International Journal of Production Research

ISSN: 0020-7543 (Print) 1366-588X (Online) Journal homepage: http://www.tandfonline.com/loi/tprs20

Real time selection of scheduling rules and knowledge extraction via dynamically controlled data mining

Gokhan Metan , Ihsan Sabuncuoglu & Henri Pierreval

To cite this article: Gokhan Metan , Ihsan Sabuncuoglu & Henri Pierreval (2010) Real time selection of scheduling rules and knowledge extraction via dynamically controlled data mining, International Journal of Production Research, 48:23, 6909-6938, DOI: 10.1080/00207540903307581

To link to this article: http://dx.doi.org/10.1080/00207540903307581

Published online: 15 Jan 2010.

Submit your article to this journal

Article views: 399

View related articles

Citing articles: 20 View citing articles

Full Terms & Conditions of access and use can be found at
http://www.tandfonline.com/action/journalInformation?journalCode=tprs20

Download by: [Bilkent University] **Date:** 13 November 2017, At: 02:37

International Journal of Production Research
Vol. 48, No. 23, 1 December 2010, 6909–6938

Taylor & Francis
Taylor & Francis Group

Real time selection of scheduling rules and knowledge extraction via dynamically controlled data mining

Gokhan Metan[a]*†, Ihsan Sabuncuoglu[b] and Henri Pierreval[c]

[a]*Department of Industrial and Systems Engineering, Lehigh University, Bethlehem, PA 18015, USA;* [b]*Department of Industrial Engineering, Bilkent University, Bilkent, Ankara 06533, Turkey;* [c]*LIMOS, UMR CNRS 6158, IFMA, Campus des Cezeaux, BP 265, F-63175 Aubiere Cedex, France*

(*Received 8 August 2007; final version received 24 August 2009*)

A new scheduling system for selecting dispatching rules in real time is developed by combining the techniques of simulation, data mining, and statistical process control charts. The proposed scheduling system extracts knowledge from data coming from the manufacturing environment by constructing a decision tree, and selects a dispatching rule from the tree for each scheduling period. In addition, the system utilises the process control charts to monitor the performance of the decision tree and dynamically updates this decision tree whenever the manufacturing conditions change. This gives the proposed system the ability to adapt itself to changes in the manufacturing environment and improve the quality of its decisions. We implement the proposed system on a job shop problem, with the objective of minimising average tardiness, to evaluate its performance. Simulation results indicate that the performance of the proposed system is considerably better than other simulation-based single-pass and multi-pass scheduling algorithms available in the literature. We also illustrate knowledge extraction by presenting a sample decision tree from our experiments.

Keywords: adaptive control; data mining; simulation; dispatching rules; dynamic scheduling; game theory; inventory management; pricing theory; radio frequency identification; scheduling

Downloaded by [Bilkent University] at 02:37 13 November 2017

1. Introduction

The scheduling problems encountered in many real life applications are stochastic and dynamic in nature, making dispatching rules a commonly used technique in practice. There are numerous such rules in the literature and simulation is the primary tool to test the performance of these rules in complex environments (i.e. stochastic and dynamic job shop problems). The results indicate that none of the dispatching rules is superior in every setup. Hence, selection of the appropriate rule(s) is not a trivial task. Another result prevalent in the literature is that switching to different rules (multi-pass) yields better performance than using one rule (single-pass) for the entire horizon.

In a single-pass scheduling algorithm, a set of candidate dispatching rules is simulated and the one with the best long-run performance is selected and used during the whole planning horizon. On the other hand, multi-pass algorithms evaluate all the candidate

*Corresponding author. Email: gom204@lehigh.edu
†This study was conducted when Gokhan Metan was at Bilkent University.

ISSN 0020–7543 print/ISSN 1366–588X online
© 2010 Taylor & Francis
DOI: 10.1080/00207540903307581
http://www.informaworld.com

dispatching rules in each relatively short scheduling period and select the best performer to be used in that interval. Thus, in the long run, this type of algorithm results in a combination of different dispatching rules.

We group the most relevant studies in the literature into five categories and provide a summary in Table 1. We start our discussion with the studies in the first category, optimal design and configuration of manufacturing systems.

Huyet and Paris (2004) developed a data-mining-based approach for optimal design and configuration of production systems. They utilised simulation and evolutionary algorithms to optimally configure production systems while utilising data mining to determine production system parameters such as transport lot sizes, number of kanbans, and dispatching rules. Later, Huyet (2006) implemented the same technique on a job shop configuration problem. Although, in both studies, the dispatching rule is considered as a configuration parameter that should be optimally selected, it is not dynamically adjusted over time for changes in shop floor conditions. Yildirim *et al.* (2006) studied a different approach, based on simulation and neural networks, to design flexible manufacturing systems. One of the fundamental differences between these studies and our study is that they approached dispatching rule selection at a tactical level, whereas our approach considers the problem from an operational perspective by continuously selecting a new dispatching rule for each scheduling period.

In the second category of research, Sha and Liu (2005) studied the due-date assignment problem and developed a data mining approach for extracting knowledge on different due-date assignment rules. In the experiments, they considered dispatching rules as a predictor for due-date assignment rules. As a result, their decision tree makes a decision by selecting a due-date assignment rule for a given dispatching rule, number of jobs in the system, processing time requirement of jobs, etc.

Another direction of research studies the generation of new dispatching rules via data mining and evolutionary algorithms. Li and Olafsson (2005) proposed a data-mining-based approach for discovering new dispatching rules. Their technique is based on extracting knowledge from the performances of a given set of dispatching rules (earliest due-date, minimum slack, etc.) by constructing a decision tree. The final output is a decision tree that is referred to as a new dispatching rule. This decision tree is later used to make a decision for releasing or not releasing a job to the machine based on attribute values such as processing time requirement and processing time difference. Once a new dispatching rule (a decision tree) is constructed, it is not dynamically updated in time for changing manufacturing conditions. Geiger *et al.* (2006), and Geiger and Uzsoy (2006) approached the same problem of new dispatching rule discovery by combining simulation and GA techniques.

The most relevant research category for our study is dispatching rule selection. There are various studies concerning scheduling via dispatching rules that employ different techniques such as iterative simulation, neural networks (NN), optimisation, and data mining. One common feature of the very early works in this category is that they used simulation as the primary tool for assessing the efficiency of dispatching rules, and selected the best-performing rule for scheduling jobs on machines. Examples of such studies are those of Wu and Wysk (1988, 1989), Ishii and Talavage (1991, 1994), Tayanithi *et al.* (1993a, b), Kim and Kim (1994), Jeong and Kim (1998) and Kutanoglu and Sabuncuoglu (2001).

One of the very first studies that applies learning techniques to scheduling problems is the work of Pierreval and Ralambondrainy (1990). In their research, learning algorithms

Downloaded by [Bilkent University] at 02:37 13 November 2017

Downloaded by [Bilkent University] at 02:37 13 November 2017

Table 1. Summary of the literature.

Category	Study	Techniques used	Test problem	Knw. Extrc.	Dyn. Updt.
Optimal design and configuration of the manufacturing system	Huyet and Paris (2004)	Simulation, GA, C4.5	Kanban system	Yes	–
	Huyet (2006)	Simulation, GA, C4.5	Job shop	Yes	–
	Yildirim *et al.* (2006)	Simulation, NN	FMS	No	–
Due-date assignment	Sha and Liu (2005)	Simulation, C4.5	Job shop	Yes	No
dispatching rule generation	Li and Olafsson (2005)	C4.5	Single machine	Yes	No
	Geiger *et al.* (2006)	Simulation, GA	Single machine	No	No
	Geiger and Uzsoy (2006)	Simulation, GA	Single machine	No	No
Dispatching rule selection	Wu and Wysk (1988, 1989)	Simulation	FMS	No	No
	Ishii and Talavage (1991, 1994)	Simulation	FMS	No	No
	Tayanithi *et al.* (1993a, b)	Simulation	FMS	No	No
	Kim and Kim (1994)	Simulation	FMS	No	No
	Jeong and Kim (1998)	Simulation	FMS	No	No
	Kutanoglu and Sabuncuoglu (2001)	Simulation	Job shop	No	No
	Cho and Wysk (1993)	Simulation, NN	FMS	No	No
	Pierreval and Mebarki (1997)	SFSR heuristic	Job shop	Yes	No
	Pierreval and Ralambondrainy (1990)	Simulation, learning	Job shop	Yes	No
	Pierreval (1993)	Simulation, NN	Flow shop	No	No
	El-Bouri and Shah (2006)	NN	Job shop	No	No
	Sun *et al.* (2004)	Simulation-optimisation	Job shop	No	No
	Lee *et al.* (1997)	Simulation, GA, C4.5	Job shop	Yes	No
	Min and Yih (2003)	Simulation, NN	Job shop	No	No
	Aissani *et al.* (2008)	Learning	Job shop	No	No
Attribute selection	Shiue and Guh (2005)	Simulation, GA, NN	FMS	No	No
	Shiue and Guh (2006)	Simulation, GA, NN	FMS	No	No
Review	Harding *et al.* (2006)	–	–	–	–

are used to mine the information captured by simulation experiments. The output of learning algorithms is a set of production rules that are then utilised as a knowledge base for selecting scheduling rules. In other studies, such as those of Cho and Wysk (1993), Pierreval (1993), Min and Yih (2003) and El-Bouri and Shah (2006), techniques that combine simulation and neural networks are proposed. One drawback of such studies is the neural networks, which lack the ability to extract knowledge for decision-makers to understand the system dynamics.

Lee *et al*. (1997) developed a scheduling approach using simulation, GA and data mining (C4.5 algorithms) to select dispatching rules, and release jobs onto the shop floor. They used data mining for learning about releasing jobs, leaving the decision on dispatching rule selection to heuristic optimisation (GA).

Under the last research category we have identified the work of Shiue and Guh (2005, 2006). They studied the problem of attribute selection for learning-based production control systems. They emphasised the issue of selecting essential system attributes in a manufacturing system where an excessive amount of information is available.

This study extends the previous work of Metan and Sabuncuoglu (2005), which addresses the problem of selecting the right dispatching rule from a set of candidate rules. The proposed system integrates the machine learning technique of data mining from computer science, the process control charts from statistical process control, and computer simulation. The objective of our system is to learn about the characteristics of the manufacturing system by constructing a decision tree and then selecting a dispatching rule for a scheduling period from this tree on-line. Moreover, we use control charts to monitor the actual performance of the decision tree. If these charts signal that the current decision tree is beginning to perform poorly, a new tree is constructed based on the recent information gathered from the manufacturing system via simulation.

In summary, our study is different from the existing literature in several ways. First, we utilise data mining, rather than black box methods such as iterative simulation or neural networks, to extract knowledge on dispatching rule selection. Second, we use statistical process control (SPC) techniques to monitor the performance of the decision tree. This approach, represented by the last criterion (i.e. Dyn. Updt., which stands for dynamic update) in Table 1, to the least of our knowledge has not been utilised in a single study in the literature. However, the world does change and there is a need for such techniques that dynamically update themselves to better respond to changing conditions. Our proposed system addresses this need by providing the power of adaptation to changes in the shop floor state. Our proposed system is able to make real-time scheduling decisions, extract knowledge, and combine SPC and learning principles to achieve adaptation to changes in the manufacturing environment.

