한국정밀공학회지 제 32 권 1 호 pp. 25-37 January 2015 / 25
J. Korean Soc. Precis. Eng., Vol. 32, No. 1, pp. 25-37 http://dx.doi.org/10.7736/KSPE.2015.32.1.25
ISSN 1225-9071(Print), ISSN 2287-8769(Online)

◆ 특집 ◆ IT 융합 생산/가공기계 지능화 기술

다중 공정계획을 가지는 정적/동적 유연 개별공정에 대한 의사결정 나무 기반 스케줄링

Decision Tree based Scheduling for Static and Dynamic Flexible Job Shops with Multiple Process Plans

유재민 [1], 도형호 [1], 권용주 [1], 신정훈 [1], 김형원 [1], 남성호 [2], 이동호 [1,✉]
Jae-Min Yu[1], Hyoung-Ho Doh[1], Yong-Ju Kwon[1], Jeong-Hoon Shin[1], Hyung-Won Kim[1], Sung-Ho Nam[2], and Dong-Ho Lee[1,✉]

1 한양대학교 산업공학과 (Department of Industrial Engineering, Hanyang University)
2 한국생산기술연구원, IT 융합공정연구실용화그룹 (The IT Converged Process R&BD Group, Korea Institute of Industrial Technology)
✉ Corresponding author: leman@hanyang.ac.kr, Tel: +82-2-2220-0475

Manuscript received: 2014.10.14 / Revised: 2014.12.16 / Accepted: 2014.12.17

This paper suggests a decision tree based approach for flexible job shop scheduling with multiple process plans. The problem is to determine the operation/machine pairs and the sequence of the jobs assigned to each machine. Two decision tree based scheduling mechanisms are developed for static and dynamic flexible job shops. In the static case, all jobs are given in advance and the decision tree is used to select a priority dispatching rule to process all the jobs. Also, in the dynamic case, the jobs arrive over time and the decision tree, updated regularly, is used to select a priority rule in real-time according to a rescheduling strategy. The two decision tree based mechanisms were applied to a flexible job shop case with reconfigurable manufacturing cells and a conventional job shop, and the results are reported for various system performance measures.

Key Words: Flexible job shop (유연개별공정), Scheduling (일정계획), Multiple process plans (다중 공정계획), Decision tree (의사결정나무)

1. Introduction

Considerable attention has been given to job shop scheduling during the last decades due to its theoretical importance and a number of practical applications. In the theoretical aspect, it is one of well-known combinatorial optimization problems that have been proved to be NP-hard in a strong sense.[1,2] Also, in the practical aspect, it has numerous applications, especially low-volume and high-variety manufacturing and service systems. See Jain and Meeran[3] for a survey on job shop scheduling.

To enlarge the applicability of the classical job shop scheduling problem, various extensions were proposed to cope with the specific system requirements, e.g. job shop

scheduling with alternative operations/machines, open shop scheduling, assembly job shop scheduling, job shop scheduling with operators, etc. Among them, we focus on the job shop scheduling problem with alternative operations and/or machines, called the flexible job shop scheduling problem (FJSP) in the literature. Unlike the classical job shop scheduling problem in which each operation of a job can be processed on a specific machine, the FJSP has operation and/or routing flexibilities.

The FJSP can be found in various manufacturing systems, especially in flexible manufacturing systems (FMSs) or reconfigurable manufacturing systems (RMSs) since they have the capability that each operation can be processed on one or more machines. Here, the RMS is the one designed for rapid changes in its hardware/software components to quickly adjust its production capacity and functionality in response to sudden market changes or intrinsic system changes. See Koren et al.[4] for more details on the RMS. In the case that one or more reconfigurable manufacturing cells are introduced to a conventional job shop for the purpose of increasing system capacity and flexibility, the resulting scheduling problem becomes the FJSP since each job can be processed through alternative operations, each of which can be processed on either the reconfigurable manufacturing cells or the conventional job shop.

The previous studies on the FJSP can be classified according to the types of process plans: (a) single process plans with only alternative machines for each operation; and (b) multiple process plans with both alternative operations and machines. Sew Iwata et al.,[5] Nasr and Elsayed,[6] Lee and Kim,[7] Low and Wu[8] for previous studies on the FJSP with single process plans. Unlike those on the FJSP with single process plans, not many studies have been done on the FJSP with multiple process plans. Lee et al.[9] suggest a genetic algorithm that minimizes makespan, and Kim et al.[10] suggest a symbiotic evolutionary algorithm that minimizes makespan and mean flow time. Also, Ozguven et al.[11] develop a mixed integer programming model. See Baykasoglu,[12] Baykasoglu et al.,[13] Doh et al.[14] and Yu et al.[15] for other approaches and applications on the FJSP with multiple process plans.

This study considers the flexible job shop scheduling problem with multiple process plans, denoted by the FJSP-MPP in this paper. As explained earlier, a multiple process plan for a job specifies both alternative operations and alternative machines for each operation of the job. According to the characteristics of the problem, the FJSP-MPP has two main decisions: (a) selecting operation/machine pairs; and (b) sequencing the jobs assigned to each machine, i.e. classical job shop scheduling.

Due to the complexity of the problem, as explained earlier, most previous studies suggested meta-heuristics or priority scheduling methods. Unlike these, we suggest a decision tree based scheduling approach in which the decision tree is used to select a priority rule combination appropriate for a specific system state. In general, the decision tree, one of the data mining techniques, is a kind of classifier expressed as a recursive partition of the instance space. In this study, we adopt the decision tree since it has several advantages, e.g. simple to understand and interpret, containing valuable information with little data, capability to add possible scenarios, etc. See Deng et al.[16] for other advantages of the decision tree.

Although the existing meta-heuristic approach can give good quality solutions, but it requires too much computation time to be used in real-time. Also, the priority scheduling approach has the burdens required for carrying out simulation runs to select the best rule for a specific system state. On the other hand, the decision tree based scheduling approach has the capability to select a best priority rule combination appropriate for a specific system state, and hence the burdens to carry out simulation runs when scheduling decision is done can be eliminated. Note that this study was motivated from a practical research project whose results must be commercialized for potential users while providing the performances competitive to the simulation based priority rule scheduling approach and hence the decision tree based scheduling approach was more appropriate. Also, in the theoretical aspect, this study is the first one to suggest the decision tree based scheduling approach for the FJSP-MPP. See Shinichi and Taketoshi,[17] Shaw et al.,[18] Piramuthu et al.,[19] Park et al.,[20] Lee et al.,[21] Arzi and Iaroslavitz,[22] Su and Shiue,[23] Kwak and Yih,[24] and Choi et al.[25] for applications of the decision tree to other scheduling problems.

