Expert Systems with Applications 38 (2011) 3514–3521

Contents lists available at ScienceDirect

Expert Systems with Applications

journal homepage: www.elsevier.com/locate/eswa

Real-time scheduling for reentrant hybrid flow shops: A decision tree based mechanism and its application to a TFT-LCD line

Hyun-Seon Choi, Ji-Su Kim, Dong-Ho Lee *

Department of Industrial Engineering, Hanyang University, South Korea

ARTICLE INFO

Keywords:
Reentrant hybrid flow shops
Real-time scheduling
Decision tree
Case study

ABSTRACT

A reentrant hybrid flow shop, typically found in the electronics industry, is an extended system of the ordinary flow shop in such a way that there exist one or more parallel machines at each serial stage and each job has the reentrant product flow, i.e., a job may visit a stage several times. Among the operational issues in reentrant hybrid flow shops, we focus on the scheduling problem that determines the allocation of jobs to the machines at each stage as well as the sequence of the jobs assigned to each machine. Unlike the theoretical approach on reentrant hybrid flow shop scheduling, we suggest a real-time scheduling mechanism with a decision tree when selecting appropriate dispatching rules. The decision tree, one of the commonly used data mining techniques, is adopted to eliminate the computational burden required to carry out simulation runs to select dispatching rules. To illustrate the mechanism suggested in this study, a case study was performed on a thin film transistor-liquid crystal display (TFT-LCD) manufacturing line and the results are reported for various system performance measures.

1. Introduction

A hybrid flow shop, an extended production system of the ordinary flow shop, consists of two or more stages in series and there exist one or more parallel machines at each stage. In general, the parallel machines at each stage are added for the objective of increasing productivity as well as flexibility. The hybrid flow shop can be found in various types of industries. The most representative one is the electronics industry, such as semiconductor wafer fabrication, printed circuit board (PCB) manufacturing, thin film transistor-liquid crystal display (TFT-LCD) manufacturing, etc. In addition, various traditional industries, such as food, chemical and steel, have various hybrid flow shops.

Hybrid flow shops can be classified into two types according to product flows: (a) those with unidirectional flows; and (b) those with reentrant flows. Here, the unidirectional flows imply that each job starts at the first stage and finishes at the last stage. On the other hand, in the reentrant flows, each job may visit each serial stage two or more times. For example, semiconductor wafer fabrication and TFT-LCD manufacturing lines have the reentrant flows. In other words, each visit of certain specified serial production stage corresponds to a layer that is built up for required circuits. Compared with the unidirectional flows, the reentrant flows generally make system operations much more complicated.

This paper focuses on the scheduling problem in hybrid flow shops with reentrant product flows, called *reentrant hybrid flow shop scheduling* in this paper. The main decisions are: (a) allocation of jobs to machines at each stage; and (b) sequence of the jobs assigned to each machine. In fact, this research was motivated from a TFT-LCD manufacturing system with a large number of complex processes and reentrant product flows. As stated earlier, the reentrant product flows make its operations much more complicated and hence the system performances get worse. To improve the system performances, it is needed to develop and implement an efficient scheduling system. One of its main parts is the robust real-time scheduling methodology that considers the dynamic features of the real TFT-LCD manufacturing system.

Most of the previous studies on reentrant hybrid flow shop scheduling are theoretic in the sense that sophisticated algorithms were devised after developing and analyzing mathematical models with various assumptions. (See Linn and Zhang (1999) for a literature review on hybrid flow shop scheduling.) For example, Bertel and Billaut (2004) suggested a genetic algorithm for reentrant hybrid flow shop scheduling that minimizes the weighted number of tardy jobs, and Choi, Kim, and Lee (2005) suggested several list scheduling algorithms for the problem with the objective of minimizing total tardiness. Recently, Choi et al. (2009) considered the two-stage reentrant hybrid flow shop scheduling problem for the

* Corresponding author. Address: Department of Industrial Engineering, Hanyang University, Seongdong-gu, Seoul 133-791, South Korea. Tel.: +82 2 2220 0475; fax: +82 2 2296 0471.

E-mail addresses: hschoi@kmac.co.kr (H.-S. Choi), iekjs@hanyang.ac.kr (J.-S. Kim), leman@hanyang.ac.kr (D.-H. Lee).

0957-4174/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.eswa.2010.08.139

objective of minimize makespan while meeting the maximum allowable due dates, and suggested several heuristic algorithms. Also, Graves, Meal, Stefek, and Zeghmi (1983) and Hsu and Shamma (1997) considered reentrant flow shop scheduling, a special case of the reentrant hybrid flow shop scheduling. Although these articles have contributions in the theoretical sense, their applications are limited since they are inherently static versions of the problem.