The rest of the paper is organised as follows. In Section 2 we present the proposed system. We describe the experimental design in Section 3. Computational results along with a sample decision tree are presented in Section 4. Finally, we give concluding remarks and future research directions in Section 5.

2. Proposed system

The goal of the proposed system is to select the best DR among candidates dispatching rules (CDRs) for a particular scheduling period. The general structure is shown in Figure 1. In the architecture of the system, there are five main subroutines, called modules.

Downloaded by [Bilkent University] at 02:37 13 November 2017

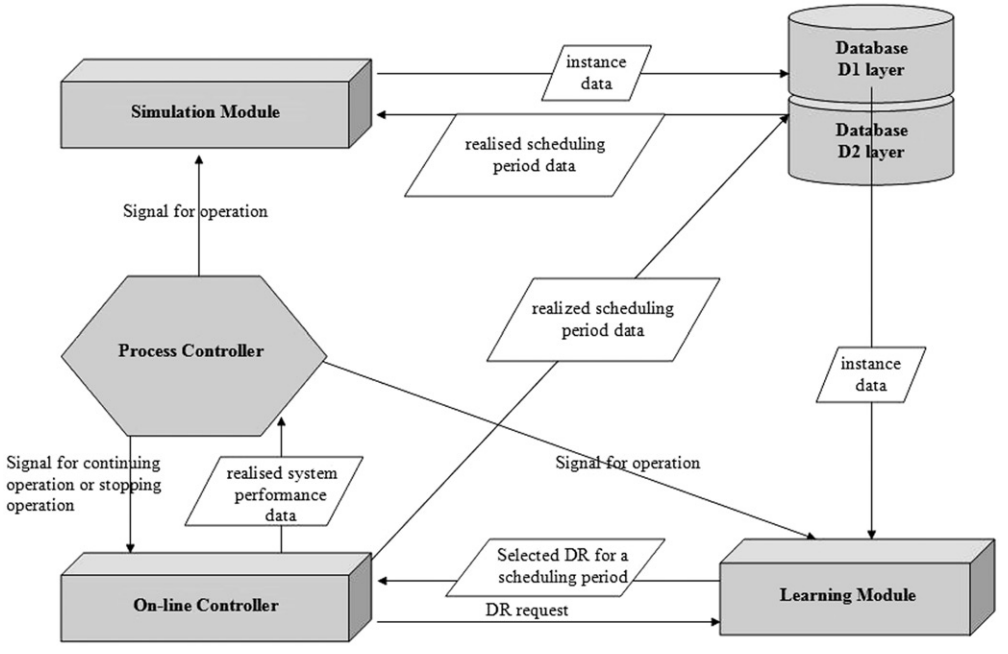

Figure 1. Proposed system – general structure.

Downloaded by [Bilkent University] at 02:37 13 November 2017

They operate in harmony to achieve the goal of selecting the best-performing dispatching rule for each scheduling period. The *database* is composed of two layers, D1 and D2, and provides necessary data for both the Simulation Module and the Learning Module. It holds two types of data. The *instance data*, which is composed of a number of attributes that take values of the manufacturing conditions and a class value that corresponds to the DR selected for a specific condition, is stored in layer D1 and is supplied to the learning algorithm to generate the learning tree. The *realised scheduling period data* includes the actual events that occur in a specific scheduling period, such as the processing times, inter-arrival times and system conditions at the beginning of a scheduling period. This data is also stored in the database (layer D2) for assessment of DRs via simulation. Note that the realised scheduling period data consists of random number generator seed numbers when experimenting in a computerised environment (rather than reflecting a real-life manufacturing system).

The *simulation module* is used to measure the performances of the candidate dispatching rules. It is invoked by the *process controller module* whenever necessary. The simulation module's outputs (instance data) are sent to the database to be further used by the *learning module* to generate the decision tree. The learning module is mainly composed of two parts: *learning module 1* (LM1) and *learning module 2* (LM2) (see Figure 2). LM1 contains the decision tree that is constructed by the learning algorithm in LM2. LM1 is responsible for selecting a new DR from the existing decision tree based on the current system state, which is described by the attribute values. The *on-line controller module* provides the current values of these attributes to LM1 and requests a new DR. In response, LM1 recommends the best DR to the on-line controller (see Figure 2). LM2 contains the learning algorithm that is used to generate the decision tree in LM1. As seen in

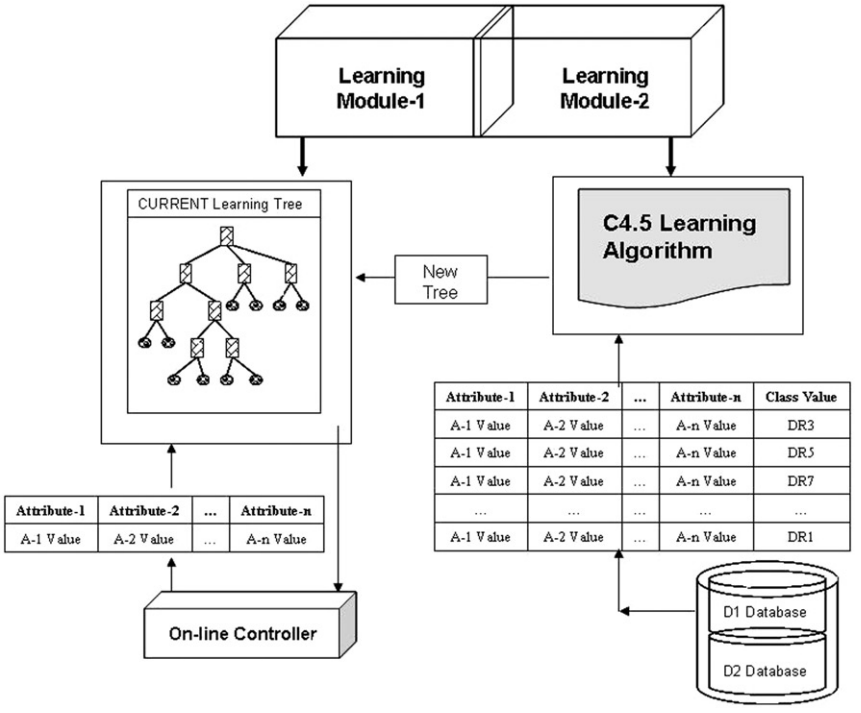

Figure 2. Learning module.

Figures 1 and 2, the algorithm is invoked by the process controller module and the necessary data (instance data) are retrieved from the D1 database.

The C4.5 algorithm (Quinlan 1993) is used to create the decision tree. Our choice of selecting the C4.5 algorithm as the learning mechanism is due to its wide acceptance in academia and industry as an efficient learning algorithm (see, for example, Lee *et al.* (1997), Huyet and Paris (2004), Sha and Liu (2005) and Li and Olafsson (2005)). It has also recently been identified as one of the top 10 data mining algorithms by the IEEE International Conference on Data Mining (ICDM) (Wu *et al.* 2008). It has also been further enhanced recently as C5 and is available as a commercial learning tool. We illustrate our new approach using C4.5, investigate its performance and the knowledge it extracts from data. Indeed, other types of algorithms could be used (either based on trees or on rule induction) without modifying the approach. The numerical experiments in later sections show the efficiency of what can be done with C4.5.

Whenever a scheduling decision is to be made according to the current scheduling strategy (e.g., hybrid approach), the decision tree selects a new dispatching rule and this decision is implemented by the on-line controller module (i.e. it employs the selected DR in actual manufacturing conditions) (see Figure 3). The on-line controller module also supplies the realised scheduling period data to the database and monitors the real system for new rule selection symptoms, the triggering events that are defined in the scheduling strategy to answer the question of 'when-to-schedule'. The *process controller module* monitors the performance of the decision tree (see Figure 4). It takes its inputs (realised value of average tardiness) from the on-line controller module and monitors the

Downloaded by [Bilkent University] at 02:37 13 November 2017

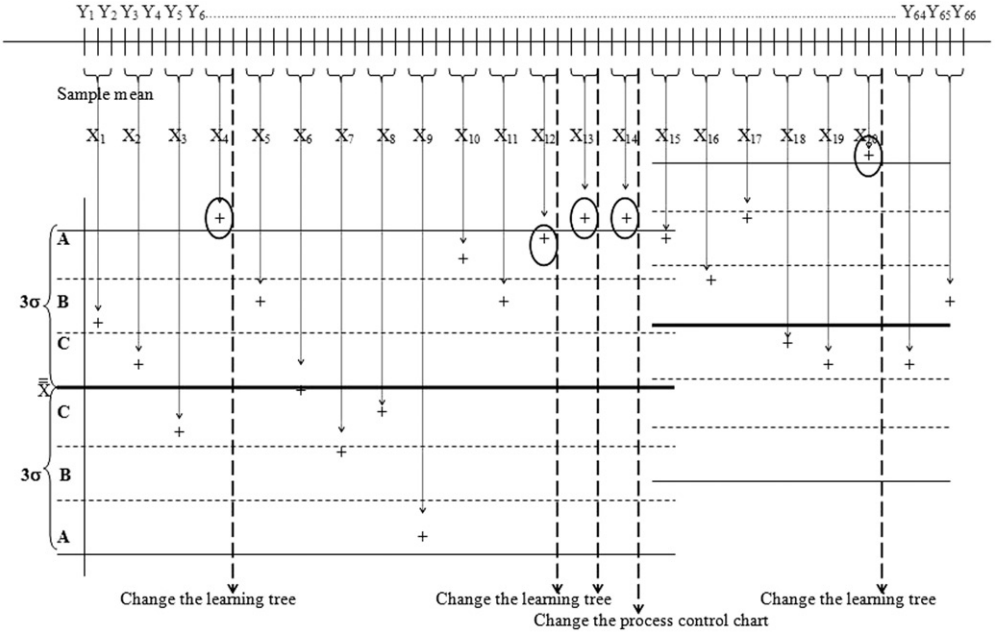

Figure 3. On-line controller module.

