Decision Tree Based Scheduling for Flexible Job Shops with Multiple Process Plans


Abstract—This paper suggests a decision tree based approach for flexible job shop scheduling with multiple process plans, i.e., each job can be processed through alternative operations, each of which can be processed on alternative machines. The main decision variables are: (a) selecting operation/machine pair; and (b) sequencing the jobs assigned to each machine. As an extension of the priority scheduling approach that selects the best priority rule combination after many simulation runs, this study suggests a decision tree based approach in which a decision tree is used to select a priority rule combination adequate for a specific system state and hence the burdens required for developing simulation models and carrying out simulation runs can be eliminated. The decision tree based scheduling approach consists of construction and scheduling modules. In the construction module, a decision tree is constructed using a four-stage algorithm, and in the scheduling module, a priority rule combination is selected using the decision tree. To show the performance of the decision tree based approach suggested in this study, a case study was done on a flexible job shop with reconfigurable manufacturing cells and a conventional job shop, and the results are reported by comparing it with individual priority rule combinations for the objectives of minimizing total flow time and total tardiness.

Keywords—Flexible job shop scheduling, Decision tree, Priority rules, Case study.

I. INTRODUCTION

This study considers the scheduling problem in flexible job shops with multiple process plans, i.e. each job can be processed through alternative operations, each of which can be processed on alternative machines. The problem, which is an extension of the conventional job shop scheduling problem, can be found in various types of manufacturing systems, especially in flexible or reconfigurable manufacturing systems since each operation can be processed on one or more machines. Here, the reconfigurable manufacturing system is the one designed at the outset 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. In case that one or more reconfigurable manufacturing cells are introduced to a conventional job shop system for the purpose of increasing system capacity and flexibility, the resulting hybrid system becomes a flexible job shop.

Since the flexible job shop scheduling problem with multiple process plans is very complicated, the previous studies suggest meta-heuristics or report the performances of priority rule combinations. Here, the priority rule combinations are required since the scheduling problem has two main decisions: (a) selecting operation/machine pairs; and (b) sequencing the jobs assigned to each machine. See Ozguven et al. [1] and Doh et al. [2] for recent studies on flexible job shop scheduling problem with multiple process plans. Although there are some merits, the meta-heuristic and the priority scheduling approaches have disadvantages. First, the meta-heuristic approach requires too much computation time to be used in real-time environment. Also, the priority scheduling approach has the burdens required for developing simulation models and carrying out simulation runs to select the best rule for a specific system state.

In this study, we suggest a decision tree based approach for the flexible job shop scheduling problem in which the decision tree is used to select a priority rule combination appropriate for a specific system state and hence the burdens required for developing simulation models and carrying out simulation runs can be eliminated. 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 for flexible job shop scheduling since it has several advantages, i.e. simple to understand and interpret, containing value even with little hard data, adding possible scenarios, etc. See Deng et al. [3] for more advantages of the decision tree.

There are several previous studies on production scheduling based on the decision tree. One of earlier studies is Shinichi and Taketoshi [4] that suggest a learning algorithm that generates a decision tree using the empirical data obtained by simulation runs, where the decision tree is used to decide the priority rule at decision points during the actual production operations. Also, Shaw et al. [5] suggest a scheduling framework to classify manufacturing patterns and to generate a decision tree that dynamically selects the priority rule for a given set of system attributes, and later, Piramuthu et al. [6] and Park et al. [7] applied the framework to flexible manufacturing systems after certain modifications. See Lee et al. [8], Arzi and Iaroslavitz [9], Su and Shiue [10], Kwak and Yih [11], Priore et al. [12], Shiue et al. [13] and Choi et al. [14] for other decision tree based scheduling approaches and applications.

As stated earlier, this study suggests a decision tree based scheduling approach for flexible job shops, especially with multiple process plans. In this approach, the decision tree is...
used to select a priority rule combination adequate for a specific system state and hence the burdens required for developing simulation models and carrying out simulation runs of the priority scheduling approach can be eliminated. The decision tree based scheduling approach suggested in this study consists of two main steps: construction and scheduling steps. In the construction step, a decision tree is constructed by a systematic procedure using the empirical data obtained by simulation runs, and in the scheduling module, a priority rule combination is selected under a specific system state using the decision tree. To show the performance of the decision tree based approach, a case study was done on a flexible job shop with reconfigurable manufacturing cells and a conventional job shop for each of the objectives of minimizing total flow time and total tardiness, and the results are reported by comparing it with individual priority rule combinations.