Real-time scheduling, one of practical scheduling approaches, is an important topic on which a number of previous researches have been done. Yamamoto and Nof (1985) proposed a scheduling and rescheduling method in which an initial schedule is generated at the beginning, and schedule revisions are done whenever there are significant operational changes, and Church and Uzsoy (1992) analyzed periodic and even-driven policies for rescheduling in single and parallel machines in the dynamic environment. Kim and Kim (1994) suggested a simulation-based real-time scheduling mechanism for flexible manufacturing systems, in which dispatching rules vary dynamically based on information obtained from simulation, and later, their model was extended by Jeong and Kim (1998) in that a systematic framework is suggested together with scheduling strategies that determine the point of time when a new dispatching rule is selected. Chang (1997) suggested another simulation-based real-time scheduling mechanism in which queueing times for the remaining operations of jobs are estimated and then incorporated into the existing scheduling heuristics for dynamic job shops. Also, Cowling and Johansson (2002) provided a general framework for using real-time information to improve scheduling decisions. For semiconductor wafer fabrication facilities, Kim, Shim, Choi, and Hwang (2003) suggested simplification methods to carry out efficient and prompt dispatching in simulation-based real-time scheduling, and Min and Yih (2003) attempted to improve system performances by combining machine and vehicle dispatching policies.

Unlike the existing approaches explained above, we suggest a real-time scheduling mechanism in which 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 dispatching rule is maintained before considering the rule change. Also, the decision tree, a schematic model to determine one of the alternatives available to a decision maker, is constructed using the information obtained from preliminary data. The real-time scheduling mechanism suggested in this paper is illustrated with a case study on a TFT-LCD manufacturing line, and the test results are reported for various system performance measures.

Although there have been a number of previous research articles on scheduling in semiconductor manufacturing systems, i.e., typical reentrant hybrid flow shops, they have limited applications since they are off-line in nature. See Wein (1988), Glassey and Resende (1988), Lu, Ramaswamy, and Kumar (1994), Kim, Lee, Kim, and Roh (1998), Hung and Chen (1998) for examples. In other words, most of them suggest certain scheduling methods whose solution qualities were shown with static and off-line simulations. Also, the existing real-time scheduling approaches for semiconductor manufacturing select priority dispatching rules using the information obtained from simulation runs and hence they may require significant amount of computational burden. Unlike these, we suggest a real-time scheduling mechanism that increases the speed of the scheduling decisions using the decision tree. Note that the system performances are directly affected by the speed of scheduling system and hence scheduling decisions and actions also have to be made in real-time.

This paper is organized as follows. In the next section, we explain the decision tree based real-time scheduling mechanism. The algorithm to construct the decision tree is also explained. The case study on the TFT-LCD manufacturing line is reported in Section 3. Finally, Section 4 summarizes the main results, gives the conclusions, and describes some areas for further research.

2. Decision tree based real-time scheduling mechanism

This section presents the decision tree based real-time scheduling mechanism suggested in this paper. First, the framework is explained. Then, the components and the algorithm to construct the decision tree are explained in details. Finally, the scheduling strategy, i.e., the time point to select a new dispatching rule, is explained.

2.1. Framework

Fig. 1 shows the framework, i.e., components and necessary information for the real-time scheduling mechanism to work. In fact, the framework is a modified version of the simulation-based one of Jeong and Kim (1998) in that the decision tree, instead of simulation, is used to select a new dispatching rule at the end of each monitoring period.

As can be seen in the figure, the real-time scheduling mechanism suggested in this paper consists of three main components: real-time controller, scheduler, and decision tree based rule selector. A brief explanation of each component is given below. (Details of the components will be explained in the next section.)

- *Real-time controller* exchanges the information with the shop floor, monitors the system states, and dispatches jobs according to the rule released by the scheduler.
- *Scheduler* determines the point of time when a new dispatching rule is to be selected, i.e., implementing the scheduling strategy, and releases the dispatching rule selected by the decision tree based rule selector.
- *Decision tree based rule selector* selects a new dispatching rule (without carrying out time-consuming simulations). The decision tree can be constructed using historical data, knowledge from experts or simulations on the performances of dispatching rules under certain system states. (In our case study, simulation was adopted since there was no preliminary knowledge on the performances of dispatching rules.)

To explain the relations among the three components, we explain three databases required for our real-time scheduling mechanism to work.

- *Decisions in planning stage* contain the information about jobs (with operations), routings, processing times, due dates, performance measures, etc.
- *System states*, updated whenever there is any change in system states, contain the information related to the current system states, i.e., number of jobs in the system, number of remaining operations for each job, processing states of each job, machine states (working, being repaired or idle), etc.
- *Data on the performances of dispatching rules* contain the information required to build up a decision tree, i.e., system performances under certain system states.