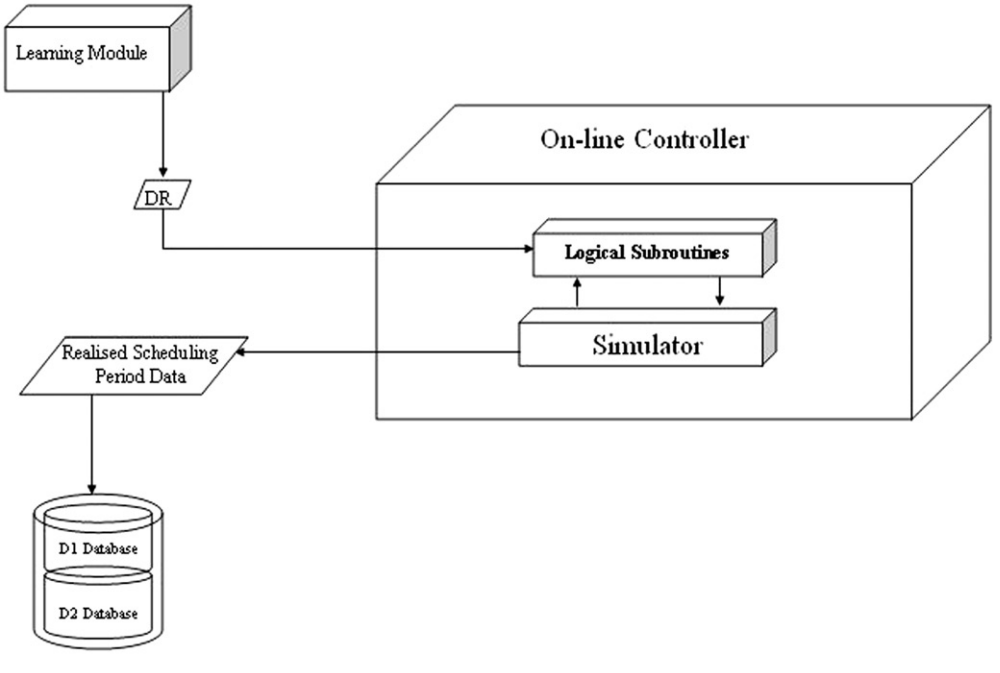

Figure 4. Process controller module and its relationships with other modules.

Downloaded by [Bilkent University] at 02:37 13 November 2017

performance of the decision tree. When the performance of the current decision tree is found to be insufficient, it requests the simulation module to provide new training data (instance data) for the learning module and then sends a signal to the learning module to update the current decision tree using this new data set. As a result, new dispatching rules are selected from this updated decision tree and the process continues in this manner.

The scheduling strategy employed in this research is composed of two critical decisions: how-to-schedule and when-to-schedule. The how-to-schedule decision determines the way in which the schedules are revised or updated. As discussed by Sabuncuoglu and Goren (2003), there are mainly three issues: scheduling scheme, amount of data used, and type of response. Our implementation is based on the 'on-line' scheduling scheme. Specifically, DRs are selected by the decision tree and the scheduling decisions are made one at a time using these selected rules. In terms of the amount of data, we apply the 'full' scheme, and as the type of response, we use the 'reschedule' option, since a new DR is selected at any time when the performance of the existing DR in use is found to be poor. The 'when-to-schedule' determines the responsiveness of the system to various kinds of disruption. The hybrid approach is employed for 'when-to-schedule' decisions, in which two different triggering events, called *new rule selection symptoms*, are defined to determine the time to select a new DR. These new rule selection symptoms and their definitions are given in Table 2.

One of the distinguishing features of the proposed scheduling system is its mechanism that continuously updates the decision tree. This continuous update is important since the manufacturing system often undergoes various types of changes over time. In this context, the process control charts ($\overline{X}$ and R charts) act as a regulator of the learning tree. Moreover, the process control charts might also need to be updated due to changes in manufacturing conditions. Hence, as the proposed system evolves over time, two important decisions need to be made.

- Is it necessary to update the existing decision tree at current time t?
- Is it necessary to update the existing process control charts at current time t?

These two questions have to be answered every time a new data point is plotted on the process control chart ($\overline{X}$ and R charts). The decisions are made by the rules defined in the logical controller sub-module of the process controller module. These rules are defined in Tables 3 and 4 and are adapted from the literature (see, for example, DeVor *et al.* (1992)).

In Figure 5, we illustrate the data points plotted on the $\overline{X}$ chart. The horizontal axis represents the time and the vertical axis is the average tardiness (i.e. performance measure). When the system continues, the Y_i values (average tardiness per scheduling period) are collected by the on-line controller at the end of each scheduling period. These observations are then grouped in size of five to create the X_i values (average of average tardiness).

Table 2. New rule selection symptoms.

Abbreviation	Name	Description
BSP	Beginning of each scheduling period	Triggers the selection of a new DR at the beginning of each new scheduling period
MP	Monitoring points	Triggers the selection of a new DR at the monitoring points whenever necessary

Downloaded by [Bilkent University] at 02:37 13 November 2017

That is,

$$Y_i = \frac{\text{total tardiness in period } i}{\text{number of tardy jobs in period } i}, \tag{1}$$

$$X_i = \frac{Y_{i-2} + Y_{i-1} + Y_i + Y_{i+1} + Y_{i+2}}{5}. \tag{2}$$

Table 3. Update only decision tree rules.

Signal	Definition	Apply to
Extreme points	$\overline{X}_i$ or R_i points that fall beyond the control limits of the $\overline{X}$ and R charts, respectively	$\overline{X}$ and R charts
Zone A signal	Two out of three $\overline{X}_i$ points in Zone A (between 2σ and 3σ) or beyond	$\overline{X}$ chart only
Zone B signal	Four out of five $\overline{X}_i$ points in Zone B (between σ and 2σ) or beyond	$\overline{X}$ chart only

Table 4. Update both the decision tree and the process control chart rules.

Signal	Definition	Apply to
Eight successive points	Eight or more successive points strictly above or below the centerline	$\overline{X}$ and R charts
Two successive signals from Rule Set 1	Two successive occurrences of 'update only decision tree signals'	$\overline{X}$ and R charts

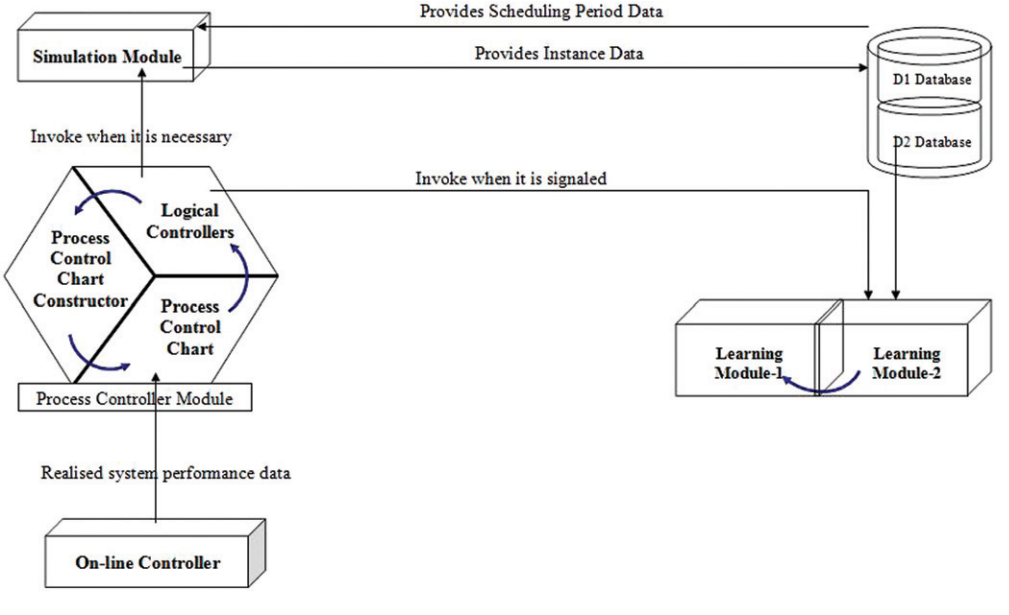

Figure 5. Operation of the $\overline{X}$ chart.

Downloaded by [Bilkent University] at 02:37 13 November 2017

Similarly, we plot the R_i values on the R chart, which are calculated as

$$R_i = Y_{\text{largest}} - Y_{\text{smallest}}, \quad \text{where } Y_{\text{largest}},\ Y_{\text{smallest}} \in \{Y_{i-2}, Y_{i-1}, Y_i, Y_{i+1}, Y_{i+2}\}. \tag{3}$$

The number of Y_i values grouped to generate X_i and R_i values is a parameter of the algorithm. We set the level of this parameter to five in our experiments to give the current decision tree a chance of survival. In other words, the performance of the decision tree is judged in a reasonable time period without leading to nervousness. In such settings where data collection is not costly, the value of this parameter can be increased to further smooth out the randomness in the data.

In Figure 5, the first circled point is captured by the extreme point signal given in Table 3. Therefore, a new decision tree is constructed at this point and the system continues with the updated decision tree. The second and third circled points are captured by zone A and extreme point signals, respectively, and the decision tree is also updated at these points. The fourth circled point in Figure 5 (i.e. X_{14}) is captured by the 'two successive signals from Rule Set 1' signal given in Table 4. Since two successive updates of the decision tree do not bring the process into balance, we conclude that the mean and/or variance of the process has shifted and thus we update the control charts as well as the decision tree. By this mechanism, the proposed scheduling system controls and updates itself to disturbances and survives over time.

Downloaded by [Bilkent University] at 02:37 13 November 2017

2.1 *Defining the system attributes*

The learning module of the system generates a decision tree that learns from and relies on the manufacturing system characteristics. Decisions on selecting dispatching rules are given by the existing decision tree on-line. In such a system, the learning algorithm requires a number of attributes that can provide valuable information about the current manufacturing system conditions. These attributes therefore play a key role in the performance of the proposed system, since they affect the quality of the tree in the construction phase as well as in the decision phase (i.e. selection of the right DRs from the decision tree for a scheduling period). Hence, appropriate attributes should be defined and used in such a way that they can represent a variety of important manufacturing system characteristics. In this section we present the guidelines that we follow when defining our attributes, and selecting the subset of attributes to be embodied into our system. Moreover, we provide an inventory of attributes we have used in this research in Appendix A, which we believe might be helpful in future research as well as industrial practice.

First of all, each attribute should capture some portion of the important information about the manufacturing system. In addition, it is important to define attributes so that their values can be calculated easily. This is due to the fact that our proposed system is an on-line scheduling system, and hence the time required to select a new dispatching rule should be negligible. Moreover, when setting the values of the attributes at any time epoch t, all we can use is the available information at that time, such as the number of jobs currently in the system, processing times and due-dates of the jobs, the realised performance of the system in the last scheduling period, etc. Based on these observations, we define a number of attributes such as total remaining processing time, maximum queue length at time t, average remaining time until due-date, etc. These attributes represent the general characteristics of the manufacturing system and its status in time. In addition, we

also take into account the characteristics and dynamics of the candidate dispatching rules and try to determine the conditions under which a specific rule performs well or poorly. In light of this idea, we define a number of attributes for each dispatching rule that might be helpful to differentiate that rule from the others. For instance, an attribute named '$n_LongProcTime$', which is the number of jobs with longer processing times than the average processing time of all jobs, is defined to distinguish the shortest processing time (SPT) rule from the others. The idea behind this is as follows: if there are many jobs with long processing time requirements, then the probability that new arriving jobs have a shorter processing time requirement than the existing jobs will be higher. This implies that the existing jobs, which have long processing times, will most probably be scheduled too late under SPT, resulting in a large average tardiness value.

After defining a number of attributes, the problem is to choose the most representative subset of attributes. To decide on this subset, we first performed a preliminary investigation on the behaviour of attribute subsets and gained the following valuable insights.