The paper is organized as follows. In the next section, the problem is described in more details. The decision tree based scheduling approach is explained in the third section, and the case study is reported in the fourth section. The final section concludes the paper with a summary and discussion of future research.

II. PROBLEM DESCRIPTION

The flexible job shop scheduling problem considered in this study can be briefly explained as follows. For a given set of jobs, the problem is to determine the operation and machine pairs of each job and the sequence of the jobs assigned to each machine according to the process routes.

As explained earlier, each job is processed according to a multiple process plan that specifies alternative operations, their sequence, and alternative machines on which an operation is to be processed. In other words, each job can be processed through alternative operations, each of which can be processed on alternative machines. The decision variables are: (a) process routing of each job, i.e. operation/machine selections; and (b) sequence of the jobs assigned to each machine, i.e. job shop scheduling. The two objectives are considered in this study. They are minimizing total flow time and total tardiness. Note that each objective is a function of job completion times.

This study considers a static and deterministic version of the problem. In other words, all jobs are ready for processing at time zero, i.e. zero ready times, and 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, i.e. once a job is processed on a machine, it will stay on that machine until its completion; (d) transportation times among machines are ignorable or can be included in processing times; and (e) due dates are not enforced as hard constraints. See Doh et al. [2] for more details.

III. DECISION TREE BASED SCHEDULING APPROACH

A. Decision Tree

As a data-mining technique, a decision tree is a kind of classifier represented as a recursive partition of the instance space. More specifically, a 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 sub-spaces 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 I. In the table, there are twelve objects, four conditional attributes and one 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 I, various decision trees can be constructed. Fig. 1 shows an example in which a path from the root node to each lead node corresponds to a decision. For example, if the values of conditional attributes A, B, C and D are 2, 3, and 1, the resulting decision is 1, i.e., d = 1.

B. Decision Tree Based Scheduling

Fig. 2 shows the decision tree based scheduling framework suggested in this study. As can be seen in the figure, the framework consists of storage, construction and scheduling, each of which is explained below.

---

**TABLE I**

<table>
<thead>
<tr>
<th>Objects</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>8</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>9</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Fig. 1 Decision tree: example**
provides the performances of priority dispatching rules under performance measure, respectively.

For this purpose, some priority rules expected to perform the best under a specific performance under consideration, are selected. For this purpose, the conditional and decision attributes in the data set, i.e., each row in Table I, is obtained by performing a simulation run under a given system state and identifying the data set. In our application, the conditional and decision attributes that analyzes the data associated with constructing a decision tree. The simulator consists of four stages, each of which is explained below (The method is a modified version of Kwon et al. [15]).

1) Construction Module

In this module, a decision tree is built up using simulator, statistical analyzer and decision tree constructor. The simulator provides the performances of priority dispatching rules under various system states, which are used by the statistical analyzer that analyzes the data associated with constructing 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 I, is obtained by performing a simulation run under a given system state and identifying the best dispatching rule.

Based on the data obtained from simulation, the decision tree constructor builds up a decision tree that will be used to select a priority rule expected to perform the best under a specific system state. For this purpose, we suggest an algorithm that consists of four stages, each of which is explained below (The method is a modified version of Kwon et al. [15]).

Stage 1. Specifying the Attributes

In this stage, the system attributes are specified that may affect the system performance measure under consideration. For this purpose, various methods, e.g. knowledge of system experts, simulation, etc., can be used.

Stage 2. Selecting the Eligible Attributes

Among the specified system attributes in stage 1, the eligible ones, which are expected to be highly influential to the system performance under consideration, are selected. For this purpose, we suggest the correlation coefficient based method.

The detailed step-by-step procedure is given below.

Step 1. Calculate the correlation coefficient value \( r_i \) between an attribute and the performance measure under consideration, i.e.

\[
 r_i = \frac{\sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n} (x_i - \bar{x})^2 \sum_{i=1}^{n} (y_i - \bar{y})^2}},
\]

where \( x_i \) and \( y_i \) denote the \( i \)th values of attribute and performance measure, respectively.

Step 2. Classify the attributes as follows.