2.2. Components

2.2.1. Real-time controller

The real-time controller exchanges the information with the shop floor, monitors the system states, and dispatches jobs according to the rule released by the scheduler. Also, it updates the

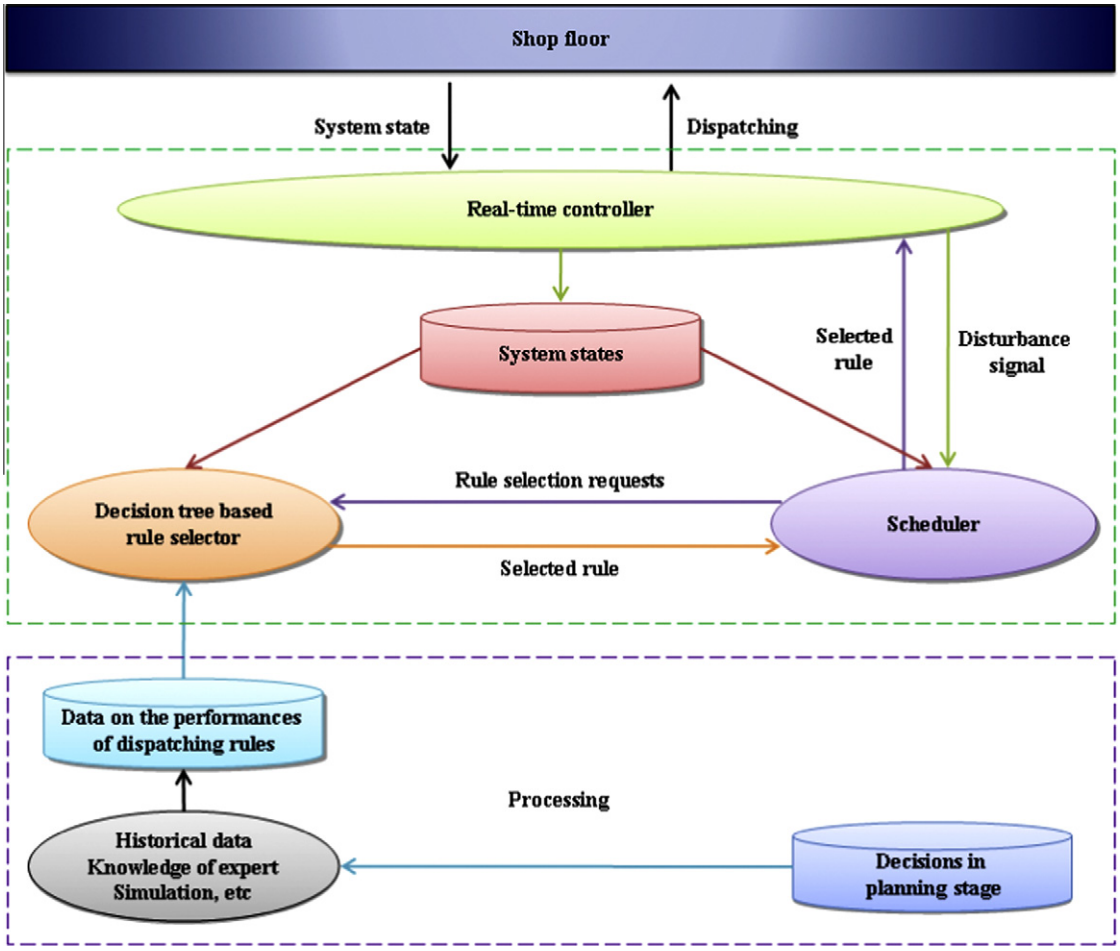

Fig. 1. An overview of the decision tree based real-time scheduling mechanism.

system states database using the system monitoring results and sends a signal to the scheduler if it senses the occurrence of a system disturbance.

2.2.2. Scheduler

The scheduler determines the point of time when a new dispatching rule is to be selected. If the real-time controller senses the occurrences of system disturbances and/or a significant difference between real and estimated performances, it sends a signal to the scheduler. This makes a decision on whether or not a new dispatching rule should be selected. When it is necessary to select a new dispatching rule, the scheduler sends a request to the decision tree based rule selector.

2.2.3. Decision tree based rule selector

When the system requests a new dispatching rule, the decision tree based rule selector selects the best dispatching rule, i.e., a rule that gives the best performances, and it is informed to the scheduler. In the next subsection, we explain the overview of the decision tree and its application to the real-time scheduling mechanism suggested in this paper.

2.3. Constructing the decision tree

The decision tree consists of three types of nodes: non-leaf nodes and leaf nodes. Here, each non-leaf node represents a choice among alternatives while leaf nodes represent classification or decision. Before explaining the decision tree in details, an example of the data set is shown in Table 1, which is adopted from Han (2008). 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 attribute A, B, C, and D are 1, 2, 2, and 1, respectively.

Using the data given in Table 1, we can make various decision trees. Among them, an example is shown in Fig. 2. In the figure, 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., $X = 1$.

Now, we explain how the decision tree is used to select a dispatching rule at the end of each monitoring period. In our

Table 1
Data set for constructing a 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

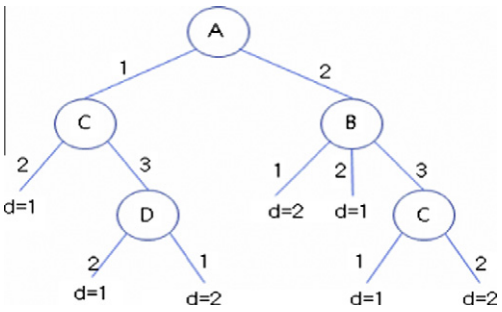

Fig. 2. Decision tree: example.

application, conditional and decision attributes in the data set correspond to the system states and the selection of dispatching rule, respectively. If simulation is used to construct the 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 set of system states and identifying the best dispatching rule. As stated earlier, the data set can be also obtained from the historical data or the knowledge of experts.