(1) Increasing the number of attributes in the subset does not necessarily improve the quality of the decision tree.
(2) The effect of each attribute in the subset on the performance of the generated decision tree also depends on the other attributes in the subset. In other words, their performances are correlated.
(3) The performance of the generated tree deteriorates when the attribute subset contains *maximum relative machine workload* and *completed processing times percentage*, Attributes 2 and 3 in Appendix A, respectively.

Our first observation has also been pointed out by other researchers in the literature (see, for instance, Shiue and Guh (2006)). Based on these observations we disregard Attributes 2 and 3 in Appendix A from further consideration. However, we still have 24 remaining attributes to consider for selection and considering every combination of these attributes requires testing of 2^{24} attribute sets, which is impractical. Therefore, we ignore the second observation above and assume that the effect of each attribute on the quality of the decision tree is independent of the other attributes in that set, and use a backward step selection heuristic to select the subset of attributes to be used.

At this point we find it necessary to make some remarks about defining and selecting system attributes. First of all, in most practical real-life applications of data mining, the attribute set is generally given as an input, and therefore there is no need to put any effort into 'defining an attribute set'. However, in our application, there is no 'natural' attribute set given to use as predictors. Therefore, we need to artificially define these attributes, which are potentially capable of capturing the system characteristics efficiently. Next, we need to select the best subset of these attributes as our predictors of the response variables. Unfortunately, the prediction performance of an attribute also depends on the other attributes in the subset. To the best of our knowledge, there is no other method than complete enumeration to find the optimal attribute subset. Therefore, we use a heuristic approach, sacrificing optimality, and look for a set of attributes that is the best of many attribute combinations considered within the search space of the heuristic. Also, the heuristic optimisation technique proposed by Shiue and Guh (2005, 2006) for attribute selection might alternatively be used to further enhance the performance of the proposed system. However, we should also note the fact that even if we seek to find the optimal

Downloaded by [Bilkent University] at 02:37 13 November 2017

attribute combination, there is no guarantee that a better set cannot exist, since, in theory, one can fabricate infinitely many different attributes.

3. Experimental settings

The simulation experiments are carried out under the following assumptions.

(1) The problem considered is a classical job shop problem with four machines as given by Baker (1984).
(2) There is no machine breakdown in the system.
(3) There is a set of candidate dispatching rules (CDR) that can be used (i.e. shortest processing time (SPT), modified due-date (MDD), modified operation due-date (MOD) and operation due-date (ODD)).

Downloaded by [Bilkent University] at 02:37 13 November 2017

When explaining the experimental results, we use three performance measures, called Multi-pass Performance (*MultiPass*), Best Performance (*BestPerf*) and the Learning Performance (*LearnPerf*). They are all measured in terms of average tardiness and defined in the following paragraphs.

Assume that we have two simulation models of the same manufacturing system, called SM1 and SM2. SM1 will represent the real-life and SM2 will represent the simulation environment, which is the imitation of SM1. Note that SM1 is the simulation model used in the on-line controller and SM2 is the multi-pass scheduling simulator, which is used to compare the performance of our proposed system with the performance of multi-pass scheduling. Since the random events occurring in real life differ from the simulated environment, these two models operate under different random number seeds. For example, SM1 uses random number seed 1 and SM2 uses seed 2. Now, assume a third simulation model, SM3, which also uses seed 1. We can think of SM3 as the playback of the realised events that occurred in SM1 in scheduling period n. Therefore, SM3 can be run for scheduling period j only if the realisation of period j in SM1 (real life) is completed. These three simulation models help us to measure the three performances we need.

Figure 6(a) shows how we measure *MultiPass* for scheduling period j. At the beginning of period j, the system state of SM2 is set equal to the system state of SM1. Then SM2 is run for each candidate dispatching rule (i.e. SPT, MDD, ODD, MOD) and r^*, the rule resulting with the minimum average tardiness value, is selected to be used in scheduling period j. r^* is passed to SM1 to realise its actual performance. At the end of scheduling period j, the realised average tardiness value is our *MultiPass* value for scheduling period j. In this sense, *MultiPass* is the average tardiness value achieved by the decisions of a multi-pass scheduling simulator.

BestPerf is the minimum average tardiness value that can ever be achieved for a scheduling period, say period j, by using any rule given in the candidate rule set. In other words, it is the best average tardiness value that we can achieve in period j subject to the parameter values of the system, such as the scheduling and monitoring period lengths, the candidate dispatching rules, etc. We can calculate this value for a scheduling period j if the realisation of period j is already completed by SM1. Then we can impose the same realisation of the random events on SM3 to answer the question: what would have been the average tardiness values if we had used dispatching rule SPT (or MDD or ODD or MOD) in period j? Then we simply set the value of *BestPerf* to the minimum of those

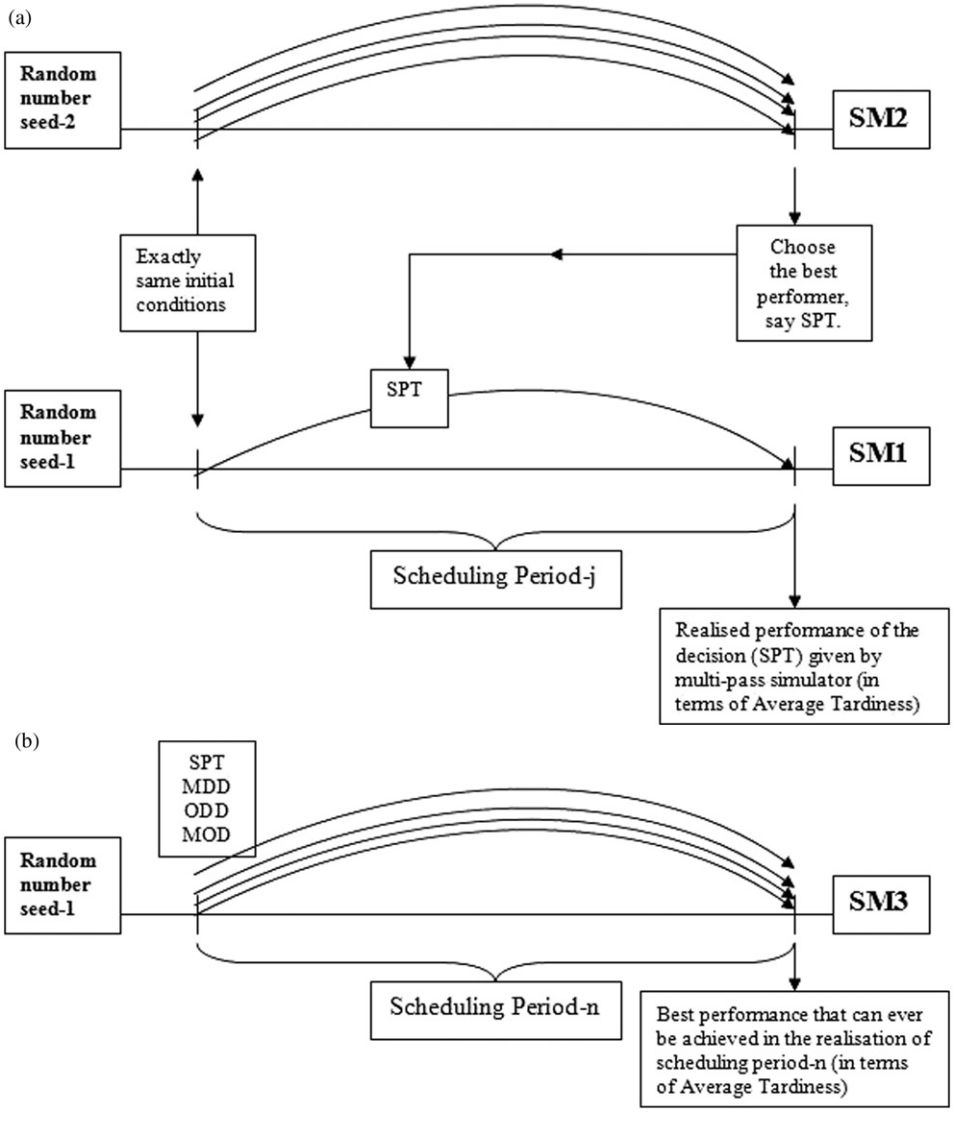

Figure 6. Performance measures. (a) Determination of multi-pass performance (*MultiPass*). (b) Determination of best Performance (*BestPerf*).

tardiness values (see Figure 6(b)). We measure this performance value to see how far our proposed system's performance (*LearnPerf*) and multi-pass scheduling simulator performance (*MultiPass*) are away from the ideal. Note that, *BestPerf* gives a lower bound for the other two performance functions.

Finally, the learning performance in period *j*, *LearnPerf*, is the realised average tardiness value of the rule selected by the decision tree (see Figure 7). That is, we request a dispatching rule from the decision tree at the beginning of scheduling period *j* based on the current values of the system attributes. This rule is used during period *j* and the average tardiness value is computed. Since this is the real performance of the dispatching rule

Downloaded by [Bilkent University] at 02:37 13 November 2017

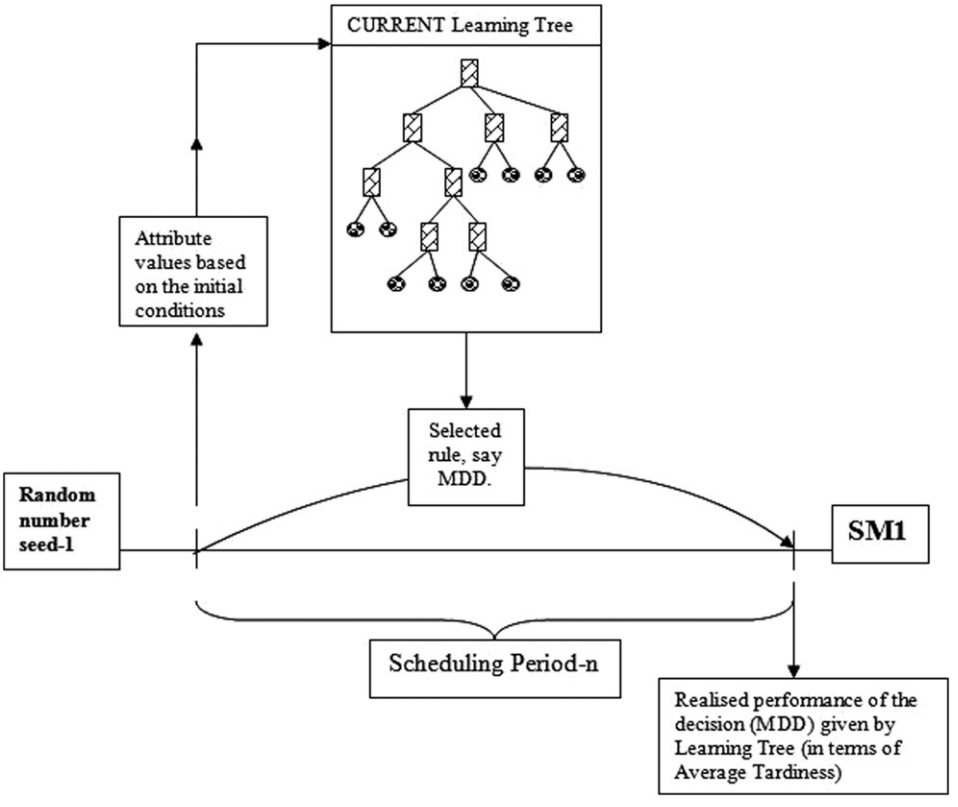

Figure 7. Performance measures: determination of the learning performance (*LearnPerf*).

selected by the tree, realisation of the rule is carried out by SM1. In the experiments, *LearnPerf* represents the performance of our proposed system.