<table>
<thead>
<tr>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
</tbody>
</table>

Group 1: Attributes with \( 0.7 < r_i \leq 1.0 \)
Group 2: Attributes with \( 0.3 < r_i \leq 0.7 \)
Group 3: Attributes with \( 0.1 < r_i \leq 0.3 \)
Group 4: Attributes with \( 0 \leq r_i \leq 0.1 \)

Step 3. Determine the eligible attributes by selecting some higher ranked groups.

Stage 3. Determining the Number of Levels for Each Attribute

In this stage, the number of levels for each eligible attribute is determined using its correlation coefficient value. More formally, calculate the number of levels as

\[
 \lceil \log_{10}\left( \sum_{k=1}^{F} \frac{n_F}{n_E} \right) \rceil,
\]

where \( n_E \) denotes the total number of eligible attributes. Also, \( \lceil \cdot \rceil \) denotes the smallest integer greater than or equal to \( \cdot \).

Stage 4. Constructing a Decision Tree

In this stage, a decision tree is constructed using the data obtained from the simulation results on the performances of candidate priority rules under various system configurations, i.e. combinations of all the levels of eligible system attributes. Among various algorithms to generate a decision tree, we use the ID3 algorithm since it is simple but proved to be effective. See Quinlan [16] for more details of the ID3 algorithm.

The ID3 algorithm is based on the entropy function to select the conditional attributes of a decision tree, where the entropy function of conditional attribute \( j \) is defined as

\[
 entropy_j = \sum_{c=1}^{C_j} - p(w_{cj}) \log_2 p(w_{cj}),
\]

where \( C_j \) denotes the number of different conditional attribute values, e.g., \( C_A, C_B, C_C, \) and \( C_D \) are, 2, 3, 3, and 3 for the example in Table I. Also, \( p(w_{cj}) \) denotes the proportion of value \( w_{cj} \) in conditional attribute \( c \). For example, \( p(1|A) = 8/12 \) and \( p(2|B) = 4/12 \) in Table I.

The detailed procedure of the ID3 algorithm is given below.

Step 1. Create the root node using the conditional attribute with the smallest entropy value and let the root node be the current node.

Step 2. For each conditional attribute value of the current node, create and connect a child node whose conditional attribute value 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.

Step 3. If all conditional attributes are considered, stop. Otherwise, let one of the unconsidered child nodes be the current node and go to Step 2.
2) Scheduling Module

In this module, a priority rule is selected using the decision tree. The decision tree can be used in two ways. First, for the static flexible job shop scheduling problem considered in this study, the decision tree is used once at the beginning of scheduling period. On the other hand, if the decision tree is used for dynamic scheduling, the decision tree can be used to select a priority rule when the current rule needs to be changed due to system changes. Here, there may be various rescheduling strategies that determine the points of time when the current priority rule is to be changed.

3) Storage Module

This module collects and stores the data required for the construction and scheduling modules, e.g., simulation results or historical data on the performances of priority rules in order to construct the decision tree or perform rescheduling.

IV. COMPUTATIONAL EXPERIMENTS

The performance of the decision tree based scheduling approach was tested using the data on a flexible job shop case, and the results are reported in this section.

As explained earlier, the flexible job shop considered in this study consists of reconfigurable manufacturing cells (RMCs) and a conventional job shop. The RMC, a state-of-the-art manufacturing technology that overcomes the limitations of flexible manufacturing systems, is the one designed for rapid changes in its hardware and software components to quickly adjust its production capacity and functionality. See Koren et al. [17] for more details on reconfigurable manufacturing systems. Also, the job shop is a conventional legacy system that consists of dedicated and flexible machines, such as marking machines, numerical control machines, cleaning machines, etc. Note that when RMCs are introduced to a conventional job shop, the resulting hybrid system becomes a type of flexible job shop. Here, an RMC can be utilized as an alternative processor that can replace the conventional job shop. Therefore, the hybrid system can be considered as a parallel system in which the operations can be done on either the RMC or the job shop. However, the two systems are different in operations and processing times even for the same part type.