The system states considered in this study are summarized below. (Note that more variables can be added for other applications.)

- Total number of remaining operations for the jobs in queue at each stage.
- Total processing time of remaining operations for the jobs in queue at each stage.
- Total number of remaining operations for the jobs being processed at each stage.
- Total processing time of remaining operations for the jobs being processed at each stage.

Note that the decision tree can be updated if there are changes in the cumulated data set. This shows the flexibility of our decision tree based real-time scheduling mechanism.

There are various algorithms to construct the decision tree. Among them, we adopt the Iterative Dichotomiser algorithm of Quinlan (1986), called the ID3 algorithm in the literature, since it has been proved to be simple but effective to express information contained implicitly in discrete valued data sets. The basic idea of the ID3 algorithm is stemmed from the information theory and the pattern recognition. Before explaining the algorithm, a set of objects is defined as a matrix $A = [aij]$, where a_{ij} denotes the value of conditional attribute j of object i. Note that in the matrix, each row vector corresponds to an object without the decision attribute. (See Table 1 for an example.)

The ID3 algorithm uses the entropy function to select the conditional attributes of a decision tree, where the entropy function measures the impurity of an arbitrary collection of objects. More formally, 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 denotes the number of different conditional attribute values, e.g., C_A, C_B, C_C, and C_D are, 2, 2, 3, and 3 for the example in Table 1. Also, $p(w_{cj}|j)$ denotes the proportion of value w_{cj} in conditional attribute j, i.e.,

$$p(w_{cj}|j) = |W_{cj}|/m,$$

where $W_{cj} = \{i|a_{ij} = w_{cj}, \forall i\}$ and m denotes the number of objects. For example, $p(1|A) = 8/12$ and $p(2|B) = 4/12$ in Table 1, and hence the entropy value of conditional attribute A can be calculated as follows:

$$entropy_A = -\frac{8}{12} \cdot \log_2 \frac{8}{12} - \frac{4}{12} \cdot \log_2 \frac{4}{12}$$

The ID3 algorithm constructs the decision tree as follows. First, all the conditional attributes are evaluated using the entropy function and the one with the smallest entropy value is selected. From the root node, a partial decision tree is constructed with the selected conditional attribute. Second, a child node is generated for each conditional attribute value of the root node and it is connected to the root node. As in the root node, the conditional attribute of the child node is set to the one with the smallest entropy value after removing the selected conditional attribute and the objects with the conditional attribute value of the root node. This is done until there is no remaining conditional attributes to be considered. A detailed procedure of the ID3 algorithm is given below.

Procedure 1 (*ID 3 algorithm for constructing a decision tree).*

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 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.

2.4. Scheduling strategy

After the decision tree is constructed, one more decision should be made on the time points when a new dispatching rule is to be selected, i.e., the time when the decision tree is called. To do this, we use the ALL strategy suggested by Jeong and Kim (1998) since it is better than the others. (See Jeong and Kim (1998) for the other scheduling strategies.) In the ALL strategy, the scheduler is called in the following cases.

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

3. Application

This section reports a case study on a TFT-LCD line. First, the system is described. Second, dispatching rules used for the case study are explained. Finally, the performances of the decision tree based real-time scheduling mechanism are reported.

3.1. System description

TFT-LCDs are high-tech display products manufactured through complex processes. A glass of semiconductor material is coated with a thin film of a chemical called photo-resist. Photo-resist coated wafers or glasses are then baked in an oven to remove solvents. Once the baking process is completed, the stepper aligns layers with mask plate and the glass is exposed to ultraviolet light. Then, the glass is developed in the developer. At the final stage, dry and wet etching processes remove thin film layers. The dry

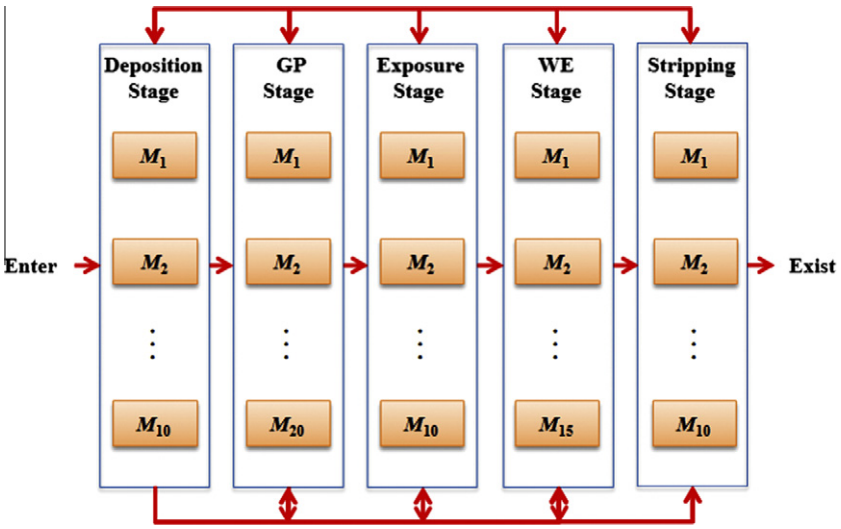

Fig. 3. TFT-LCD manufacturing processes.

