An Integrated Scheduling Approach for Flexible JSP

Tao Ze[1, a],Wan Runbei [2, b]

[1] School of Mechanical Engineering, Shenyang Ligong University, Shenyang 110159, China

[2] University of Southern California, Los Angeles, 90007, USA

[a] taoze@tsinghua.edu.cn, [b] runbeiwa@usc.edu

Keywords: Petri net; object-oriented modeling; job shop scheduling problem(JSP)

Abstract. A colored Petri net is used to model the job shop scheduling problem with dual-resource constraints. The objective of scheduling problems is to minimize make-span. Firstly, the Petri net model of the flexible JSP is constructed based on object-oriented approach. Secondly, GA is applied based on the colored Petri net model. Finally, one example is applied to test the effectiveness of the method.

Introduction

Today, the manufacturing environment is characterized as having diverse products, high quality, short delivery time, and unstable customer demand. In order to provide wide product variety and quick response to changes in market place, flexible manufacturing systems(FMSs) have been adopted broadly in modern production environments[1]. As one class of typical production scheduling problems, job shop scheduling with flexible processing routing is one of the strongly NP-complete combinatorial optimization problems. It is difficult to obtain performance such as make-span, machine load distribution, job queue length and so on. Thus, a performance evaluation method of scheduling through simulation based on flexible product line model is important. Hence, the need arises in flow shop scheduling problem (FSP) for powerful graphical and analytical tools such as Petri net (PN) and search techniques such as genetic algorithm (GA)[2-5].

Petri net model in flexible JSP scheduling

In order to describe a colored Petri net model for flexible job shop scheduling. Consider the following FMS system, there are three types of machines in the system, i.e., type I: machine 1, 2, type Ⅱ: machine 3, and typeⅢ: machine 4, 5 and 6. The resource demand on workers and machines are listed in table 1and table 2.

Table 1. Working table of workers and machines

Labor	Machine 1	Machine 2	Machine 3	Machine 4	Machine 5	Machine 6
1	∨	∨				
2		∨	∨			
3				∨	∨	
4					∨	∨

Table 2. Processing time of machines

Job	Operation	Time	Machine type	Job	Operation	Time	Machine type
1	1	10	II	3	3	7	III
	2	7	I	4	1	5	I
	3	8	III		2	7	III
2	1	10	II		3	10	I
	2	12	III	5	1	9	I
	3	8	I		2	7	III
3	1	8	I		3	10	I

Published by Atlantis Press, Paris, France.

2	9	II	6	1	8	II

Petri net model of above example as shown in Fig.1:

p_1^1: the waiting job;

p_1^2: the processing job;

p_2^1: the input buffer of machines;

p_2^2: the preset input buffer of machines;

p_2^3: the occupied input buffer of machines;

p_2^4: the occpuied machines;

p_2^5: the processing machine;

p_2^6: the waiting jobs to output buffer after processing;

p_2^7: the occupied output buffer of machines;

p_2^8: the idle machines;

p_2^9: the idle output buffer of machines;

p_2^{10}: the repairing of breakdown machine;

p_3^1: the idle worker;

p_3^2: the busy worker;

p_3^3: the processing worker on machine;

m_1: requesting machine of job;

m_2: requesting the next operation after one;

t_1^1: requesting processed of jobs;

t_1^2: processing the next operation after one;

t_2^1: the input buffer of machine is preset by certain job;

t_2^3: the machine is preset by certain job;

t_2^4: beginning of processing job;

t_2^2: the input buffer of machine is occupied by certain job;

t_2^6: the output buffer of machines is occupied;

t_2^5: ending of processing

t_2^7: the output buffer of machines is released;

t_2^8: the breaking down of machines;

t_2^9: the repairing of machines;

t_3^1: the worker preset by machine;

t_3^2: processing job on machine by worker;

t_3^3: ending of worker processing;

m_3: requesting worker of machine;

m_4: requesing processing after worker arranging;

m_5: releasing worker after machine processing;