In the simulation experiments, two levels of utilisation (i.e. low and high) and two levels of due-date tightness (i.e. loose and tight) are considered. The two levels of utilisation are 80% and 90%. Due-dates are set by using the total work (TWK) due-date assignment rule. The high and low levels are set in such a way that the percentage of tardy (PT) jobs is approximately 10% and 40% under the FCFS rule for the loose and tight due-date cases, respectively.

4. Computational results

In an early research study, Metan and Sabuncuoglu (2005) investigated the impact of various decision variables such as the scheduling period length (SPL) and the monitoring period length (MPL) on system performance. In this section we refer to their experimental results when fine-tuning the same experimental parameters.

4.1 *Job shop scheduling using a static decision tree*

In this section, we use our learning-based scheduling system in a job shop environment (Baker 1984) and measure the performance of the proposed system when the tree is

Downloaded by [Bilkent University] at 02:37 13 November 2017

Table 5. Experimental design of scheduling using a static decision tree.

Factors	Levels					
Due-date tightness	Tight					
Dispatching rule set	{SPT, MDD, ODD}			{SPT, MDD, ODD, MOD}		
Utilisation	80%			90%		
SPL	1000			7500		
MPL	250	500	1000	500	2500	7500
β	0.2	1	–	0.2	1	–

Table 6. Summary of performance values for Rule Set {SPT, MDD, ODD}.

Utilisation (%)	MPL	*BestPerf*	*LearnPerf*	*MultiPass*	Single pass	Δ_1 (%)	Δ_2 (%)
80	250	0.648	0.799	0.894	0.905	23.3	37.96
80	500	0.655	0.876	0.896	0.905	33.74	36.79
80	1000	0.679	0.891	0.895	0.905	31.22	31.81
90	500	1.1	1.383	1.494	1.545	25.72	35.81
90	2500	1.139	1.487	1.532	1.545	30.55	34.5
90	7500	1.196	1.51	1.514	1.545	26.25	26.58

$$\Delta_1 = 100\frac{LearnPerf - BestPerf}{BestPerf}, \quad \Delta_2 = 100\frac{MultiPass - BestPerf}{BestPerf}.$$

constructed only once. That is, the decision tree is not updated over time. For that reason, we call this application scheduling with a static decision tree.

The nested experimental design is given in Table 5. Two different dispatching rule sets are also considered in the experiments, where one of the sets contains MOD and the other set does not. In the simulation experiments, we take 20 replications for each experimental condition and each replication is composed of two phases: the Warm-up Phase and the Testing Phase. In the warm-up phase we generate the necessary instance data for our learning algorithm to construct the decision tree. This phase is composed of 2000 scheduling periods, which provides us a training data set that contains 2000 instance data (i.e. each scheduling period provides one instance data). At the end of this warm-up phase, a decision tree is constructed using this training data set. Then the second phase, the testing phase, starts. In the testing phase, the dispatching rules for each scheduling period are selected from the decision tree. This phase also contains 2000 scheduling periods for each replication and the statistics (i.e. *BestPerf*, *LearnPerf* and *MultiPass*) are collected in this phase.

Tables 6 through 8 summarise the experimental results. As mentioned before, *BestPerf* gives the lower bounds on both *MultiPass* and *LearnPerf*. In all of the experimental conditions, our learning-based scheduling system performs better than the simulation-based multi-pass scheduling (see Tables 6 and 7). However, *LearnPerf* approaches *MultiPass* as we increase the length of the monitoring period. At the extreme, when there is no monitoring at all, the performances of learning-based and simulation-based scheduling approaches become almost equal. This result is consistent with the findings reported by

Downloaded by [Bilkent University] at 02:37 13 November 2017

Table 7. Summary of performance values for Rule Set {SPT, MDD, ODD, MOD}.

Utilisation (%)	MPL	*BestPerf*	*LearnPerf*	*MultiPass*	Single pass	Δ_1 (%)	Δ_2 (%)
80	250	0.359	0.415	0.492	0.52	15.59	37.04
80	500	0.374	0.428	0.44	0.52	14.43	17.64
80	1000	0.383	0.435	0.44	0.52	13.57	14.88
90	500	0.52	0.568	0.684	0.704	9.23	31.53
90	2500	0.539	0.587	0.632	0.704	8.9	17.25
90	7500	0.559	0.591	0.595	0.704	5.72	6.44

$$\Delta_1 = 100\frac{LearnPerf - BestPerf}{BestPerf}, \quad \Delta_2 = 100\frac{MultiPass - BestPerf}{BestPerf}.$$

Metan and Sabuncuoglu (2005), where smaller values for MPL result in better average tardiness values for *BestPerf*. Therefore, it is very important to set the appropriate SPL, MPL and β values to obtain the maximum efficiency from the learning-based system.

For small values of MPL (i.e. 250 for 80% and 500 for 90% utilisation levels), the percentage difference between *LearnPerf* and *BestPerf* (Δ_1 in Tables 6 and 7) is considerably smaller (at least half) than the percentage difference between *MultiPass* and *BestPerf* (Δ_2 in Tables 6 and 7). Also, the percentage difference between *LearnPerf* and *BestPerf* is found to be better in the high-utilisation case (i.e. 90% utilisation) than in the low-utilisation case. This indicates that our proposed scheduling system provides improvement opportunities even more when the utilisation of the manufacturing system increases. As a final note, these performance values are statistically different at the 0.95 confidence level.

The main difference in the underlying experimental conditions, which the results given in Tables 6 and 7 are based upon, is the candidate dispatching rule set used. Our intention here is to investigate the impact of a very competitive dispatching rule, such as MOD, on the performances of different scheduling methods. When we compare the results given in Table 6 with the results in Table 7, we observe that inserting MOD into the existing candidate dispatching rule set (i.e. {SPT, MDD, ODD}) improves all the performance metrics (*BestPerf*, *LearnPerf* and *MultiPass*) significantly (almost 50% better results). Moreover, *LearnPerf* and *MultiPass* move closer to *BestPerf* (small Δ_1 and Δ_2 values in Table 7 compared with Table 6) when MOD is added to the rule set. Alternatively, we expect *LearnPerf* and *MultiPass* to move away from *BestPerf* when a low-quality DR is added to the candidate dispatching rule set.

We also keep track of the dispatching rule usage in the experiments for both *MultiPass*, *LearnPerf* and *BestPerf* (see Table 8). For small values of MPLs, the learning-based scheduling system uses a mix of dispatching rules very close to the best dispatching rule mix. On the other hand, for small MPLs, simulation-based multi-pass scheduling yields a significantly different mix than the best mix of dispatching rules. For large values of MPL, the dispatching rule mixes used by each scheduling system converge.

4.2 *Job shop scheduling using dynamically updated decision trees*

In the preceding section, we presented the experimental results for the new scheduling system when the decision tree was static. That is, we created a decision tree based on the

Downloaded by [Bilkent University] at 02:37 13 November 2017

Table 8. Dispatching rule mix (%) used by scheduling method.

	Utilisation:	80%	80%	80%	90%	90%	90%
	Rules/MPL:	250	500	1000	500	2500	750
Multi-pass	SPT	13	14	14	2	2	2
Multi-pass	MDD	34	34	34	58	58	58
Multi-pass	ODD	52	51	51	38	39	39
Learning	SPT	10	14	14	2	4	2
Learning	MDD	54	38	32	84	63	58
Learning	ODD	34	46	53	12	32	38
Best	SPT	9	13	14	2	3	2
Best	MDD	54	39	33	82	63	58
Best	ODD	36	47	52	14	33	38

Downloaded by [Bilkent University] at 02:37 13 November 2017

Table 9. Experimental design of scheduling using dynamically updated decision trees.

Factors	Levels
DR set	{MOD, MDD, ODD, SPT}, {MDD, ODD, SPT}
Arrival rate parameter sequence	{0.8, 0.9, 0.7, 0.9, 0.8}
Horizon lengths (number of SPs)	1000
Training data set	Full, partial (one-fifth of horizon length)
SM2 type	Reactive, non-reactive, partially reactive
Due-date tightness	Adjusted, not adjusted
(SPL, MPL, β)	{(1000, 250, 0.2), (7500, 500, 0.2)}

data collected from the manufacturing system during the warm-up period, and used this decision tree during the entire planning horizon to select DRs. In this sense, we have not yet utilised the statistical control charts to monitor the performance of the constructed decision tree. In this section, we now dynamically monitor the performance quality of the decision tree using the control charts and update the decision tree whenever the control charts signal an update. Therefore, the scheduling system generates a sequence of decision trees over time, which is governed by the statistical control charts and the set of decision tree update signals defined in Table 3.

In the simulation experiments, we consider a manufacturing system in which the internal parameters change over time (i.e. arrival rate of jobs, due-date tightness levels). The details of the experimental design are given in Table 9. We use five planning horizons, where each horizon contains 1000 scheduling periods. At the beginning of each horizon, we change some of the parameters of the manufacturing system. For example, in Table 9, the factor 'arrival rate parameter sequence' represents the value used as the job arrival rate during each horizon. Specifically, in horizons 1, 2, 3, 4 and 5, the jobs' inter-arrival times are exponentially distributed with parameters 0.8, 0.9, 0.7, 0.9 and 0.8, respectively. For the construction of the decision tree, we consider two different strategies, which are represented by the factor 'Training Data Set' in Table 9. When this factor is set to 'Full', the decision tree is constructed based upon all the accumulated data points since the beginning of the experiment. On the other hand, if the level is set to 'Partial', then the most recent 200 data points (one-fifth of a horizon length) are used each time the decision tree is updated.

We consider three levels for the SM2 type: reactive, non-reactive and partially reactive. When the SM2 type is 'reactive', the multi-pass simulation model is updated immediately when there is any parameter change in the actual manufacturing environment. In other words, if the arrival rate of the jobs changes in the real world, this information is made available for the multi-pass simulation model immediately. Intuitively, this is very difficult to achieve, if not impossible, in real-world implementation, because when any parameter of the manufacturing system changes it can be made available to the simulation model of the system only after a period of time. This time delay in updating the simulation parameters is almost inevitable, since detecting a shift in the parameters requires data collection and statistical analysis. For this reason, we also consider the scenario of the 'partially reactive' SM2 type. Under this scenario, the multi-pass simulation model is still updated for the arrival rate changes, but with some time delay and inaccuracy. Specifically, the arrival rate is updated with a delay of 200 scheduling periods (one-fifth of a horizon length) after the actual change in the real world takes place, and its value is set to values in the sequence 0.8, 0.875, 0.725, 0.875, 0.8 for horizons 1 through 5, respectively. As another extreme, we consider the scenario in which SM2 is a 'non-reactive' model. In this case the simulation model is not updated for any changes in the manufacturing environment. For example, when the arrival rate changes from 0.8 to 0.9 in the real world, the multi-pass simulation model (SM2) continues to operate under the initial arrival rate, which is 0.8.