Table 2
Product routes and processing time for the case study.

Product type	DS 1	GP 2	EP 3	WE 4	SP 5	GP 6	EP 7	WE 8	SP 9	GP 10	EP 11	WE 12	SP 13	GP 14	EP 15	WE 16	SP 17	GP 18	EP 19	WE 20	SP 21	GP 22	EP 23	WE 24	SP 25	Sum
Product-01	36[a]	54	48	27	126	72	216	0	0	90	78	36	249	72	60	18	180	60	30	18	144	0	0	0	0	1614
Product-02	36	54	30	18	144	72	228	0	0	90	36	42	234	72	78	30	168	60	48	18	150	0	0	0	0	1608
Product-03	54	48	36	30	252	72	60	42	78	0	0	42	300	54	72	18	144	48	36	18	180	0	0	0	0	1584
Product-04	42	48	36	18	120	60	192	0	0	48	42	24	0	0	0	24	180	48	72	18	144	42	54	18	144	1374
Product-05	36	54	30	18	108	60	204	0	0	60	42	24	0	0	0	24	180	72	60	18	90	60	42	30	144	1356
Product-06	36	54	30	24	132	60	192	0	0	60	72	24	0	0	0	30	204	60	72	18	144	60	30	30	120	1452
Product-07	72	72	42	108	324	0	288	0	0	90	72	168	360	90	660	0	0	72	48	48	300	0	0	0	0	2814
Product-08	90	72	60	120	288	72	300	0	0	108	108	180	300	120	240	0	0	120	36	36	240	0	0	0	0	2490
Product-09	72	72	60	126	324	66	348	0	0	72	90	126	348	84	408	48	240	0	0	0	0	0	0	0	0	2484
Product-10	36	72	36	30	126	84	216	0	0	90	72	30	0	0	0	48	252	108	108	30	162	72	24	48	162	1806
Product-11	72	72	54	30	108	30	348	30	0	0	0	30	288	0	0	30	336	0	0	12	144	0	0	48	300	1932

[a] Unit: minutes.

etching process uses reactive species, such as atoms or radicals from the gas plasma, to etch away a portion of the object material. When these species react with the material located on the plate, the open region of material is transformed into a volatile state and removed from the matrix. In this process, the reaction velocity is fast, and fine patterns can be formed uniformly.

The TFT-LCD fabrication process, a typical bottleneck among the whole processes, is similar to the semiconductor wafer fabrication process in that its complexity comes from a large number of operations as well as reentrant flows. The TFT-LCD fabrication process considered in this study can be described as Fig. 3. As can be seen in the figure, there are five serial stages, called deposition (DS) with 10 machines, gate photo (GP) with 20 machines, exposure (EP) with 10 machines, wet etching (WE) with 15 machines, and stripping (SP) with 10 machines, in the line.

In the TFT-LCD manufacturing line, 11 product types are produced. The routes and processing times are summarized in Table 2. According to the operations managers of the line, due dates of jobs were generated from $DU(2.0 \cdot T_i, 4.0 \cdot T_i)$, where $DU(l, u)$ and T denote a discrete uniform distribution with range $[l, u]$ and the sum of the operation times of the job i, respectively. Preemption is not allowed due to the technical problems. It is assumed that the transportation time is ignored since the material handling system is not the bottleneck in the line, and set-up times are included in the processing times. Finally, the other problem data are summarized below. Note that some of the data are artificial due to the confidential problem and the difficulties to obtain the exact data.

- Jobs arrive with an inter-arrival time generated from EXP(10), where EXP(λ) is an exponential distribution with a mean of λ.
- Major machine breakdowns occur with an inter-failure time of EXP(15,000), and repair times were generated from EXP(500).
- Minor breakdowns occur with an interval generated from EXP(6000) for each machine, and repair time follows EXP(150).
- Buffer size, i.e., maximum number of available waiting jobs at each stage, was set to 200.

Due to a large number of operations and reentrant flows, the TFT-LCD manufacturing line has low system throughput, long flow time, and bad due date related performance measures. Therefore, our motive is to suggest new and practical real-time scheduling mechanism that can help to improve its system performances. Multiple performance measures are considered in this study. They are: (a) maximizing system throughput; (b) minimizing mean flow time; (c) minimizing mean tardiness; and (d) minimizing the number of tardy jobs.

3.2. Dispatching rules

Dispatching rules are used for selecting a job among those waiting in a queue at each stage when a machine becomes available.