In this study, we consider static and dynamic versions

of the FJSP-MPP. In the static case, all jobs are given in advance and the decision tree is used to select a priority dispatching rule to process all the jobs, i.e. there are no rule changes over the scheduling horizon. On the other hand, in the dynamic case, the jobs arrive over time and the decision tree, updated regularly using the state-of-the art operation data, is used to select a priority dispatching rule in real-time according to a rescheduling strategy, e.g. machine breakdowns, urgent orders, etc. To show the performances of the two decision tree based scheduling approaches, they were applied to a flexible job shop with reconfigurable manufacturing cells and a conventional job shop, and the test results are reported for various performance measures.

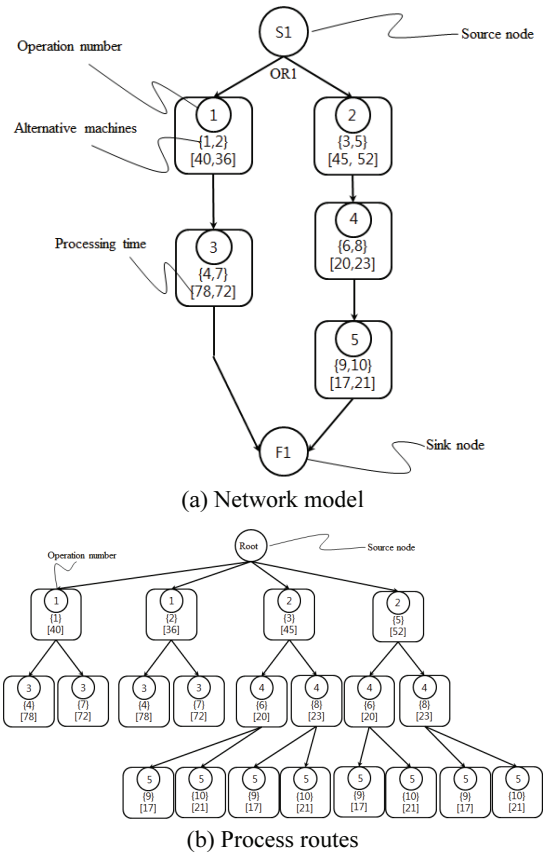
The rest of this paper is organized as follows. In the next section, the FJSP-MPP is described in more details, together with a mixed integer programming model for the static problem. The decision tree based scheduling approach is explained in Section 3, and case results are reported in Section 4. Finally, Section 5 concludes the paper with a summary and discussion of future research.

2. Problem description

As stated earlier, we consider two types of the FJSP-MPP, i.e. static and dynamic ones. Of the two types, this section explains the static version of the problem in which all jobs are given in advance. Note that the dynamic problem is the same as the static one except that the jobs arrive randomly over time, not given in advance.

In the static FJSP-MPP, there are *n* independent jobs, each of which is to be processed on *m* machines. Each job is processed according to a pre-determined multiple process plan that specifies alternative operations, their sequence and alternative machines on which each operation of the job is to be processed.

More specifically, a multiple process plan can be represented by the network model of Ho and Moodie,[26] in which there are three types of nodes: source, intermediate and sink nodes. Source and sink nodes are dummy ones that represent starting and ending of a job, respectively. Intermediate nodes represent alternative operations, alternative machines and processing times. Also, each arc represents the precedence relation between the corresponding two operations. In particular, the OR



(a) Network model

(b) Process routes

Fig. 1 Multiple process plan: an example

relations are used to represent alternative operation/machine pairs. In other words, if a job meets an OR relation, the job must select one of the corresponding alternative operation/machine pairs. In the model, a job is completed through a path from source to sink node, called a process route in this paper. Fig. 1(a), adopted from Doh et al.[14] shows an example for a multiple process plan with 1 OR relation and 5 intermediate nodes. In this figure, we can see that there are two process routes, i.e. S1-1-3-F1 and S1-2-4-5-F1. Also, each intermediate node has the information on alternative machines and processing times. For example, operation 1 can be processed on either machine 1 or 2 in which the corresponding processing times are 40 and 36, respectively. Also, Fig. 1(b) shows a tree that represents all possible process routes.

Now, the static version of the FJSP-MPP can be briefly explained as follows. *For a given set of jobs, each*

with a multiple process plan, the problem is to determine the operation/machine pairs to process each job (process routes) and the sequence of the jobs assigned to each machine according to the selected process routes. The objectives considered in this study are makespan, total flow time and total tardiness, respectively.

According to the static version of the problem, all jobs are ready for processing at time zero, i.e. zero ready times. Also, it is assumed that the job descriptors, such as multiple process plans, processing times, due dates, etc., are deterministic and given in advance. Other assumptions made for the problem are: (a) each machine can process only one operation at a time; (b) setup times are sequence-independent and hence can be included in processing times; (c) preemption is not allowed; and (d) transportation times among machines are ignorable or can be included in processing times.

The problem can be described more clearly as the following mixed integer programming model of Ozguven et al..[11] Before presenting the model, the notations used are summarized below.

Sets and indices

i jobs, $i \in I$

j operations, $j \in O$

k machines, $k \in M$

p index for process routes of job i, $p \in P_i$, where P_i denotes the set of process routes for job i

O_{ip} ordered set of operations in process route p of job i, where $O_{ip} \in O$

o_{ipj} operation j of process route p of job i, where $o_{ipj} \in O_{ip}$ (Note that $o_{ipf(i,p)}$ and $o_{ipl(i,p)}$ imply the first and the last operations of O_{ip}.)

O_i set of operations in all process routes of job i, i.e. $O_i = \cup_{p \in Pi} O_{ip}$

M_j set of alternative machines on which operation j can be processed,

Parameters

t_{ipjk} processing time of operation o_{ipj} on machine k

L large number

Decision variables

S_{ipjk} start time of operation o_{ipj} on machine k

C_{ipjk} completion time of operation o_{ipj} on machine k

C_i the completion time of job i

Z_{ip} = 1 if process route p of job i is selected and 0 otherwise

X_{ipjk} = 1 if machine k is selected for operation o_{ipj} and 0 otherwise

$Y_{iji,j,k}$ = 1 if operation j of job i precedes operation j of job i on machine k and 0 otherwise

Now, the mixed integer programming model is given below.