Object of jobs:

$$C_0 = \{C_0^1, C_0^2, \cdots, C_0^{10}\}$$

$$C_0^1 = (J_1, 1, (M_3M_1/M_2M_4/M_5/M_6), (10,7,8))$$

$$C_0^2 = (J_2, 1, (M_3M_4/M_5/M_6M_1/M_2), (10,12,8))$$

$$C_0^3 = (J_3, 1, (M_1/M_2M_3M_4/M_5/M_6), (8,9,7))$$

$$C_0^4 = (J_4, 1, (M_1/M_2M_4/M_5/M_6M_1/M_2), (5,7,10))$$

$$C_0^5 = (J_5, 1, (M_1/M_2M_4/M_5/M_6M_1/M_2), (9,7,10))$$

$$C_0^6 = (J_6, 1, M_3, 8)$$

Class object of machines:

$$C_1 = \{C_1^1, C_1^2, \cdots, C_1^6\}$$

$$C_1^1 = (M_1, 1, W_1)$$

$$C_1^2 = (M_2, 1, W_1/W_2)$$

$$C_1^3 = (M_3, 1, W_2)$$

$$C_1^4 = (M_4, 1, W_3)$$

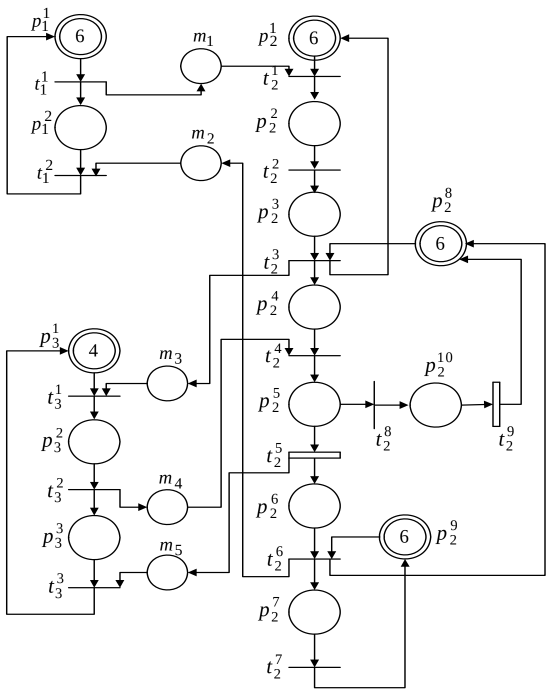

Fig.1 the colored Petri net model

$C_1^5 = (M_5, 1, W_3 / W_4)$

$C_1^6 = (M_6, 1, W_4)$

Class object of workers:

$C_2 = \{C_2^1, C_2^2, C_2^3, C_2^4\}$

$C_2^1 = (W_1, 1, \varnothing)$

$C_2^2 = (W_2, 1, \varnothing)$

$C_2^3 = (W_3, 1, \varnothing)$

$C_2^4 = (W_4, 1, \varnothing)$

The initial markings:

$M_0(P_1^1) = \{C_0^1, C_0^2, \cdots, C_0^6\}$ $\quad M_0(P_2^1) = M_0(P_2^8) = M_0(P_2^9) = \{C_1^1, C_1^2, \cdots, C_1^6\}$

$M_0(P_3^1) = \{C_2^1, C_2^2, C_2^3, C_2^4\}$

PNs and object-oriented design concepts are complementary to achieve the goal of system development at incremental stage and enable us to model the complex manufacturing systems indetail, so that the strategies to operate these systems effectively can be applied in a more realistic environment.

GA Algorithm

GA is powerful and broadly applicable stochastic search and optimization techniques based on principles from the evolution theory. Each solution in the population is called a chromosome, which represents a point in the search space. Each solution in the population is called a chromosome, which represents a point in the search space. In this paper, the chromosomes are evolved through successive iterations by crossover and mutation. Also, a fitness value is assigned to each individual according to the problem-specific objective function. The feasibility of chromosomes is checked in order to ensure that a solution decoded from a chromosome lies in the feasible region of the problem[5]. The details of GA are as follows:

- Permutation representation represents a solution of a problem as chromosome.
- Create initial population and fitness function: Create the initial population randomly, which may lead to solutions with diverse forms.
- Crossover operation: Before the operation, divide population into K sub-populations (4 sub-populations in simulation): select the optimum individual in each sub-population and crossover with other individuals until producing K new populations, then select the best individual. Adopt MPPX, MGOX, MGPMX1, and MGPMX2 [10] in simulation, which can make the population have obvious diversity.

Selection. Select optimum chromosome between offspring chromosome produced by crossover in different sub-population and parent, which can accelerate the evolution process.

- Mutation: INV mutation is used to produce small perturbations on chromosomes in order to maintain the diversity of population.
- Enhancing memory ability: In order to avoid losing the current optimum solution in the search process, save the current optimum solution through adding memory.
- Criterion of algorithm termination: Given the iterations T, if the searching iterations are equal to T then stop searching.