(See Panwalkar and Iskander (1977) and Blackstone, Phillips, and Hogg (1982) for surveys on various dispatching rules.) The dispatching rules tested in the case study are summarized below. Note that other rules can be added since the real-time mechanism is flexible in this respect.

FCFS (first come first served): select an operation that arrived at the queue first.
SPT (shortest processing time): select an operation with the shortest operation processing time, i.e., min p_j, where p_j denotes the processing time of operation j.
LPT (longest processing time): select an operation with the longest operation processing time, i.e., max p_j.
LOR (least operation remaining): select an operation with the least number of remaining operations, i.e., min o_j, where o_j denotes the remaining operations of operation j (number of successor operations including itself).
MOR (most operation remaining): select an operation with the largest number of remaining operations, i.e., max o_j.
LWR (least work remaining): select the operation with the least remaining work, i.e., min r_j, where r_j denotes the remaining work of operation j (sum of processing times of the successor operations including itself).
MWR (most work remaining): select the operation with the most remaining work, i.e., max r_j.
PWR (processing time to work remaining): select an operation with the smallest ratio of the processing time to remaining work, i.e., min p_j/r_j.
POR (processing time to operation remaining): select an operation with the smallest ratio of the processing time to remaining operations, i.e., min p_j/o_j.
EDD (earliest due date): select an operation with the earliest due date, i.e., min d_j, where d_j denotes the due date of the job in which operation j is included.
SLACK (minimum slack): select an operation with the minimum slack time, i.e., min $\{d_j - r_j - t\}$, where t is the current time.
MDD (modified due date): select an operation with the minimum modified due date, where the modified due date of operation j is defined as $\max\{d_j, t + r_j\}$.
S/RO (slack per remaining operations): select an operation with the smallest ratio of slack time to the remaining operations, i.e., $(d_j - r_j - t)/o_j$.
S/RW (slack per remaining work): select an operation with the smallest ratio of slack time to the remaining work, i.e., $(d_j - r_j - t)/r_j$.

3.3. Experimental design and results

The main purpose of the test is to compare the decision tree based real-time scheduling mechanism (that eliminates the computational burdens of simulation runs for selecting dispatching rules) with the existing simulation-based one (that selects dispatching rules using time-consuming simulation results).

As stated earlier, the performance measures considered in this study are system throughput, mean flow time, mean tardiness, and the number of tardy jobs. In this study, the data for constructing the decision tree were obtained from steady-state simulation runs because the system has no preliminary data. (See Law and Kelton (1991) for more details on steady-state simulations.) The two real-time scheduling mechanisms, together with the simulation model, were coded in C++ and the test was done on a workstation with an Intel Xeon processor operating at 3.2 GHz clock speed.

The comparisons were done in two cases. The first case assumes that the shop floor is not operated during the simulation run time for deciding a new dispatching rule and hence the losses in system performances are not considered. In this case, the simulation-based mechanism gives better results than the decision tree based one because the decision tree based one is an approximation of the simulation-based one. Nevertheless, the decision tree based mechanism has an inherent merit in that the simulation model needs not be required. On the other hand, in the second case, the shop floor is operated with the current dispatching rule during the simulation run time and hence the losses in system performances are explicitly considered.

In the test, we performed the comparisons according to three levels of the performance limit in the scheduling strategy (1%, 5% and 10%). Recall that one of the scheduling strategies is that a new rule is selected if a certain performance value exceeds a predetermined performance limit at the end of each monitoring period. For each level of the performance limit, we performed five replications for each of the eight combinations for two levels for the simulation time for deciding a new dispatching rule in the simulation-based mechanism (500 and 1000) and three levels for the length of the periodic monitoring period (2500, 5000, and 10,000). The performance measure used is the relative performance ratio, which is defined as

$$100 \times (C_a - C_{best})/C_{best}$$

for the minimization objectives (mean flow time, mean tardiness, and the number of tardy jobs), and

$$100 \times (C_{best} - C_a)/C_{best}$$

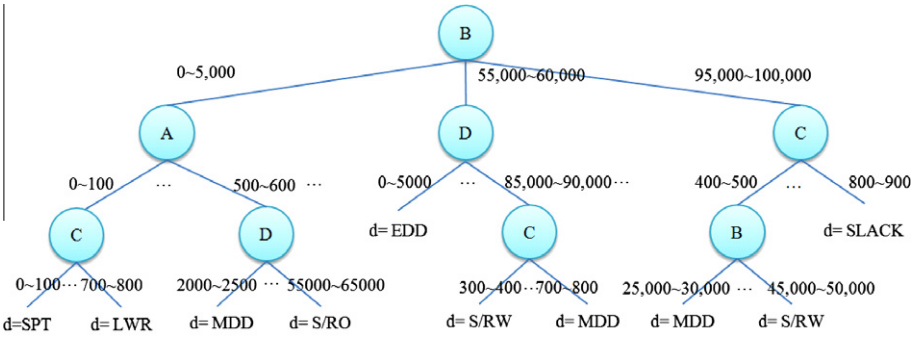

Fig. 4. A part of the decision tree used in the case study.