The RMC consists of numerical control (NC) machines, a loading/unloading (L/U) station and a central buffer. Each 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 a pallet. Also, a fixture type can be used for a specific set of part types. One or more tools are required to perform an operation on a part type, and each tool requires one or more slots in the tool magazine. The central buffer which is an automatic storage and retrieval system (AS/RS) is used to store in-process parts within the RMC. Since the RMC has a limited central buffer, an upper limit is imposed on the number of parts circulating in the system. 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 L/U station and removed from the pallet together with the fixture. Table II summarizes the system components.

<table>
<thead>
<tr>
<th>TABLE II SYSTEM COMPONENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>M/C Code</td>
</tr>
<tr>
<td>---------------------</td>
</tr>
<tr>
<td>RVMC1A</td>
</tr>
<tr>
<td>RVMC1B</td>
</tr>
<tr>
<td>RVMC1C</td>
</tr>
<tr>
<td>RLMU</td>
</tr>
<tr>
<td>RVMC2A</td>
</tr>
<tr>
<td>RVMC2B</td>
</tr>
<tr>
<td>RVMC2C</td>
</tr>
<tr>
<td>RLHMC1A</td>
</tr>
<tr>
<td>RLHMC1B</td>
</tr>
<tr>
<td>RLHMC1C</td>
</tr>
<tr>
<td>RLHMC2A</td>
</tr>
<tr>
<td>RLHMC2B</td>
</tr>
<tr>
<td>RLHMC2C</td>
</tr>
<tr>
<td>MK1</td>
</tr>
<tr>
<td>MK2</td>
</tr>
<tr>
<td>VM1C</td>
</tr>
<tr>
<td>VM2C</td>
</tr>
<tr>
<td>HC1M</td>
</tr>
<tr>
<td>HC2M</td>
</tr>
<tr>
<td>HC3M</td>
</tr>
<tr>
<td>CFM</td>
</tr>
<tr>
<td>HFMMC</td>
</tr>
<tr>
<td>GDM</td>
</tr>
<tr>
<td>DRGD</td>
</tr>
<tr>
<td>INS</td>
</tr>
<tr>
<td>CLM</td>
</tr>
</tbody>
</table>

It is assumed that part types and their quantities to be produced during the upcoming period are given in advance, and each part type is produced requires a predetermined set of operations. Table III lists all the operations, together with the machines that each operation can be processed.
plan with 4 OR relations and 15 intermediate nodes. In this
problem. Also, it is assumed that fixture allocation is done in
advance, i.e., the given set of common pallets is divided into
mutually exclusive subsets, within which the pallets are
equipped with a predetermined fixture type. Finally, we assume
that the number of fixtures is enough to clamp parts on pallets.
As explained earlier, each job is processed according to a
multiple process plan, where the multiple process plan can be
represented as a network that consists of three types of nodes:
source, intermediate and sink. The source and sink are dummy
nodes that represent the start and the end of a part processing,
respectively. Each intermediate node represents alternative
machines and operations, together with the corresponding
processing times. Also, an arc connecting two nodes represents
the precedence relation between the corresponding two
operations. In particular, if a part meets an OR relation, it must
select one of the corresponding alternative operation/machine
pairs. In summary, a part is completed through a path (set of
alternative operation/machine pairs). See Kim et al. [18] for more
details on the loading plans.

Table III

<table>
<thead>
<tr>
<th>Operations</th>
<th>Description</th>
<th>Available machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMK</td>
<td>Marking</td>
<td>MK1, MK2</td>
</tr>
<tr>
<td>OVR</td>
<td>Vertical Removing</td>
<td>RVMC1A, RVMC1B, RVMC1C</td>
</tr>
<tr>
<td>OHR</td>
<td>Horizontal Removing</td>
<td>RMH1A, RMH1B, RMH1C</td>
</tr>
<tr>
<td>OCR</td>
<td>Cubic Removing</td>
<td>CFM</td>
</tr>
<tr>
<td>OFC</td>
<td>Face Cutting</td>
<td>RVMC1A, RVMC1B, RVMC1C</td>
</tr>
<tr>
<td>OCFM</td>
<td>Horizontal Face Milling</td>
<td>RHMC1A, RHMC1B, RHMC1C</td>
</tr>
<tr>
<td>OINS</td>
<td>Inspection</td>
<td>HFMCMM, INS</td>
</tr>
<tr>
<td>OMFHCMM</td>
<td>Horizontal Face Milling/Cutting &amp; Measuring</td>
<td>HFMCMM</td>
</tr>
<tr>
<td>OGM</td>
<td>Grinding Mark</td>
<td>MK2</td>
</tr>
<tr>
<td>ODR</td>
<td>Drilling</td>
<td>VMC1, VMC2</td>
</tr>
<tr>
<td>OGD</td>
<td>Grinding</td>
<td>GDM, DRGD</td>
</tr>
<tr>
<td>ODRGD</td>
<td>Drilling &amp; Grinding</td>
<td>DRGD</td>
</tr>
</tbody>
</table>