[P] Minimize $f(C_i, C_2, \ldots C_{|I|})$

subject to

$$\sum_{p \in P_i} Z_{ip} = 1$$

for all $i \in I$

$$\sum_{k \in M_j} X_{ipjk} = Z_{ip}$$

for all $i \in I, p \in P_i$, and $j \in O_{ip}$

$$S_{ipjk} + C_{ipjk} \le L \cdot X_{ipjk}$$

for all $i \in I, p \in P_i, j \in O_{ip}$, and $k \in M_j$

$$C_{ipjk} \ge S_{ipjk} + t_{ipjk} - L \cdot (1 - X_{ipjk})$$

for all $i \in I, p \in P_i, j \in O_{ip}$, and $k \in M_j$

$$\sum_{p \in P_i} S_{ipjk} \ge \sum_{p \in P_{i'}} C_{i'pj'k} - L \cdot Y_{iji'j'k}$$

for all $i, i' \in I, i<i', j \in O_{ip}, j' \in O_{i'p}$ and $k \in M_j \cap M_{j'}$

$$\sum_{p \in P_{i'}} S_{i'pj'k} \ge \sum_{p \in P_i} C_{ipjk} - L \cdot (1 - Y_{iji'j'k})$$

for all $i, i' \in I, i<i', j \in O_{ip}, j' \in O_{i'p}$, and $k \in M_j \cap M_j$

$$\sum_{k \in M_j} S_{ipjk} \ge \sum_{k \in M_j} C_{ip,j-1,k}$$

for all $i \in I, p \in P_i$, and $j \in O_{ip} \backslash \{o_{ipf(I,p)}\}$

$$C_i \ge \sum_{p \in P_i} \sum_{k \in M_j} C_{ip,o_{ipl(i,p)},k}$$

for all $i \in J$

$$S_{ipjk}, C_{ipjk}, C_i \ge 0$$

for all $I \in I, p \in P_i, j \in O_{ip}$, and $k \in M_j$

$$Z_{ip}, X_{ipjk} \in \{0,1\}$$

for all $I \in I, p \in P_i, j \in O_{ip}$, and $k \in M_j$

$$Y_{iji'j'k} \in \{0,1\}$$

for all $I, i' \in I, I < i', j \in O_{ip}, j' \in O_{i'p}$, and $k \in M_j \cap M_{j'}$

In the model, the objective function is generally represented as a function of job completion times, which includes makespan, total flow time, total tardiness, etc. More formally, each of the five objectives can be represented as follows.

- Makespan: $f_1(C_i, C_2, \ldots C_{|I|}) = \max_{i \in I}\{C_i\}$, where $|I|$ denotes the cardinality of set I.
- Mean flow time: $f_2(C_i, C_2, \ldots C_{|I|}) = \Sigma_{i \in I} C_i / |I|$

- Mean tardiness: $f_3(C_i, C_2, \ldots C_{|I|}) = \Sigma_{i \in I} T_i / |I|$, where $T_i = \max\{0, C_i - d_i\}$ (tardiness of job i) and d_i denotes the due-date of job i.

See Doh et al.[14] for more details on the implications of the constraints. As explained earlier, the dynamic version of the FJSP-MPP is the static one that relaxes the assumption that all jobs are ready for processing at time zero. In other words, the jobs arrive stochastically over time, i.e. nonzero and stochastic ready times.

3. Decision tree based scheduling approaches

3.1 Decision tree: overview

As a data-mining technique, a decision tree is a kind of classifier represented as a recursive partition of the instance space. More specifically, the decision tree is a rooted one that consists of non-leaf and leaf nodes, where non-leaf nodes represent a choice among alternatives, i.e. splitting the instance space into two or more subspaces according to a certain discrete function of the input attributes values, while leaf nodes represent classification or decision.

Before explaining the decision tree in details, an example data set is summarized in Table 1, adopted from Choi et al.[25] In the table, there are 12 objects, 4 conditional attributes and 1 decision attribute. For example, object 1 implies that the decision is 1 ($X = 1$) if the values of conditional attributes A, B, C and D are 1, 2, 2, and 1, respectively.

Using the data given in Table 1, a decision tree can be constructed as Fig. 2 that shows a path from the root node to each lead node corresponds to a decision. For example, if the values of conditional attributes A, B and C are 2, 3, and 1, the resulting decision is 1, i.e. $d = 1$.

3.2 Static approach

As explained earlier, the static approach uses the decision tree to select a priority rule combination to process the set of given jobs, i.e. no rule changes over the scheduling horizon, and hence it is useful for planning purposes. The static decision tree based mechanism suggested in this study is shown in Fig. 3.

As can be seen in the figure, the static mechanism consists of storage, construction and scheduling modules, each of which is explained below.

Table 1 Data set for decision tree: example

Objects	Conditional attributes				Decision attribute
	A	B	C	D	X
1	1	2	2	1	1
2	1	2	3	2	1
3	1	2	2	3	1
4	2	2	2	1	1
5	2	3	2	2	2
6	1	3	2	1	1
7	1	2	3	1	2
8	2	3	1	2	1
9	1	2	2	2	1
10	1	1	3	2	1
11	2	1	2	2	2
12	1	1	2	3	1

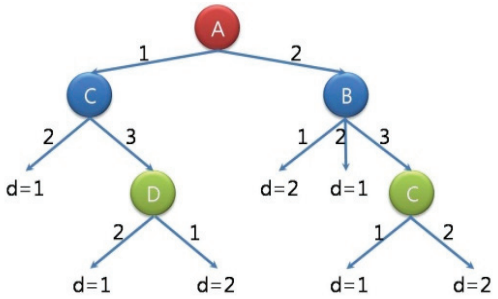


Fig. 2 Decision tree: example

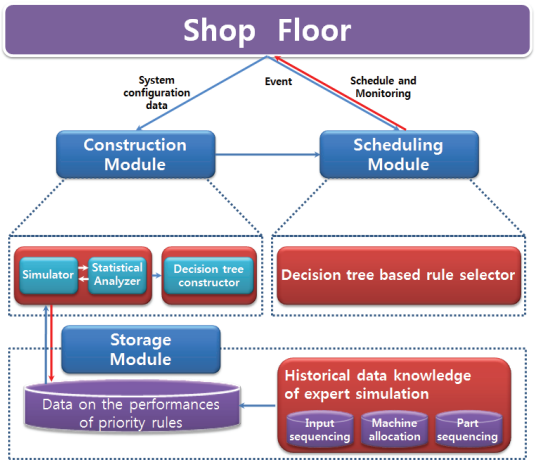


Fig. 3 Static decision tree based scheduling mechanism

3.2.1 Storage module

This module collects and stores the date required for construction and scheduling modules. The required data incudes system configuration data (machines, material

handling and storage facilities, pallets and fixtures), process plan data (part types, operation/machine pairs and processing times), simulation or historical data on the performances of priority rules and so on.