Case Study

To illustrate the effectiveness of the proposed approach, one problem is tested in this section. The result can be represented as a Gantt graph with make-span as the criterion, which depicts the starting and ending time of all operations on each machine. Consider the following FMS system. The resource demand on workers and machines are listed in table 1, table 2, respectively. The algorithm parameters are that population size is 60, crossover rate is 0.85, mutation rate is 0.01.

From table 2, for a dual-resource problem, each job is allowed to have one more routings. Table 1 shows that one labor can control two different machines. The scheduling result is shown in Fig.2. Fig.2 shows the distribution between working procedure and machines. The abscissa indicates time duration consumed by jobs and the ordinate indicates machine. Different jobs and their operations are expressed with four-digit numbers in the Gantt graph. The first two represent job number, the third represents operation number, and the last is for labor. For example, on the machine 1 "0311" represents operation 1 of job 3 processed by labor 1.

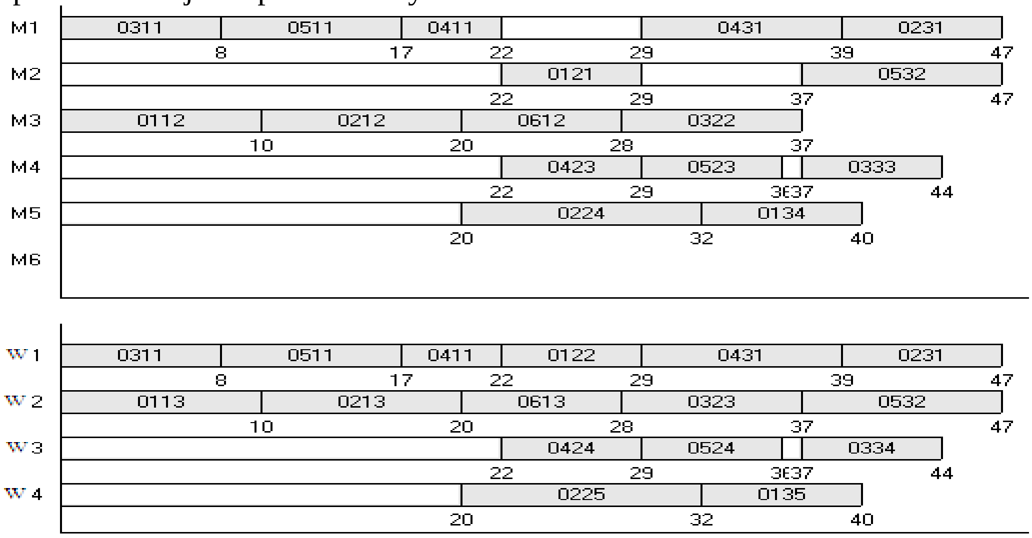

Fig.2 Gantt graph of scheduling

Conclusion

In this paper, a flexible job shop scheduling problem is studied. The colored Petri net model is constructed. The scheduling results can be carried out using a GA with make-span as the criterion based on the constructed Petri net model. According to the scheduling results, the method is proved to be effective and feasible.

References

[1] ChenJ, ChenFF. Performance modeling and evaluation of dynamic tool allocation in flexible manufacturing systems using coloured Petri nets: An object-oriented approach. Int J Adv Manuf Technol Vol 21(2): 98-109(2003).

[2] Chen J H, Fu L C, Lin M H. Petri-net and GA-based approach to modeling, scheduling, and performance evaluation for wafer fabrication. IEEE Transaction on Robotics and Automation, 17(5): 619-636(2001).

[3] Hao D, Jiang C J, and Lin L. Petri Net Based Modeling and GA Based Scheduling for FMS. Chinese Journal of Computers, 28 (2): 201-208(2005).

[4] Jiang S J, Li Z H. Petri nets based dynamic optimal model for flexible manufacturing system. Computer Integrated Manufacturing Systems, 11 (4): 462-466(2005).

[5] Byung J, Hyung R, Hyun S. A hybrid genetic algorithm for the job shop scheduling problems, Computers & Industrial Engineering, 45: 597-613(2003).

[6] Thonton H W, Hunsucker JL. A new heuristic for minimal makespan in flow shops with multiple processors and no intermediate storage. European Journal of Operational Research 152:96-114(2004).