It is assumed that a loading plan is given to specify the
assignments of operations and their cutting tools on the
machines. See Kim et al. [18] for more details on the loading
problem. Also, it is assumed that fixture allocation is done in
advance, i.e., the given set of common pallets is divided into
mutually exclusive subsets, within which the pallets are
equipped with a predetermined fixture type. Finally, we assume
that the number of fixtures is enough to clamp parts on pallets.

Since the RMC is being developed, we could not obtain the
real data on part types. Instead, we generated various data based
on the experiences of the project partners. More specifically,
we generated 5000 instances. In the instances, the number of
part types and the production quantities were generated from
DU(30, 100). Finally, the capacity of the central buffer is
36 and the available number of pallets (with fixtures) was
generated from DU(5, 20).

In the decision tree based scheduling approach, the decision
tree was constructed using the four-stage algorithm explained
earlier, where the eligible attributes selected by the correlation
coefficient based method for the problem of minimizing the
total flow time are summarized below. (Initially, we identified
6 static attributes.)

- Number of part types (for total flow time and tardiness)
- Production quantity (for total flow time and tardiness)
- Processing time (for total flow time and tardiness)

Also, to show the performance of the four-stage algorithm,
we constructed another decision tree using the C5.0 algorithm.
For this purpose, we used a commercial software package. The
detailed decision trees are not represented here due to its size.)

In the test, the decision tree based scheduling approach is
compared with 216 priority rule combinations, where the rule
combinations are those of 4 input sequencing rules (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. [19] and Yu et al. [20] for the detailed descriptions of the priority rules tested. (The detailed descriptions are skipped due to the space limitation.) 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

$$\frac{100 \cdot (C_a - C_{best})}{C_{best}},$$

where $C_a$ is the objective value obtained using rule combination $a$ for the instance and $C_{best}$ is the best objective value among those obtained from the 216 rule combinations, the four-stage algorithm and the C5.0 algorithm.

Test results are summarized in Fig. 4 that show the average relative performance ratios of the priority rule combinations, the four-stage algorithm and the C5.0 algorithm out of 5000 test instances. In the figure, the x- and y-axis represent the methods and the relative performance ratios, respectively. As can be seen in the graph, the decision tree based scheduling approach (four-stage and the C5.0 algorithms) outperforms the best rule combinations and particularly, gives stable performance. Also, of the two decision tree construction algorithms, the four-stage algorithm suggested in this study was slightly worse than the C5.0 algorithm (0.004%). However, the four-stage algorithm gave lighter decision tree with less number of eligible attributes than the C5.0 algorithm.

V. CONCLUSION

In this study, we suggested a decision tree based approach for flexible job shop scheduling with multiple process plans. The main decisions 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 developing simulation models and carrying out simulation runs can be eliminated. In the decision tree based scheduling approach, a four-stage algorithm was suggested to construct a decision tree. To show the performance of the decision tree based scheduling approach, a case study was done on a flexible job shop with reconfigurable manufacturing cells and a conventional job shop and the results showed that the decision tree based scheduling approach outperforms the simple priority rule combination based approach that requires simulation runs for each of the objectives of minimizing total flow time and total tardiness.

This study can be extended in several directions. First, it is needed to consider the dynamic flexible job shop scheduling in which jobs arrive over time, not given in advance. In this case, the four-stage decision tree construction algorithm can be used after other dynamic attributes are identified. Second, more case studies for other shop configurations are worth to be performed. Finally, this study can be extended to a decision tree based real-time scheduling mechanism with rescheduling strategies, decision tree update methodologies, etc.

ACKNOWLEDGMENT

This work was supported by the Ministry of Knowledge Economy (MKE) grant funded by Korea government (Grant Code: 10033895-2009-11). This is gratefully acknowledged.

REFERENCES