3.2.2 Construction module

This module constructs a decision tree. As can be seen in Fig. 3, the decision tree is built using simulator, statistical analyzer and decision tree constructor.

The simulator provides the performances of priority dispatching rules under given system states, which are used by the statistical analyzer that analyzes the data required to construct a decision tree. In our application, the conditional and decision attributes in the data set correspond to the system states and the selection of priority dispatching rule, respectively. If the simulation technique is used to construct a decision tree, an object in the data set, i.e., each row in Table 1, is obtained by performing a simulation run under a given system state and identifying the best dispatching rule.

Based on the data obtained from the simulator, the decision tree constructor builds up a decision tree to select a priority dispatching rule expected to perform the best under a specific system state. For this purpose, we suggest the following procedure.

Procedure 1 (Constructing a decision tree)**:**

Step 1: Specify the system attributes that may affect the system performances under consideration. For this purpose, various methods, e.g. knowledge of system experts, simulation, etc. can be used.

Step 2: Among the system attributes specified in Step 1, select eligible ones as follows.

(a) Calculate correlation coefficient (r_a) between attribute a and the performance measure under consideration, i.e.

$$r_a = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \cdot \sum (y_i - \bar{y})^2}},$$

where x_i and y_i denote the attribute and performance values of the ith simulation run (or historical data), respectively.

(b) Select the eligible ones using the correlation coefficients, i.e. those with $|r_a| > r_{min}$, where $r_{\min}$ is the minimum correlation coefficient for selecting the eligible ones (parameter).

Step 3: For each eligible attribute, determine number of levels as follows.

$$\left\lceil \left| r_a - \left(\sum_{k=1}^{n_F} r_k / n_F \right) \right| \times 10 \right\rceil,$$

where n_F is the number of eligible attributes. ($\lceil \bullet \rceil$ denotes the smallest integer greater than or equal to $\bullet$.)

Step 4: Obtain the data on the performances of priority dispatching rules under various system configurations, i.e. combinations of all the levels of eligible system attributes. For this purpose, simulation or historical data can be used.

Step 5: Construct a decision tree using the ID3 algorithm given below. (Among various algorithms to generate a decision tree, we use the ID3 algorithm since it is simple but proved to be effective.)

(a) Create the root node using the conditional attribute with the smallest entropy value, where the entropy function of conditional attribute j is defined as

$$entropy_j = \sum_{c=1}^{C_j} -p(w_{cj}|j) \cdot \log_2 p(w_{cj}|j),$$

where C_j and $p(w_{cj}|j)$ denote the number of different conditional attribute values and the proportion of value w_{cj} in conditional attribute j.

(b) Let the root node be the current node.

(c) For each conditional attribute value of the current node, create and connect a child node whose conditional attribute is set to the one with the smallest entropy value after updating the data set, i.e. entropy values are calculated after removing the conditional attribute of the current node and the objects with the conditional attribute value of the current node.

(d) If all the conditional attributes are considered, stop. Otherwise, let one of unconsidered child nodes be the current one and go to Step (c).

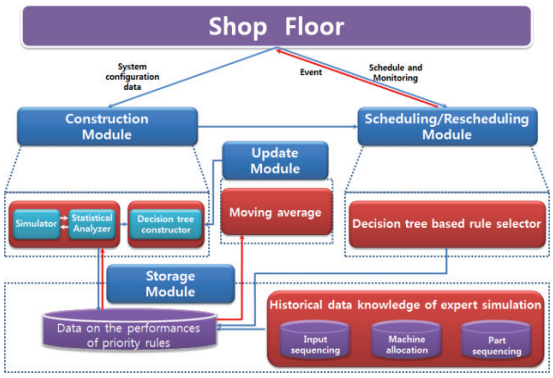


Fig. 4 Dynamic decision tree based scheduling mechanism

3.2.3 Scheduling module

In this module, a priority dispatching rule is selected using the decision tree based rule selector. For the static FJSP-MPP considered here, as stated earlier, the rule selector is used once at the beginning of the scheduling horizon.

3.3 Dynamic approach

Fig. 4 shows the dynamic decision tree based scheduling mechanism for the FJSP-MPP in which jobs arrive over time. In the dynamic mechanism, the decision tree is used to select an appropriate dispatching rule at the end of each monitoring period so that the computational burden required for carrying out simulation runs can be eliminated. Here, the monitoring period is the time period during which a priority dispatching rule is maintained before considering the rule change. To cope with the nature of the dynamic problem, the mechanism additionally needs a rescheduling strategy and a decision tree update method.

3.3.1 Storage module

This module collects and stores the date required for construction, scheduling/rescheduling and update modules. However, unlike the static one in which all the data used once at the beginning of scheduling horizon, the storage module in the dynamic mechanism maintains and updates the relevant data during the scheduling horizon.

3.3.2 Construction module

This module constructs an initial decision tree used to select the first priority dispatching rule at the beginning of the scheduling horizon. For this purpose, the decision tree construction procedure explained in the static mechanism is used. Recall that the decision tree is updated regularly, which will be explained later, In the dynamic mechanism.

3.3.3 Scheduling/rescheduling module

This module selects a priority rule at the point of time when a new one is needed due to system changes. For this purpose, we use the ALL strategy suggested by Jeong and Kim[27] since it is better than the others. In this strategy, the decision tree is used to select a new priority dispatching rule in the following cases.

- Beginning of the scheduling horizon
- Major system disturbances, e.g. machine breakdowns, urgent orders, etc.
- Minor system disturbances, e.g. tool breakages
- Getting the performances worse, e.g., certain performance value exceeds a pre-determined limit, at each monitoring period.