Another factor considered in the experiments is the due-date tightness, which has two different levels: adjusted and not adjusted. For the adjusted case, we set the allowance factor k for setting the due-dates such that the percent tardy is always 40% under the FCFS rule. For the not adjusted case, the flow allowance factor is always set to 5.5. Therefore, the first case corresponds to a policy such that the manufacturing firm adjusts its due-date setting policy when the arrival rate of the jobs changes. Alternatively, in the second case, no action is taken for setting the due-dates of the jobs when the utilisation of the shop floor fluctuates.

From the results of Metan and Sabuncuoglu (2005), two levels, (1000, 250, 0.2) and (7500, 500, 0.2), are considered for SPL, MPL and β parameter combinations. At the beginning of each experiment, there is a warm-up period with a length of 200 scheduling periods to supply necessary initial data to the system to construct the first decision tree as well as the control charts. Statistics are collected after the warm-up period and each experimental condition is run for five consecutive time horizons, which is equivalent to a total of 5000 scheduling periods. Experimental results are summarised in Tables 10 and 11. Recall that *BestPerf* provides the lower bound on both *MultiPass* and *LearnPerf*.

Based on these results, the proposed scheduling system outperforms the simulation-based scheduling approach (*MultiPass*) in 38 experimental conditions out of 48 cases. In these 38 cases, *LearnPerf* is found to be closer to *BestPerf* by 2.34–40.87% compared with *MultiPass*. In two cases, both *MultiPass* and *LearnPerf* are found to be equal. In the remaining eight cases, simulation-based scheduling (*MultiPass*) performs slightly better than *LearnPerf* (i.e. between 1.68% and 7.83% better). However, in these cases, the SM2 type is reactive, which is an extremely idealistic condition to achieve in the real world.

We also cross-compare *LearnPerf* with itself for full and partial training data set cases. Using all available data points in constructing a decision tree always results in better performance (see Tables 10 and 11). At first glance, this seems to be counter-intuitive, because when parameters of the manufacturing system change, learning using only the most recent data points is expected to yield better performance. However, the results show

Downloaded by [Bilkent University] at 02:37 13 November 2017

Table 10. Average tardiness values for DR set {MDD, ODD, SPT}.

Due-date tightness	SM2 type[a]	Training data set {SPL, MPL, β}	Full {1000, 250, 0.2}	Full {7500, 500, 0.2}	Partial {1000, 250, 0.2}	Partial {7500, 500, 0.2}
Adjusted	Reactive	*MultiPass*	1.18	1.25	1.18	1.25
Adjusted	Reactive	*LearnPerf*	1.1	1.11	1.13	1.2
Adjusted	Reactive	*BestPerf*	0.98	1.02	0.98	1.02
Adjusted	Non-reactive	*MultiPass*	1.37	1.52	1.37	1.52
Adjusted	Non-reactive	*LearnPerf*	1.1	1.11	1.13	1.2
Adjusted	Non-reactive	*BestPerf*	0.98	1.02	0.98	1.02
Adjusted	P. reactive	*MultiPass*	1.25	1.32	1.25	1.32
Adjusted	P. reactive	*LearnPerf*	1.1	1.11	1.13	1.20
Adjusted	P. reactive	*BestPerf*	0.98	1.02	0.98	1.02
Not adjusted	Reactive	*MultiPass*	2.38	1.79	2.38	1.75
Not adjusted	Reactive	*LearnPerf*	2.30	1.75	2.42	1.90
Not adjusted	Reactive	*BestPerf*	2.15	1.71	2.15	1.71
Not adjusted	Non-reactive	*MultiPass*	2.59	2.26	2.59	2.26
Not adjusted	Non-reactive	*LearnPerf*	2.30	1.75	2.42	1.90
Not adjusted	Non-reactive	*BestPerf*	2.15	1.71	2.15	1.71
Not adjusted	P. Reactive	*MultiPass*	2.49	2.02	2.49	2.02
Not adjusted	P. Reactive	*LearnPerf*	2.30	1.75	2.42	1.90
Not adjusted	P. Reactive	*BestPerf*	2.15	1.71	2.15	1.71

[a]P. reactive is the partially reactive setting.

Table 11. Average tardiness values for DR set {MDD, ODD, SPT, MOD}.

Due-date tightness	SM2 type[a]	Training data set {SPL, MPL, β}	Full {1000, 250, 0.2}	Full {7500, 500, 0.2}	Partial {1000, 250, 0.2}	Partial {7500, 500, 0.2}
Adjusted	Reactive	*MultiPass*	0.81	0.65	0.81	0.65
Adjusted	Reactive	*LearnPerf*	0.81	0.65	0.82	0.68
Adjusted	Reactive	*BestPerf*	0.71	0.60	0.71	0.60
Adjusted	Non-reactive	*MultiPass*	0.96	0.73	0.96	0.73
Adjusted	Non-reactive	*LearnPerf*	0.81	0.65	0.82	0.68
Adjusted	Non-reactive	*BestPerf*	0.71	0.60	0.71	0.60
Adjusted	P. reactive	*MultiPass*	0.87	0.69	0.87	0.69
Adjusted	P. reactive	*LearnPerf*	0.81	0.65	0.82	0.68
Adjusted	P. reactive	*BestPerf*	0.71	0.60	0.71	0.60
Not adjusted	Reactive	*MultiPass*	1.52	1.20	1.52	1.20
Not adjusted	Reactive	*LearnPerf*	1.20	1.15	1.61	1.27
Not adjusted	Reactive	*BestPerf*	1.15	1.10	1.15	1.10
Not adjusted	Non-reactive	*MultiPass*	1.67	1.35	1.67	1.35
Not adjusted	Non-reactive	*LearnPerf*	1.20	1.15	1.61	1.27
Not adjusted	Non-reactive	*BestPerf*	1.15	1.10	1.15	1.10
Not adjusted	P. reactive	*MultiPass*	1.60	1.26	1.60	1.26
Not adjusted	P. reactive	*LearnPerf*	1.20	1.15	1.61	1.27
Not adjusted	P. reactive	*BestPerf*	1.15	1.10	1.15	1.10

[a]P. reactive is the partially reactive setting.

Downloaded by [Bilkent University] at 02:37 13 November 2017

that the learning algorithm benefits from the availability of past data as well as recent data.

For the rule set {MOD, MDD, ODD, SPT}, regardless of the type of training data set used (i.e. full or partial), the parameter combination (7500, 500, 0.2) always results in better performance for *LearnPerf* compared with the performance of the combination (1000, 250, 0.2) (Table 11). The reason for the combination (7500, 500, 0.2) yielding better results when MOD is in the DR set is simply due to the performance of MOD dominating the performance of the other rules when it is used for a long period of time.

As stated before, partially reactive SM2 is a more realistic case, where the simulation model used for the scheduling decisions of the multi-pass approach is updated with some time delay and inaccuracy. Such imperfections may originate from delayed detection of the parameter changes in the actual manufacturing environment. Therefore, the comparison of the learning-based (*LearnPerf*) and the simulation-based (*MultiPass*) systems for this factor level is of special importance. When the SM2 type is partially reactive, *LearnPerf* is better than *MultiPass* in 14 cases out of 16. Specifically, compared with *MultiPass* in these 14 cases, *LearnPerf* is found to be much closer to *BestPerf* by 3.26–34.79%. In the remaining two cases, *MultiPass* is closer to *BestPerf* only 0.9% more than *LearnPerf*, which means they are practically equal.

Downloaded by [Bilkent University] at 02:37 13 November 2017

4.3 *Extracting knowledge from the decision tree*

In addition to the performance improvement that can be achieved by using the proposed scheduling system, there is another advantage of this system: *extracting knowledge to collect expertise on job shop control*. Table 12 presents a sample decision tree generated by the proposed system. The size of the tree reaches as small as two and as many as eight decision nodes before stopping at a leaf node. Thirteen unique attributes are used in the decision tree. The actual attribute names are replaced by acronyms in order to fit the decision tree on a single page. However, we preserve the numbering of each attribute so that they are consistent with the attribute list provided in the appendix. The classification variable is the dispatching rule and it has three possible values in this example, which are SPT, MDD, and ODD. The decision tree originates from node 0, which has a total of 200 observations. The three numbers in parentheses under node 0 represent the number of observations associated with each class value. In other words, there are 20, 113 and 67 observations in the training data set having SPT, MDD and ODD as their associated class values, respectively. The same numbering scheme is used at each branching node. The sum of observations at child nodes is equal to the total number of observations at the immediate parent node.

First, branching is performed over the attribute *MeanTardiness* (ATT26), which is the last period's average tardiness value. This result is also consistent over different decision trees generated in different experimental settings. This observation suggests that the previous period's average tardiness value carries the most relevant information towards making a decision on the next period's dispatching rule selection. Branching on this attribute is also in parallel with our objective function, minimising the average tardiness. Since a large value of the previous period's average tardiness value might strongly indicate a large average tardiness value for the next period, the decision tree can better respond to the situation by choosing the most appropriate dispatching rule. For instance, when we look at the mix of best dispatching rules when the previous period's average tardiness is

Downloaded by [Bilkent University] at 02:37 13 November 2017

Table 12. A sample decision tree generated by the proposed system.

Node 0	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9
(20,113,67)	ATT26 = 0								
–	(7,60,62)	ATT11 ≤ 29							
–	–	(7,60,51)	ATT9 ≤ 1						
–	–	–	(6,16,25)	ATT1 ≤ 15					
–	–	–	–	(6,12,25)	ATT14 ≤ 695				
–	–	–	–	–	(3,12,25)	ATT5 = BSP			
–	–	–	–	–	–	(3,5,6)	ATT18 ≤ 55		
–	–	–	–	–	–	–	(1,5,6)	ATT7 ≤ 4.27	
–	–	–	–	–	–	–	–	(0,5,1)	: MDD
–	–	–	–	–	–	–	+	ATT7 > 4.27	
–	–	–	–	–	–	–		(1,0,5)	: ODD
–	–	–	–	–	–	+	ATT18 > 55		
–	–	–	–	–	–		(2,0,0)	: SPT	
–	–	–	–	–	+	ATT5 = MP			
–	–	–	–	–		(0,7,19)	: ODD		
–	–	–	–	+	ATT14 > 695				
–	–	–	–		(3,0,0)	: SPT			
–	–	–	+	ATT1 > 15					
–	–	–		(0,4,0)	: MDD				
–	–	+	ATT9 > 1						
–	–		(1,44,26)	ATT4 ≤ 1489					
–	–		–	(1,30,7)	ATT23 ≤ 47				
–	–		–	–	(1,12,7)	ATT13 ≤ 97			
–	–		–	–	–	(1,6,7)	ATT19 ≤ 106		
–	–		–	–	–	–	(0,6,3)	ATT7 ≤ 4.02	
–	–		–	–	–	–	–	(0,6,0)	: MDD
–	–		–	–	–	–	+	ATT7 > 4.02	
–	–		–	–	–	–		(0,0,3)	: ODD
–	–		–	–	–	+	ATT19 > 106		
–	–		–	–	–		(1,0,4)	: ODD	
–	–		–	–	+	ATT13 > 97			
–	–		–	–		(0,6,0)	: MDD		

(*continued*)

Downloaded by [Bilkent University] at 02:37 13 November 2017

Table 12. Continued.

