, *IFAC – PapersOnLine*, Vol. 48, No. 3, 816-821, doi:10.1016/j.ifacol.2015.06.184

[13] Jin, H.; Wong, M.-L. (2010). Adaptive, convergent, and diversified archiving strategy for multiobjective evolutionary algorithms, *Expert Systems with Applications*, Vol. 37, No. 12, 8462-8470, doi:10.1016/j.eswa.2010.05.032

[14] Yi, J.-H.; Deb, S.; Dong, J.-Y.; Alavi, A. H.; Wang, G.-G. (2018). An improved NSGA-III algorithm with adaptive mutation operator for Big Data optimization problems, *Future Generation Computer Systems*, Vol. 88, 571-585, doi:10.1016/j.future.2018.06.008

[15] Rohaninejad, M.; Kheirkhah, A.; Fattahi, P.; Vahedi-Nouri, B. (2015). A hybrid multi-objective genetic algorithm based on the ELECTRE method for a capacitated flexible job shop scheduling problem, *The International Journal of Advanced Manufacturing Technology*, Vol. 77, No. 1-4, 51-66, doi:10.1007/s00170-014-6415-1

[16] Luo, H.; Du, B.; Huang, G. Q.; Chen, H.-P.; Li, X.-L. (2013). Hybrid flow shop scheduling considering machine electricity consumption cost, *International Journal of Production Economics*, Vol. 146, No. 2, 423-439, doi:10.1016/j.ijpe.2013.01.028

[17] Lei, D.-M. (2011). Simplified multi-objective genetic algorithms for stochastic job shop scheduling, *Applied Soft Computing*, Vol. 11, No. 8, 4991-4996, doi:10.1016/j.asoc.2011.06.001

[18] Cai, X.; Li. M.-Y.; Wang, K.; Xiao, J. (2013). Float-code based differential evolutionary multi-objective optimization for flexible job-shop scheduling problem, *Application Research of Computers*, Vol. 30, No. 4, 999-1003

[19] Wang, Y.; Yang, O.; Wang, S. N. (2019). A solution to single-machine inverse job-shop scheduling problem, *International Journal of Simulation Modelling*, Vol. 18, No. 2, 335-343, doi:10.2507/IJSIMM18(2)CO7

[20] 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