3.3.4 Update module

To improve the system performances further, the initial decision tree is updated at the beginning of each update period using the up-to-date data on the usage of the decision tree while operating the system. Here, the update period is the time period during which the current decision tree is used without change. In this study, we consider the case that the update period has an equal length, i.e. regular updates.

To update the decision trees, we suggest a moving average based method with three main steps: (a) selecting an attribute's level in the current decision tree; (b) dividing the selected level into smaller ones; and (c) constructing a new decision tree.

More specifically, in the first step, selected is the attribute's level with the highest usage frequency during the last u update periods, where the moving average method is used to calculate the usage frequency of each level, i.e. averaging the usage frequency of all attributes' levels from the recent u update periods. Then, the selected level is divided into v smaller ones with equal length. (In our case study, u and v were set to 10 and 10,

respectively.) Here, the new data on the performances of priority rules under each of the divided level is obtained using simulation. Finally, a new decision tree is constructed using the new and existing data.

4. Case study

To show the performances of the two decision tree based mechanisms proposed in this study, they were applied to a flexible job shop case, and the results are reported in this section.

4.1 System description

The flexible job shop considered in this study consists of reconfigurable manufacturing cells (RMCs) and a conventional job shop.

The RMC consists of numerical control machines, a loading/unloading station and a central buffer. Each numerical control machine has an automatic tool changer and a tool magazine with limited capacities. A part can be fed into the RMC through the loading/unloading station after it is clamped onto a pallet with a required fixture type. Note that common pallets are used in the RMC, i.e. any fixture types can be mounted on the pallet. It is assumed that the system has enough fixtures to clamp parts on pallets. Also, it is assumed that the fixture allocation is done in advance, i.e. the set of common pallets is divided into mutually exclusive subsets, where the pallets are equipped with a predetermined fixture type. The central buffer is used to store in-process parts within the RMC. After released into the RMC, a part with a required fixture type on a pallet goes into the central buffer and waits for processing. Each part stored in the central buffer is sent to the machines for operations. After the required operations are finished, the part leaves the system through the loading/unloading station and removed from the pallet together with the fixture. Also, the job shop is a conventional legacy system that consists of dedicated and flexible machines, such as numerical control machines, cleaning machines. Table 2 summarizes the components of the real system considered in this study.

When the RMCs are introduced to a conventional job shop, the resulting system becomes a flexible job shop in which the RMCs can be utilized as an alternative

Table 2 Components of the flexible job shop

	Machine Code	Description
RMC 1	RVMC1A	Vertical Machining Center (RMC1)
	RVMC1B	Vertical Machining Center (RMC1)
	RVMC1C	Vertical Machining Center (RMC1)
	RLU	Loading/Unloading station
RMC 2	RVMC2A	Vertical Machining Center (RMC2)
	RVMC2B	Vertical Machining Center (RMC2)
	RVMC2C	Vertical Machining Center (RMC2)
	RLU	Loading/Unloading station
RMC 3	RHMC1A	Horizontal Machining Center (RMC3)
	RHMC1B	Horizontal Machining Center (RMC3)
	RHMC1C	Horizontal Machining Center (RMC3)
	RLU	Loading/Unloading station
RMC 4	RHMC2A	Horizontal Machining Center (RMC4)
	RHMC2B	Horizontal Machining Center (RMC4)
	RHMC2C	Horizontal Machining Center (RMC4)
	RLU	Loading / Unloading station
Job shop (legacy)	MK1	Marking Machine 1
	MK2	Marking Machine 2
	VMC1	Vertical Machining Center 1
	VMC2	Vertical Machining Center 2
	HMC1	Horizontal Machining Center 1
	HMC2	Horizontal Machining Center 2
	HMC3	Horizontal Machining Center 3
	CFM	Cubic Face Milling Machine
	HFMCMM	Horizontal Face Milling/Cutting & Measuring Machine
	GDM	Grinding Machine
	DRGD	Drilling & Grinding Machine
	INS	Inspection
	CLM	Cleaning

Table 3 Operations and their available machines

Operations	Description	Available machines
OMK	Marking	MK1, MK2
OVR	Vertical Removing	RVMC1A, RVMC1B, RVMC1C RVMC2A, RVMC2B, RVMC2C VMC1, VMC2, CFM, GDM
OHR	Horizontal Removing	RHMC1A, RHMC1B, RHMC1C RHMC2A, RHMC2B, HMC2C HMC1, HMC2, CFM, FMCMM
OCR	Cubic Removing	CFM
OFC	Face Cutting	RVMC1A, RVMC1B, RVMC1C, RVMC2A, RVMC2B, RVMC2C
OHFM	Horizontal Face Milling	RHMC1A, RHMC1B, RHMC1C, RHMC2A, RHMC2B, MK1
OCFM	Cubic Face Milling	CFM
OINS	Inspection	HFMCMM, INS
OHFMCM	Horizontal Face Milling/Cutting & Measuring	HFMCMM
OGM	Grinding Mark	MK2
ODR	Drilling	VMC1, VMC2
OGD	Grinding	GDM, DRGD
ODRGD	Drilling & Grinding	DRGD