Node 0	Node 1	Node 2	Node 3	Node 4	Node 5	Node 6	Node 7	Node 8	Node 9
–	–		–	+	ATT23 > 47				
–	–		–		(0,18,0)	: MDD			
–	–		+	ATT4 > 1489					
–	–			(0,14,19)	ATT14 ≤ 456				
–	–			–	(0,3,15)	: ODD			
–	–			+	ATT14 > 456				
–	–				(0,11,4)	: MDD			
–	+	ATT11 > 29							
–		(0,0,11)	: ODD						
+	ATT26 > 0								
	(13,53,5)	ATT6 ≤ 33							
	–	(9,10,2)	ATT18 ≤ 33						
	–	–	(0,0,2)	: ODD					
	–	+	ATT18 > 33						
	–		(9,10,0)	ATT1 ≤ 12					
	–		–	(5,10,0)	ATT7 ≤ 4.53				
	–		–	–	(1,10,0)	: MDD			
	–		–	+	ATT7 > 4.53				
	–		–		(4,0,0)	: SPT			
	–		+	ATT1 > 12					
	–			(4,0,0)	: SPT				
	+	ATT6 > 33							
		(4,43,3)	: MDD						

zero (i.e. numbers given under the branch ATT26=0), we observe that each of the dispatching rules (i.e. MDD and ODD) performs the best approximately half of the time. However, when the previous period's average tardiness is greater than zero, ODD loses its attractiveness, SPT gains more importance and MDD becomes the dominant dispatching rule.

At the second level of the decision tree, we see further branching on *AvgRemainTimeDDate* (ATT11), which is the average remaining time until the due-date, and *TotRemainProcTime* (ATT6), which captures the total remaining processing time of all jobs. Let us first discuss the branching on *AvgRemainTimeDDate* under the parent branch of ATT26 = 0. When the average remaining time until the due-date exceeds a threshold, the decision tree clearly identifies ODD as the best dispatching rule for the next time period. However, if the system operates under a tighter due-date schedule (i.e. ATT11 $\leq$ threshold value), the best dispatching rule is not immediately clear at this level and further branching is necessary. However, we observe under this scenario that the best dispatching rule mix (i.e. (SPT, MDD, ODD)=(7, 60, 51)) favours MDD slightly more than the other candidates as the system operates under tighter due-dates.

The second branching at the same level is performed on the attribute that captures the total remaining processing time of all jobs (i.e. *TotRemainProcTime* (ATT6)). When the total processing requirements of the jobs currently in the system are greater than a threshold, and given the fact that the last period's average tardiness is already greater than zero, the best dispatching rule is identified to be MDD by the decision tree. However, if the jobs in the system do not have larger processing time requirements, SPT and ODD become competitive with MDD in some cases, which are identified by further deep branches.

Another interesting attribute used in the decision tree is *AvgRemainProcTime* (ATT7), which is the acronym for the average remaining processing time. This attribute's value is simply calculated by dividing the total remaining processing time by the number of jobs currently in the system. We observe branching over this attribute at three different places in our decision tree right before we reach a leaf. In two of these three cases, it differentiates MDD and ODD at the leaf and in another case it differentiates MDD and SPT.

As is clear from the above discussion, it is a very challenging task, if not impossible, to extract such knowledge to gain such expertise on job shop control without using a scheduling system similar to the one proposed in this research. We know from previous research in the literature that there is no single dispatching rule that performs the best under all conditions. However, we can extract the information that is hidden and dissolved in the data concerning the conditions and circumstances that make a dispatching rule more efficient and desirable over other rules. This expertise can be utilised in many applications such as designing new and more efficient dispatching rules, categorising existing rules based on the conditions under which they efficiently operate, and for providing more insights into job shop scheduling to practitioners in industry.

5. Conclusion

We have developed a novel scheduling approach that combines simulation, data mining, and statistical process control (SPC) principles. Our experiments have shown that its performance compares very well with approaches such as simulation-based multi-pass that are known to be very efficient. The resulting scheduling system can be implemented on the shop floor. Another alternative benefit offered by the proposed scheduling system is that

Downloaded by [Bilkent University] at 02:37 13 November 2017

production managers can use the knowledge extracted in the form of decision trees to make decisions. Such knowledge highlights the important factors to take into account when making decisions and what is advisable to do when, depending on the operating conditions.

We have developed a data-mining-based scheduling system that extracts knowledge from the data that comes from a job shop manufacturing configuration. We have compared the performance of the proposed scheduling system (using both a static and a dynamically updated decision tree) via simulation and emphasised the conditions in which the proposed system works better than the simulation-based multi-pass scheduling method. This knowledge provides extremely valuable insights into job shop control and the proposed system is an effective tool for collecting such expertise. Finally, we have provided an extensive list of manufacturing attributes in the appendix. These attributes describe a wide range of shop floor conditions and can be used to mine the information hidden beneath the vast amount of data in different manufacturing settings.

Our key insights are as follows.

(i) It is important to set the monitoring policy appropriately. Monitoring too frequently prevents dispatching rules from performing their functions and monitoring very infrequently overlooks the underperformers.
(ii) Selecting the right rules for the candidate dispatching rule set is also important for achieving further improved performance.
(iii) When creating the decision tree, utilising all historical data points, rather than using only the most recent subset of observations, improves the performance.
(iv) The proposed scheduling system provides an attractive and cost-efficient method for capturing and adjusting to changes in manufacturing environments using SPC concepts.

All the results presented in this paper should be interpreted with reference to the assumptions and experimental conditions and the job shop system studied. One future research direction would be to study the performance of the proposed scheduling system in different manufacturing configurations. Such an investigation would potentially help extract common knowledge that applies to a variety of settings. The general principles of using simulation, data mining and certain SPC concepts can be used for other types of dynamic scheduling problems such as flowshop problems and AGV dispatching. Other future research avenues include the following. (i) New and more advanced attributes can be defined and their performances can be investigated for alternative manufacturing conditions, preferably in real-life settings. (ii) We have provided a heuristic technique for selecting a good combination of attributes that can capture the information hidden within the manufacturing data. More powerful techniques that guarantee the selection of optimal or near-optimal combinations of the attributes are needed to be developed. We have identified two recent studies (Shiue and Guh 2005, 2006) in this direction and more research on this problem is needed. This would significantly enhance the decision tree quality and in turn improve the system performance. (iii) Alternative data mining approaches can be substituted for the C4.5 algorithm and their performances can be compared. This will also provide a valuable guideline to practitioners in choosing the right tool among the many competing learning algorithms for their applications. Investigating the performance of 'rules induction' techniques is of special importance since these techniques extract knowledge in terms of a list of simple 'if ... then ...' logical rules.

Downloaded by [Bilkent University] at 02:37 13 November 2017

References

Aissani, N., Beldjilali, B., and Trentesaux, D., 2008. Use of machine learning for continuous improvement of the real time heterarchical manufacturing control system performances. *International Journal of Industrial and Systems Engineering*, 3 (4), 474–497.

Baker, K.R., 1984. Sequencing rules and due-date assignments in a job shop. *Management Science*, 30 (9), 1093–1104.

Cho, H. and Wysk, R.A., 1993. A robust adaptive scheduler for an intelligent workstation controller. *International Journal of Production Research*, 31 (4), 771–789.

DeVor, R.E., Chang, T-h., and Sutherland, J.W., 1992. *Statistical quality design and control*. Engelwood Cliffs, NJ: Prentice Hall.

El-Bouri, A. and Shah, P., 2006. A neural network for dispatching rule selection in a job shop. *International Journal of Advanced Manufacturing Technology*, 31 (3–4), 342–349.

Geiger, C.D. and Uzsoy, R., 2006. Learning effective dispatching rules for batch processor scheduling. *International Journal of Production Research*, 46 (6), 1431–1454.

Geiger, C.D., Uzsoy, R., and Aytug, H., 2006. Rapid modeling and discovery of priority dispatching rules: An autonomous learning approach. *Journal of Scheduling*, 9 (1), 7–34.

Harding, J.A., *et al*., 2006. Data mining in manufacturing: A review. *Journal of Manufacturing Science and Engineering*, 128 (4), 969–976.

Huyet, A.L., 2006. Optimization and analysis aid via data-mining for simulated production systems. *European Journal of Operational Research*, 173 (3), 827–838.

Huyet, A.L. and Paris, J.L., 2004. Synergy between evolutionary optimization and induction graphs learning for simulated manufacturing systems. *International Journal of Production Research*, 42 (20), 4295–4313.

Ishii, N. and Talavage, J.J., 1991. A transient-based real-time scheduling algorithm in FMS. *International Journal of Production Research*, 29 (12), 2501–2520.

Ishii, N. and Talavage, J.J., 1994. A mixed dispatching rule approach in FMS scheduling. *International Journal of Flexible Manufacturing Systems*, 2 (6), 69–87.

Jeong, K.-C. and 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 (9), 2609–2626.

Kim, M.H. and Kim, Y.-D., 1994. Simulation-based real-time scheduling in a flexible manufacturing system. *Journal of Manufacturing Systems*, 13 (2), 85–93.

Kutanoglu, E. and Sabuncuoglu, I., 2001. Experimental investigation of iterative simulation-based scheduling in a dynamic and stochastic job shop. *Journal of Manufacturing Systems*, 20 (4), 264–279.

Lee, C.-Y., Piramuthu, S., and Tsai, Y.-K., 1997. Job shop scheduling with a genetic algorithm and machine learning. *International Journal of Production Research*, 35 (4), 1171–1191.

Li, X. and Olafsson, S., 2005. Discovering dispatching rules using data mining. *Journal of Scheduling*, 8 (6), 515–527.

Metan, G. and Sabuncuoglu, I., 2005. A simulation based learning mechanism for scheduling systems with continuous control and update structure. *Proceedings of the 2005 winter simulation conference*, 2148–2156.

Min, H.-S. and 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 (16), 3921–3941.

Pierreval, H., 1993. Neural network to select dynamic scheduling heuristics. *Journal of Decision Systems*, 2 (1), 173–190.

Pierreval, H. and Ralambondrainy, H., 1990. A simulation and learning technique for generating knowledge about manufacturing systems behavior. *Journal of the Operational Research Society*, 41 (6), 461–474.

Downloaded by [Bilkent University] at 02:37 13 November 2017

Pierreval, H. and Mebarki, N., 1997. Dynamic selection of dispatching rules for manufacturing system scheduling. *International Journal of Production Research*, 35 (6), 1575–1591.

Quinlan, J.R., 1993. *C4.5 programs for machine learning*. San Francisco, California: Morgan Kaufmann.

Sabuncuoglu, I. and Goren, S., 2003. A review of reactive scheduling research: proactive scheduling and new robustness and stability measures, Technical working paper. Department of Industrial Engineering, Bilkent University.

