Journal of Scheduling **8:** 515–527, 2005.

DISCOVERING DISPATCHING RULES USING DATA MINING

XIAONAN LI AND SIGURDUR OLAFSSON

Department of Industrial Engineering, Iowa State University, 2019 Black Engineering, Ames, IA 50011

ABSTRACT

This paper introduces a novel methodology for generating scheduling rules using a data-driven approach. We show how to use data mining to discover previously unknown dispatching rules by applying the learning algorithms directly to production data. This approach involves preprocessing of historic scheduling data into an appropriate data file, discovery of key scheduling concepts, and representation of the data mining results in a way that enables its use for job scheduling. We also consider how by using this new approach unexpected knowledge and insights can be obtained, in a manner that would not be possible if an explicit model of the system or the basic scheduling rules had to be obtained beforehand. All of our results are illustrated via numerical examples and experiments on simulated data.

KEY WORDS: data mining, scheduling, decision trees, dispatching rules

1. INTRODUCTION

Production scheduling provides an important role in most manufacturing facilities and has received a great amount of attention from both the operations research and artificial intelligence communities. There is, however, some discontinuity between scheduling practice and the academic research of scheduling models and algorithms. In particular, for complex systems it may be difficult in practice to account for all relevant aspects in an optimization model or to elicit all relevant scheduling rules directly from an expert. In practice, on the other hand, scheduling is often done on an ad-hoc basis that does not rely solely on well-defined rules or principles, but also on the intuition and experience of the scheduler. When such knowledge is not explicitly captured, any model-based approach may fail to account for important considerations.

Data mining and knowledge discovery is an emerging area of research and applications that draws on machine learning and statistical methods to learn previously unknown and useful knowledge from examples in large databases. Data mining has made a significant impact on numerous industries, but its function in manufacturing and production scheduling in particular, has only received moderate attention to date. Its applicability in this area is evident, however, as the strengths of data mining lie where it is difficult or impossible to capture all aspects of a system a priori in a model, either because of its complexity or because of incomplete existing knowledge, and large volumes of data are generated by the system. Both situations commonly exist in production environments, which are often too complex for simple mathematical models to adequately capture all essential elements of the system, and much of the knowledge of the operation of the system may be implicit and would thus not be captured by the mathematical models.

Correspondence to: Sigurdur Olafsson, Department of Industrial Engineering, Iowa State University, 2019 Black Engineering, Ames, IA 50011. E-mail: olafsson@iastate.edu.

In this paper, we show how data mining on production data can be used to capture both explicit and implicit knowledge that is used to create production schedules. Using this process new dispatching rules are discovered that can then be applied to automate the scheduling function. Furthermore, hidden patterns discovered in the schedule generation may add insights not realized by the schedulers themselves, suggesting ways in which current scheduling practices can be improved.

The idea of this data mining approach to production scheduling is to complement more traditional operations research approaches. The data mining approach is particularly applicable for large, complex production environments, where the complexity makes it difficult to model the system explicitly. However, an implicit assumption is that it is worthwhile to capture the current practices from historical data. Furthermore, this approach does not seek to directly improve any scheduling performance measure. Thus, combining the data mining with some type of optimization approach that improves the discovered dispatching rules would be a natural extension of the work described in this paper.

The remainder of the paper is organized as follows. In Section 2, we briefly review some of the closely related research, and in particular, how machine learning has been used for scheduling in prior work. In Section 3, we show how decision tree data mining can be used to learn dispatching rules and give a general framework for a knowledge discovery in production data process. In Section 4, we evaluate how well such decision trees perform as dispatching rules, and consider the data engineering issue of attribute construction and selection. We show how this both improves the performance of the subsequent data mining models and provides important insights into how scheduling decision are made. Finally, Section 5 contains some concluding remarks and future research directions.