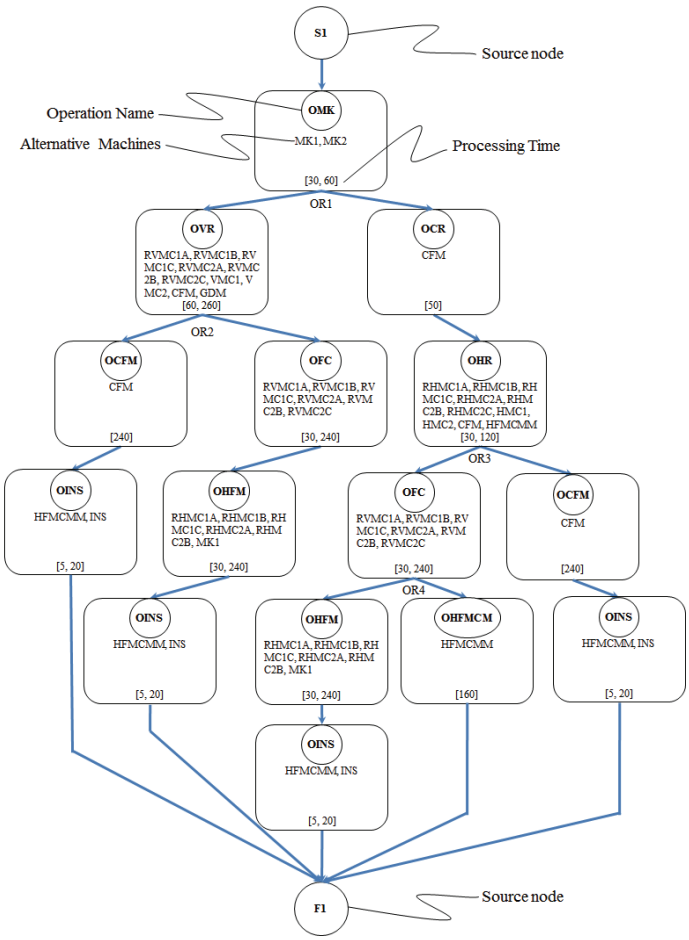


Fig. 5 Multiple process plan for an example part type

processor that can replace the conventional job shop. In other words, the operations can be done on either the RMCs or the job shop. It is assumed that part types to be produced during the upcoming period are given in advance and all the operations required for producing the part types, together with the available machines on which their operations can be processed, are summarized in Table 3.

As explained earlier, each part is processed according to a multiple process plan. Fig. 5 shows an example of the network model for a part type. In this figure, we can see that there are five paths from the source to the sink node, e.g. S1–OMK–OVR–OCFM–OINS–F1, where the operation OMK can be processed by either MK1 or MK2 whose processing times are 30 and 60, respectively. Since the pilot RMC was developed, we could not obtain the real data on part types. Instead, we generated the part data based on the experiences of the project partners.

4.2 Test results: static system

To test the performance of the static mechanism, the first test was done on the system in which all parts are given in advance at the beginning of scheduling horizon.

In the test, the minimum correlation coefficient r_{min} for selecting the eligible attributes was set to 0.3, i.e. selected are the eligible ones with $|r_a| > 0.3$, and the resulting decision tree under the total tardiness measure is shown in Fig. 6. As can be seen in the figure, the selected eligible attributes are: number of part types, production quantity and processing time.

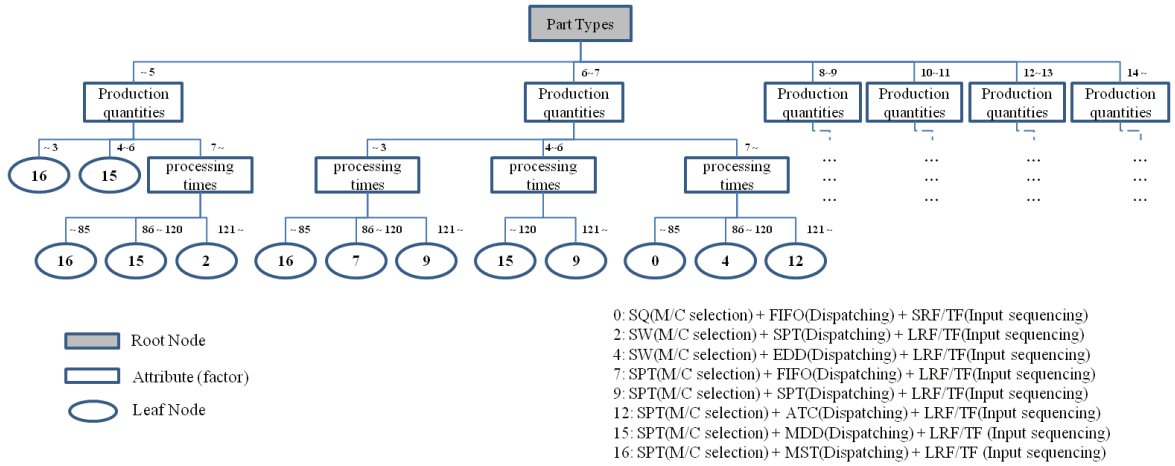


Fig. 6 Static decision tree under the total tardiness measure

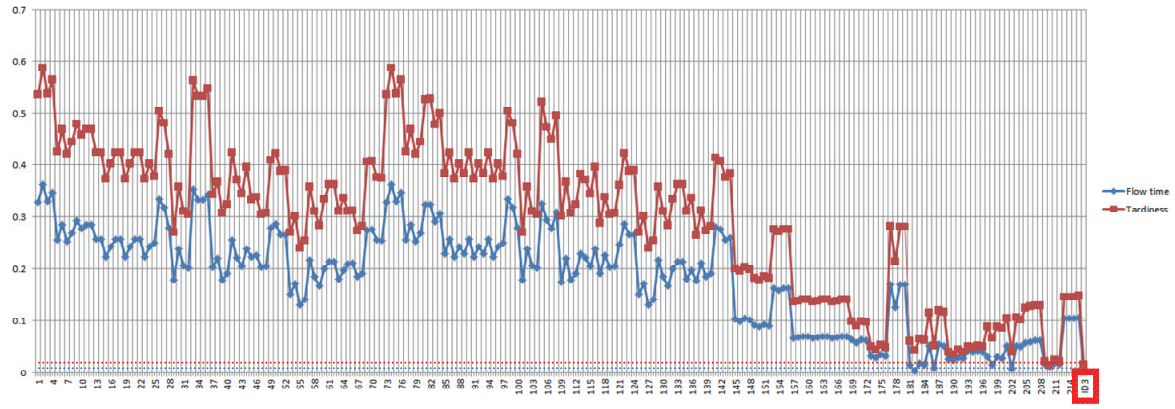


Fig. 7 Test results: static case

The decision tree based mechanism was compared with 216 priority rule combinations under the total flow time and the total tardiness measures, where the rule combinations are those with 4 input sequencing rules to RMCs (SPPT, LPPT, SRF/TF and LRF/TF), 3 operation/machine selection rules (SQ, SW and SP) and 18 part sequencing rules (FIFO, SOPT, WINQ, LWKR, LOPR, SJPT, EDD, CR, ATC, COVERT, MDD, MST, P-FIFO, P-SOPT, P-WINQ, P-LWKR, P-LOPR and P-SJPT). See Doh et al.[14] and Yu et al.[15] for the detailed descriptions of the priority rules.