Sha, D.Y. and Liu, C.-H., 2005. Using data mining for due date assignment in a dynamic job shop environment. *International Journal of Advanced Manufacturing Technology*, 25 (11–12), 1164–1174.

Shiue, Y.-R. and Guh, R.-S., 2005. Learning-based multi-pass adaptive scheduling for a dynamic manufacturing cell environment. *Robotics and Computer-Integrated Manufacturing*, 22 (3), 203–216.

Shiue, Y.-R. and Guh, R.-S., 2006. The optimization of attribute selection in decision tree-based production control systems. *International Journal of Advanced Manufacturing Technology*, 28, 737–746.

Sun, R.-L., *et al.*, 2004. Iterative learning scheduling: a combination of optimization and dispatching rules. *Journal of Manufacturing Technology Management*, 15 (3), 298–305.

Tayanithi, P., Minivannan, S., and Banks, J., 1993a. A knowledge-based simulation architecture to analyze interruptions in a flexible manufacturing system. *Journal of Manufacturing Systems*, 11 (3), 195–214.

Tayanithi, P., Minivannan, S., and Banks, J., 1993b. Complexity reduction during interruption analysis in a flexible manufacturing system using knowledge-based on-line simulation. *Journal of Manufacturing Systems*, 12 (2), 153–169.

Wu, S.D. and Wysk, R.A., 1988. Multi-pass expert control system – a control/scheduling structure for flexible manufacturing cells. *Journal of Manufacturing Systems*, 7 (2), 107–120.

Wu, S.D. and Wysk, R.A., 1989. An application of discrete-event simulation to on-line control and scheduling in flexible manufacturing. *International Journal of Production Research*, 27 (9), 1603–1623.

Wu, X., *et al.*, 2008. Top 10 algorithms in data mining. *Knowledge and Information Systems*, 14 (1), 1–37.

Yildirim, M.B., *et al.*, 2006. Machine number, priority rule, and due date determination in flexible manufacturing systems using artificial neural networks. *Computers & Industrial Engineering*, 50 (1–2), 185–194.

Downloaded by [Bilkent University] at 02:37 13 November 2017

Appendix A: Notation and definition of the system attributes

Notation

M	set of machines, $M = \{1, 2, 3, 4\}$
m	number of machines in the manufacturing system, which is equal to the cardinality of M, $\|M\|$
Q	set of queues in front of the machines, $Q = \{1, 2, 3, 4\}$
I_q	set of jobs in queue q at time t, $q \in Q$
O_m	set of operations of all jobs in the system that is to be processed on machine m, $m \in M$
I_t	set of jobs in the system at time t
n	cardinality of I_t, that is the number of jobs in the system at time t
J_i	set of all operations of job i, $i \in I$
$\hat{J}_i$	set of all remaining operations of job i, $i \in I$, $\hat{J} \subseteq J$
r_i	release time of job i, $i \in I$
p_{ij}	processing time of operation j of job i, $i \in I$, $j \in J$
$p_{i,\mathrm{cur}}$	processing time of job i at the machine just after its current queue, $i \in I$, $j \in J$

$i_{\langle j \rangle}$ machinery location of the jth operation of job i, $i \in I$, $j \in J$
d_i due-date of job i, $i \in I$
d_{ij} due-date of operation j of job i, $i \in I$, $j \in J$
o_i number of operations of job i, $i \in I$
k flow allowance factor

Definition of the system attributes

Attribute 1: Number of jobs in the system (n_Jobs)

This attribute stores the number of jobs in the system at time t.

Attribute 2: The percentage of maximum relative machine workload (MaxRelativeMachLoad%)

$$MaxRelativeMachLoad\% = 100 \frac{\max_{q \in Q}(\sum_{i \in I_q} p_{i,\mathrm{cur}})}{\sum_{i \in I_t} \sum_{j \in \hat{J}} p_{ij}}\%. \tag{A1}$$

Attribute 3: Percentage of completed processing times (CompletedProcTime%)

$$CompletedProcTime\% = 100 \frac{\sum_{i \in I_t} \sum_{j \in J - \hat{J}} p_{ij}}{\sum_{i \in I_t} \sum_{j \in J} p_{ij}}\%. \tag{A2}$$

Attribute 4: Relative tightness ratio (RelativeTightness)

$$RelativeTightness = \frac{\bar{d}_t}{\bar{p}}, \quad \text{where } \bar{d}_t = \max\left\{0, \frac{\sum_{i \in I_t}(d_i - t)}{n}\right\}, \quad \bar{p} = \frac{\sum_{i \in I_t} \sum_{j \in \hat{J}} p_{ij}}{n}. \tag{A3}$$

Attribute 5: Rule updating signal (RuleUpdtSgnl)

$$RuleUpdtSgnl = \begin{cases} 0, & \text{if the signal is BSP,} \\ 1, & \text{if the signal is MP.} \end{cases} \tag{A4}$$

Attribute 6: Total remaining processing time (TotRemainProcTime)

$$TotRemainProcTime = \sum_{i \in I_t} \sum_{j \in \hat{J}} p_{ij}. \tag{A5}$$

Attribute 7: Average remaining processing time (AvgRemainProcTime)

$$AvgRemainProcTime = \frac{TotRemainProcTime}{n} = \frac{\sum_{i \in I_t} \sum_{j \in \hat{J}} p_{ij}}{n}. \tag{A6}$$

Downloaded by [Bilkent University] at 02:37 13 November 2017

Attribute 8: Average slack time (AvgSlackTime)

$$AvgSlackTime = \frac{\sum_{i\in I}(d_i - t) - \sum_{i\in I}\sum_{j\in \hat{J}} p_{ij}}{n}. \tag{A7}$$

Attribute 9: Average period queue length (AvgQueueLength)

$$AvgQueueLength = \frac{\sum_{q\in Q} QT_q(t - SPL, t)}{|Q|\, SPL}. \tag{A8}$$

Attribute 10: Maximum queue length at time t (MaxQueueLength)

$$MaxQueueLength = \max_{q\in Q}\{|I_q|\}, \quad \text{where } |I_q| \text{ is the cardinality of set } I_q. \tag{A9}$$

Attribute 11: Average remaining time until due-dates (AvgRemainTimeDDate)

$$AvgRemainTimeDDate = \frac{\sum_{i\in I}(d_i - t)}{n}. \tag{A10}$$

Attribute 12: Number of jobs with a long processing time (n_LongProcTime)

$$n\_LongProcTime = |L|, \text{ such that } L = \{i \mid p_i \geq \bar{p}, i \in I_t\}, \quad \text{where } \bar{p} = \frac{\sum_{i\in I}\sum_{j\in J} p_{ij}}{n}. \tag{A11}$$

Attribute 13: Percentage of jobs with long processing times (LongProcTime%)

$$LongProcTime\% = 100\frac{n\_LongProcTime}{n}\%. \tag{A12}$$

Attribute 14: Difference between the maximum and average processing times (ΔMaxAvgProcTime)

$$\Delta MaxAvgProcTime = \max_{i\in I}\left\{\sum_{j\in J} p_{ij}\right\} - \bar{p}, \quad \text{where } \bar{p} = \frac{\sum_{i\in I}\sum_{j\in J} p_{ij}}{n}. \tag{A13}$$

Downloaded by [Bilkent University] at 02:37 13 November 2017

Attribute 15: Percentage difference between maximum and average processing times (ΔMaxAvgProcTime%)

$$\Delta MaxAvgProcTime\% = 100\frac{\Delta MaxAvgProcTime}{\bar{p}}\%. \tag{A14}$$

Attribute 16: Maximum due-date (MaxDDate)

$$MaxDDate = \max\left\{0, \max_{i\in I}\{d_i\} - t\right\}. \tag{A15}$$

Attribute 17: Number of jobs with long due-dates (n_LongDDate)

$$n\_LongDDate = |S|, \quad \text{where } S = \{i \mid d_i \geq \bar{d}, i \in I\}, \quad \bar{d} = \frac{\sum_{i\in I} d_i}{n}. \tag{A16}$$

Attribute 18: Percentage of jobs with long due-dates (n%_LongDDate)

$$n\%\_LongDDate = 100\frac{|S|}{n}, \quad \text{where } S = \{i \mid d_i \geq \bar{d}, i \in I\}, \quad \bar{d} = \frac{\sum_{i\in I} d_i}{n}. \tag{A17}$$

Attribute 19: Maximum coefficient of variation of processing times of machines (MaxCvProcTimeMach)

$$MaxCvProcTimeMach = \max_{m\in M}\{cv_m\}, \tag{A18}$$

where

$$cv_m = \frac{\bar{p}_m}{\sigma_m}, \quad \bar{p}_m = \frac{\sum_{i\in I}\sum_{j\in O_m} p_{ij}}{|O_m|}, \quad \sigma_m = \sqrt{\frac{\sum_{i\in I}\sum_{j\in O_m}(p_{ij} - \bar{p}_m)^2}{|O_m| - 1}}. \tag{A19}$$

Attribute 20: Average coefficient of variation of the job processing times (AvgCvJobProcTimes)

$$AvgCvJobProcTimes = \frac{\sum_{i\in I} cv_i}{n}, \tag{A20}$$

where

$$cv_i = \frac{\bar{p}_i}{\sigma_i}, \quad \bar{p}_i = \frac{\sum_j p_{ij}}{|O_i|}, \quad \sigma_m = \sqrt{\frac{\sum_j (p_{ij} - \bar{p}_i)^2}{|O_i| - 1}}. \tag{A21}$$

Downloaded by [Bilkent University] at 02:37 13 November 2017

Attribute 21: Average variance of the job processing times (AvgVarJobProcTimes)

$$AvgVarJobProcTimes = \frac{\sum_{i \in I}(\sigma_i)^2}{n}, \quad \text{where } \sigma_i \text{ is calculated as in Attribute 20.} \tag{A22}$$

Attribute 22: Maximum coefficient of variation of the job processing times (MaxCvJobProcTimes)

$$MaxCvJobProcTimes = \max_{i \in I}\{cv_i\}. \tag{A23}$$

Attribute 23: Difference between the maximum and average coefficient of variation of the job processing times (ΔMax − AvgCv)

This attribute is simply the difference between Attribute 22 and Attribute 20.

Attribute 24: Number of jobs already tardy (n_Tardy)

$$n\_Tardy = |S|, \quad \text{where } S = \left\{ i \,\middle|\, \sum_{j \in \hat{J}} p_{ij} > d_i - t, i \in I_t \right\}. \tag{A24}$$

Attribute 25: Average remaining processing time of already tardy jobs (AvgRemainProcTime_Tardy)

$$AvgRemainProcTime\_Tardy = \frac{\sum_{i \in S}\sum_{j \in \hat{J}} p_{ij}}{|S|}. \tag{A25}$$

Attribute 26: Last period's average tardiness value (MeanTardiness)

The value of this attribute at time t is set to the mean tardiness (our performance measure) of the last scheduling period, which ended at time t. We define this attribute because the realised system performance at the last scheduling period may carry some information about the most appropriate dispatching rule for the next scheduling period.

Downloaded by [Bilkent University] at 02:37 13 November 2017