Table 3
Results for the comparison test.

	Simulation times	Monitoring period	Simulation-based mechanism				Decision tree based mechanism			
			Throughput	Mean flow time	Mean tardiness	Number of tardy jobs	Throughput	Mean flow time	Mean tardiness	Number of tardy jobs
(a) Results for performance limit 1%										
Case 1[a]	500	2500	0.00[c]	0.00	0.00	0.00	2.22	2.39	2.51	2.02
		5000	0.00	0.00	0.00	0.00	3.61	4.28	5.88	2.91
	1000	5000	0.00	0.00	0.00	0.00	3.23	3.62	4.81	2.26
		10,000	0.00	0.00	0.00	0.00	2.49	2.01	3.39	2.22
Case 2[b]	500	2500	0.19	0.14	0.11	0.05	0.29	0.32	0.28	0.43
		5000	0.19	0.08	0.21	0.21	0.33	0.36	0.22	0.36
	1000	5000	0.23	0.33	0.26	0.21	0.26	0.25	0.39	0.34
		10,000	0.21	0.22	0.31	0.19	0.29	0.30	0.34	0.42
(b) Results for performance limit 5%										
Case 1	500	2500	0.00	0.00	0.00	0.00	1.89	2.62	2.54	2.51
		5000	0.00	0.00	0.00	0.00	4.21	3.88	4.51	3.83
	1000	5000	0.00	0.00	0.00	0.00	3.78	4.27	4.69	4.28
		10,000	0.00	0.00	0.00	0.00	3.52	2.51	5.32	3.93
Case 2	500	2500	0.23	0.26	0.18	0.24	0.26	0.32	0.26	0.35
		5000	0.16	0.19	0.25	0.30	0.23	0.36	0.22	0.28
	1000	5000	0.23	0.26	0.23	0.23	0.19	0.29	0.39	0.25
		10,000	0.26	0.33	0.32	0.24	0.31	0.24	0.24	0.27
(c) Results for performance limit 10%										
Case 1	500	2500	0.00[c]	0.00	0.00	0.00	1.41	2.91	2.09	2.22
		5000	0.00	0.00	0.00	0.00	2.19	2.21	2.12	2.41
	1000	5000	0.00	0.00	0.00	0.00	2.12	3.62	3.32	2.89
		10,000	0.00	0.00	0.00	0.00	2.54	2.77	3.14	2.49
Case 2	500	2500	0.11	0.23	0.18	0.23	0.12	0.34	0.31	0.31
		5000	0.22	0.21	0.27	0.19	0.19	0.28	0.36	0.22
	1000	5000	0.32	0.32	0.36	0.26	0.29	0.25	0.29	0.34
		10,000	0.29	0.21	0.31	0.21	0.21	0.34	0.36	0.25

[a] The case that the shop floor is not operated during the simulation time for deciding a new dispatching rule.
[b] The case that the shop floor is operated during the simulation time for deciding a new dispatching rule.
[c] Average relative performance ratio (out of five instances).

for the maximization objective (system throughput), where C_a is the solution value obtained from real-time scheduling mechanism a and C_{best} is the better one of the two solution values.

When constructing the decision tree using the preliminary simulation results, the conditional attributes were defined in the form of range instead of value. The resulting decision tree can be represented as Fig. 4 in which conditional attributes A, B, C and D denote the total number of remaining operations for the jobs in queue at each stage, the total processing time of remaining operations for the jobs in queue at each stage, the total number of remaining operations for the jobs being processed at each stage, and the total processing time of remaining operations for the jobs being processed at each stage, respectively.

Test results on the two real-time scheduling mechanisms are summarized in Table 3. As can be seen in the table, the main result is that the differences in performances are not significantly large. (Recall that the decision tree based mechanism needs not require simulation runs.) In particular, for the second case in which the shop floor is operated with the current dispatching rule (before change) during the simulation time, there were no significant differences for all performance measures, which implies that the losses in system performances due to poor dispatching rules during the simulation run time are significant. Also, we found that the decision tree based mechanism may give better performances for some measures and parameter values. It was observed that the gaps between the two mechanisms get smaller as the performance limit gets increased (from 1% to 10%) because the current bad dispatching rule is used longer under larger performance limits. Therefore, we can see that the rule section mechanism plays an important role for immediate responses to changes in system states. In summary, we can argue that the decision tree based mechanism is worth to be considered for practical scheduling problems, especially, in the scheduling systems without preparing simulators.

4. Concluding remarks

We considered the scheduling problem in reentrant hybrid flow shops that have a number of applications in various manufacturing and service systems. Unlike the existing theoretical approaches, we suggested a real-time scheduling mechanism in which a decision tree is used to select an appropriate dispatching rule so that the computational burden required for carrying out simulations can be eliminated. The decision tree based real-time scheduling mechanism was applied to a TFT-LCD manufacturing line, i.e., a typical reentrant hybrid flow shop, and the test results showed that it is competitive to the simulation-based one with respect to various performance measures such as system throughput, mean flow time, mean tardiness, and the number of tardy jobs.