For evaluation of the results, we use the relative performance ratio because we could not obtain the optimal solutions. Here, the relative performance ratio for a test instance is defined as

$$100 \cdot (Z - Z_{best}) / Z_{best},$$

where Z is the objective value (total flow time and total tardiness) obtained using a rule combination or the static decision tree based mechanism for the instance and Z_{best} is the best objective value among those obtained from the 216 rule combinations and the mechanism.

For the test, we generated 5000 instances in which the number of part types and the production quantities were generated from $DU(5, 14)$ and $DU(1, 8)$, where $DU(a, b)$ denotes the discrete uniform distribution with a range $[a, b]$. For each part type, the number of operations and the number of alternative operation/machine pairs were generated from $DU(5, 14)$ and $DU(1, 8)$. Also, the multiple process plans were generated randomly, so that

various process routing configurations can be considered. Finally, the processing times were generated from *DU*(50, 149). Finally, the central buffer capacity is 36 according to the pilot RMC developed in our research project and the number of fixtures was generated from *DU*(1, 4).

Test results are summarized in Fig. 7 that shows the average relative performance ratios of the priority rule combinations and the decision tree based mechanism. In the figure, the *x*- and *y*-axis represent the methods and the relative performance ratios, respectively. As can be seen in the figure, the decision tree based mechanism (ID3) is competitive to the best rule combinations in overall average and particularly, gave stable performance, i.e. less variable performances.

4.3 Test results: dynamic system

Two dynamic decision tree based mechanism mechanisms were compared in this study, i.e. those without and with updating the decision tree. In the dynamic mechanism without updating the decision tree, the decision tree was constructed using procedure 1 (static case), and it remains the same over the scheduling horizon. Also, the rescheduling decisions were done at major system disturbances including machine breakdowns and urgent orders. One the other hand, the dynamic mechanism that updates the decision tree constructs the initial decision tree using procedure 1and updates it at pre-specified update period of the same length using the moving average based method explained earlier. As in the non-update mechanism, the rescheduling decisions were done at major system disturbances.

In this test, the basic data, such as number of part types, production quantities, number of operations, number of alternative operation/machine pairs, multiple process plans and processing times, were generated using the same method as in the static test. Also, according to the dynamic nature of the system, the parts arrive the system with exp(60), where exp(λ) denotes the exponential distribution with mean λ. Also, MTTF (mean time to failure) and MTTR (mean time to repair) were generated from exp(10000) and expo(40). Finally, the urgent order with critically-short due date was generated for each part type with probability 0.1.

Test results are summarized in Table 4 that shows the amounts of improvement that the two dynamic mechanisms gave over the best rule combination (LRF/TF-SW-LWKR) for throughput, mean flow time and mean tardiness measures, respectively. It can be seen from the table that both mechanisms improves the best rule combination significantly due to their capability to change the rule combinations according to system states. In fact, the amounts of improvement over the best rule combination were 8.63%, 8.22% and 5.47% (without updates) and 12.54%, 13.68% and 7.31% (with updates) for throughput, total flow time and total tardiness, respectively. Also, of the two mechanisms, the updating mechanism outperforms the non-updating one, which shows the effectiveness of the decision tree update module proposed in this study. Finally, computation times are not reported here since they were very short, i.e. less than 1 second.

Table 4 Test results on the dynamic system

	Mechanism without updates	Mechanism with updates
Throughput	8.63*	12.54
Total flow time	8.22	13.68
Total tardiness	5.47	7.31

* amount of improvement over the best rule combination (LRF/TF-SW-LWKR)

5. Concluding remarks

This study suggested a decision tree based scheduling approach for flexible job shops with multiple process plans in which each job can be processed through alternative operations that can be processed on alternative machines. The main decision variables are selecting operation/machine pair and sequencing the jobs assigned to each machine. The decision tree is used to select a priority rule appropriate for a specific system state and hence the burdens required for carrying out simulation runs when scheduling decision is done can be eliminated.

Two decision tree based scheduling mechanisms were suggested i.e. static one for planning purposes and dynamic one for real-time scheduling. In the static case, all jobs are given in advance and the decision tree is used to select a priority dispatching rule to process all the jobs, i.e. no rule changes over the scheduling horizon. Also, in the dynamic case, the jobs arrive over time and the

decision tree is updated regularly and used to select a priority rule in real-time according to a rescheduling strategy, e.g. machine breakdowns, urgent orders, etc. To show the performances of the two mechanisms, they were applied to static and dynamic flexible job shops with reconfigurable manufacturing cells and a conventional job shop, and the results showed that both mechanisms improves the simple priority rule based scheduling approach for throughput, flow time and tardiness measures. In particular, the dynamic mechanism that updates the decision tree gave additional improvement over the non-updating one.

This study can be extended in several directions. First, it is needed to develop more sophisticated methods to construct and update the decision tree, especially in the dynamic mechanism. Second, patterns and knowledge learned by data mining techniques are not always usable. How to ascertain the usable knowledge in a large amount rules and patterns deserves the attention. Finally, more case studies on other system configurations are worth to be performed.

ACKNOWLEDGEMENTS

This work was supported by the Ministry of Trade, Industry and Energy (MOTIE) grant funded by Korea government (Grant Code: 10033895-2009-11). This is gratefully acknowledged.

REFERENCES

1. Lenstra, J. K., Rinnooy Kan, A. H. G., and Brucker, P., “Complexity of Machine Scheduling Problems,” Annals of Discrete Mathematics, Vol. 1, pp. 343-362, 1977.
2. Garey, M. R., Johnson, D. S., and Sethi, R., “The Complexity of Flowshop and Jobshop Scheduling,” Mathematics of Operations Research, Vol. 1, No. 2, pp. 117-129, 1976.
3. Jain, A. S. and Meeran, S., “Deterministic Job-shop Scheduling: Past, Present and Future,” European Journal of Operations Research, Vol. 113, No. 2, pp. 390-434, 1999.
4. Koren, Y., Heisel, U., Jovane, F., Moriwaki, T., Pritschow, G., et al., “Reconfigurable Manufacturing Systems,” Annals of the CIRP, Vol. 48, No. 2, pp. 527-540, 1999.
5. Iwata, K., Murotsu, Y., Oba, F., and Okamura, K., “Solution of Large-scale Scheduling Problems for Job-Shop Type Machining Systems with Alternative Machine Tools,” Annals of the CIRP, Vol. 29, No. 1, pp. 335-338, 1980.
6. Nasr, N. and Elsayed, E. A., “Job Shop Scheduling with Alternative Machines,” International Journal of Production Research, Vol. 28, No. 9, pp. 1595-1609, 1990.
7. Lee, D.-H. and Kim, Y.-D., “Scheduling Algorithms for Flexible Manufacturing Systems with Partially Grouped Machines,” Journal of Manufacturing Sytems, Vol. 18, No. 4, pp. 301-309, 1999.
8. Low, C. and Wu, T.-H., “Mathematical Modeling and Heuristic Approaches to Operation Scheduling Problems in an FMS Environment,” International Journal of Production Research, Vol. 39, No. 4, pp. 689-708, 2001.
9. Lee, Y.-H., Jeong, C.-S., and Moon, C., “Advanced Planning and Scheduling with Outsourcing in Manufacturing Supply Chain,” Computers & Industrial Engineering, Vol. 43, No. 1, pp. 351-374, 2002.
10. Kim, Y.-K., Park, K., and Ko, J., “A Symbiotic Evolutionary Algorithm for the Integration of Process Planning and Job Shop Scheduling,” Computers & Operations Research, Vol. 30, No. 8, pp. 1151-1171, 2003.
11. Ozguven, C., Ozbakir, L., and Yavuz, Y., “Mathematcal Models for Job-shop Scheduling Problems with Routing and Process Plan Flexibility,” Applied Mathematical Modelling, Vol. 34, No. 6, pp. 1539-1548, 2010.
12. Baykasoglu, A., “Linguistic-based Meta-heuristic Optimization Model for Flexible Job-shop Scheduing,” International Journal of Production Research, Vol. 40, No. 17, pp. 4523-4543, 2002.
13. Baykasoglu, A., Özbakır, L., and Sönmez, A. I., “Using Multiple Objective Tabu Search and Grammars to Model and Solve Multi-objective Flexible Job-shop Scheduling Problems,” Journal of Intelligent Manufacturing, Vol. 15, No. 6, pp. 777-785, 2004.

14. Doh, H.-H., Yu, J.-M., Kim, J.-S., Lee, D.-H., and Nam, S.-H., “A Priority Scheduling Approach for Flexible Job Shops with Multiple Process Plans,” International Journal of Production Research, Vol. 51, No. 12, pp. 3748-3764, 2013.
15. Yu, J.-M., Doh, H.-H., Kim, J.-S., Kwon, Y.-J., Lee, D.-H., and Nam, S.-H., “Input Sequencing and Scheduling for a Reconfigurable Manufacturing System with a Limited Number of Fixtures,” International Journal of Advanced Manufacturing Technology, Vol. 67, No. 1-4, pp. 157-169, 2013.
16. Deng, H., Runger, G., and Tuv, E., “Bias of Importance Measures for Multi-valued Attributes and Solutions,” Proc. of the 21st International Conference on Artificial Neural Networks, pp. 293-300, 2011.
17. Shinichi, N. and Taketoshi, Y., “Dynamic Scheduling System Utilizing Matching Learning as a Knowledge Acquisition Tool,” International Journal of Production Research, Vol. 30, No. 2, pp. 411-431, 1992.
18. Shaw, M. J., Park, S., and Raman, N., “Intelligent Scheduling with Machine Learning Capabilities: the Induction of Scheduling Knowledge,” IIE Transactions, Vol. 24, No. 2, pp. 156-168, 1992.
19. Piramuthu, S., Raman, N., and Shaw, M. J., “Learning-based Scheduling in a Flexible Manufacturing Flow Line,” IEEE Transactions on Engineering Management, Vol. 41, No. 2, pp. 172-182, 1994.
20. Park, S.-C., Raman, N., and Shaw, M. J., “Adaptive Scheduling in Dynamic Flexible Manufacturing Systems: a Dynamic Rule Selection Approach,” IEEE Transactions on Robotics and Automation, Vol. 13, No. 4, pp. 486-502, 1997.
21. Lee, C.-Y., Piramuthu, S., and Tsai, Y.-K., “Job Shop Scheduling with a Genetic Algorithm and Machine Learning,” International Journal of Production Research, Vol. 35, No. 4, pp. 1171-1191, 1997.
22. Arzi, Y. and Iaroslavitz, L., “Operating an FMC by a Decision-tree-based Adaptive Production Control System,” International Journal of Production Research, Vol. 38, No. 3, pp. 675-697, 2000.
23. Su, C. T. and Shiue, Y. R., “Intelligent Scheduling Controller for Shop Floor Control Systems: a Hybrid Genetic Algorithm/Decision Tree Learning Approach,” International Journal of Production Research, Vol. 41, No. 12, pp. 2619-2641, 2003.
24. Kwak, C. and Yih, Y. “Data Mining Approach to Production Control in the Computer Integrated Testing Cell,” IEEE Transactions on Robotics and Automation, Vol. 20, No. 1, pp. 107-116, 2004.
25. Choi, H.-S., Kim, J.-S., and Lee, D.-H., “Real-time Scheduling for Reentrant Hybrid Flow Shops: a Decision Tree Based Mechanism and its Application to a TFT-LCD Line,” Expert Systems with Applications, Vol. 38, No. 4, pp. 3514-3521, 2011.
26. Ho, Y. C. and Moodie, C. L., “Solving Cell Formation Problems in a Manufacturing Environment with Flexible Processing and Routing Capabilities,” International Journal of Production Research, Vol. 34, No. 10, pp. 2901-2923, 1996.
27. Jeong, K.-C. and Kim, Y.-D., “A Real-time Scheduling Mechanism for a Flexible Manufacturing System: using Simulation and Dispatching Rules,” International Journal of Production Research, Vol. 36, No. 9, pp. 2609-2626, 1998.

이 논문은 International Science Council의 주관으로 2014년 싱가포르에서 개최된 Manufacturing, Information and Industrial Engineering 학술대회(ICMIIE)에서 발표된 논문 “Decision Tree based Scheduling for Flexible job Shops with Multiple Process Plans”의 개정증보판 입니다.

This paper is an improved version of the conference paper entitled “Decision Tree based Scheduling for flexible Job Shops with Multiple Process Plans” presented at the International Conference on Manufacturing, Information and Industrial Engineering (ICMIIE), which was held in Singapore, 2014 and organized by the International Science Council.