As a modification of the existing simulation-based real-time scheduling mechanism, this research can be extended in several directions. First, other algorithms to construct the decision tree may be used. In other words, it may be needed to construct more sophisticated decision trees. Second, more case studies that incorporate specific system characteristics are worth to be performed.

Acknowledgement

This work was supported by Brain Korea 21 Grant funded by Korean Government. This is gratefully acknowledged.

References

Bertel, S., & Billaut, J. C. (2004). A genetic algorithm for an industrial multiprocessor flow shop scheduling problem with recirculation. *European Journal of Operational Research, 159*, 651–662.

Blackstone, J. H., Phillips, D. T., Jr., & Hogg, G. L. (1982). A state-of-the art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research, 20*, 7–45.

Chang, F. (1997). Heuristics for dynamic job shop scheduling with real-time updated queueing time estimates. *International Journal of Production Research, 35*, 651–665.

Choi, S.-W., Kim, Y.-D., & Lee, G.-C. (2005). Minimizing total tardiness of orders with reentrant lots in a hybrid flowshop. *International Journal of Production Research, 43*, 2149–2167.

Choi, H.-S., Kim, H.-W., Lee, D.-H., Yun, J., Yoon, C. Y., & Chae, K. B. (2009). Scheduling algorithms for two-stage reentrant hybrid flow shops: minimizing makespan under the maximum allowable due-dates. *International Journal of Advanced Manufacturing Technology, 42*, 963–973.

Church, L. K., & Uzsoy, R. (1992). Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing, 5*, 153–163.

Cowling, P., & Johansson, M. (2002). Using real-time information for effective dynamic scheduling. *European Journal of Operational Research, 139*, 230–244.

Glassey, C. R., & Resende, M. C. G. (1988). A scheduling rule for job release in semiconductor fabrication. *Operations Research Letters, 7*, 213–217.

Graves, S. C., Meal, H. C., Stefek, D., & Zeghmi, A. H. (1983). Scheduling of reentrant flow shops. *Journal of Operations Management, 3*, 197–207.

Han, S. W. (2008). Rough set based decision tree with static and dynamic entities. Ph.D. Dissertation. Seoul, South Korea: Hanyang University.

Hsu, C., & Shamma, J. S. (1997). Receding horizon control for manufacturing systems. *Proceedings of the American Control Conference*, 1914–1918.

Hung, Y. F., & Chen, I. R. (1998). A simulation study of dispatch rules for reducing flow times in semiconductor wafer fabrication. *Production Planning and Control, 9*, 714–722.

Jeong, K.-C., & Kim, Y.-D. (1998). A real-time scheduling mechanism for a flexible manufacturing system: Using simulation and dispatching rules. *International Journal of Production Research, 36*, 2609–2626.

Kim, M.-H., & Kim, Y.-D. (1994). Simulation-based real-time scheduling mechanism in a flexible manufacturing system. *Journal of Manufacturing Systems, 13*, 85–93.

Kim, Y.-D., Lee, D.-H., Kim, J.-U., & Roh, H.-K. (1998). A simulation study on lot release control, dispatching and batch scheduling in semiconductor wafer fabrication facilities. *Journal of Manufacturing Systems, 17*, 107–117.

Kim, Y.-D., Shim, S.-O., Choi, B., & Hwang, H. (2003). Simplification methods for accelerating simulation-based real-time scheduling in a semiconductor wafer fabrication facility. *IEEE Transactions on Semiconductor Manufacturing, 16*, 290–298.

Law, A. M., & Kelton, W. D. (1991). *Simulation modeling and analysis*. New York: McGraw-Hill.

Linn, R., & Zhang, W. (1999). Hybrid flow shop scheduling: A survey. *Computers and Industrial Engineering, 37*, 57–61.

Lu, S. C. H., Ramaswamy, D., & Kumar, P. R. (1994). Efficient scheduling policies to reduce mean and variance of cycle-time in semiconductor manufacturing plants. *IEEE Transactions on Semiconductor Manufacturing, 7*, 374–388.

Min, H. S., & Yih, Y. (2003). Selection of dispatching rules on multiple dispatching decision points in real-time scheduling of a semiconductor wafer fabrication system. *International Journal of Production Research, 41*, 3921–3941.

Panwalkar, S., & Iskander, W. (1977). A survey of scheduling rules. *Operations Research, 25*, 45–61.

Quinlan, J. R. (1986). Introduction of decision trees. *Machine Learning, 1*, 80–106.

Wein, L. M. (1988). Scheduling semiconductor wafer fabrication. *IEEE Transactions on Semiconductor Manufacturing, 1*, 115–130.

Yamamoto, M., & Nof, S. Y. (1985). Scheduling/rescheduling in the manufacturing operating system environment. *International Journal of Production Research, 23*, 705–722.