2. RELATION TO OTHER RESEARCH

Artificial intelligence methods for scheduling have received considerable attention over the past two decades (see e.g., Aytug et al., 1994; Kanet and Adelsberger, 1987; Kusiak and Chen, 1988; Noronha and Sarma, 1991; Priore et al., 2001). Such methods were developed in part to address the limitation imposed by the need for an explicit model of the system. Examples of such methods include neural networks (Jain and Meeran, 1998; Min, Yih, and Kim, 1998), induction (Shaw, Park, and Raman, 1992), hybrid approaches (Kim, Min, and Yih, 1998; Lee, Piramuthu, and Tsai, 1997), and unsupervised learning (Bowden and Bullington, 1996; Li and She, 1994). Of these various methods, the work that is most related to the present paper are inductive learning methods that have, for example, been used to select between several dispatching rules. Some of this work is described further below and more comprehensive surveys can be found in Aytug et al. (1994) and Priore et al. (2001).

In an early work, Nakasuka and Yoshida (1992) used empirical data to generate a binary decision automatically. The empirical data were obtained through iterative production line simulations and afterwards the binary decision tree decides in real time which rule to be used at decision points during the actual production operations. Later, Wang et al. (1995) proposed a scheduling system with inductive learning called the System Attribute Oriented Knowledge Based Scheduling System (SAOSS). A simulation model is again used to generate training examples. A continuous ID3 algorithm is then employed to induce decision rules for scheduling by converting corresponding decision trees into the hidden layers of a self-generated neural network. The connection weights

between hidden units of the self-generated neural network imply the scheduling heuristics, which are then formulated into scheduling rules.

Shaw, Park, and Raman (1992) also present an inductive learning approach integrated with simulation. Similarly to the work mentioned above, a simulation model is used to generate training examples for a given state of manufacturing process. The simulation runs are replicated for each scheduling rule considered. The rule that has the best performance is the class value for this set of attributes' values. As for the criteria for best performance, the scheduling objectives should be designed in advance. An ID3 algorithm is employed to describe the system pattern for selecting each scheduling rule. A decision tree is formulated and then it is translated into a set of patterns directed heuristics for scheduling.

In a later extension of this work, Piramuthu et al. (1993), and Piramuthu, Raman, and Shaw (1994) employ the C4.5 algorithm, a refinement of the ID3 algorithm with pruning capability. Improvement has been done in two aspects: (a) a different decision tree is constructed, for choosing chattering threshold values under different patterns. As a result, overreaction to filter noise arising from transient patterns will be avoided efficiently; (b) a critic module is employed to refine the decision tree. Periodically, the performance of the system is compared with that of an individual scheduling rule. If the performance of the system degraded, the decision tree needs to be refined by generating new training examples to certain over generalized concepts. In this way, the system can learn new knowledge incrementally.

As this sampling of prior work indicates, inductive learning in production scheduling has primarily been devoted to issues such as selecting the best dispatching rule by learning from simulated data. This, of course, assumes that all the dispatching rules are known in advance and that the performance of these rules can be accurately simulated. A notable exception to this is the recent work of Geiger, Uzsoy, and Aytug (2003), where a genetic algorithm is used to discover new dispatching rules in a flow shop environment. However, to the best of our knowledge data mining applied directly to production data to discover new and interesting dispatching rules has not been considered before.

3. LEARNING DISPATCHING RULES

As indicated by the literature in Section 2, a significant amount of work has been carried out related to the use of artificial intelligence in scheduling. However, the novelty of the approach advocated here is to apply data mining directly to production data to discover previously unknown meaningful patterns, such as new and interesting dispatching rules. The potential benefits of this approach include:

- The implicit knowledge of expert schedulers is discovered and can be used to generate future schedules with little or no direct involvement of such experts.
- Existing scheduling practices are generalized into explicit scheduling rules. These rules can then be applied both to situations that have occurred before and to new scenarios.
- In addition to the predictive scheduling rules that allow for dispatching of jobs, structural knowledge may be gained that leads to new rules that improve scheduling performance.

3.1. Framework for inductive learning

In this section, we propose a general framework for knowledge discovery in production data, and specifically data mining of dispatching rules. Thus, from the data mining perspective we specify

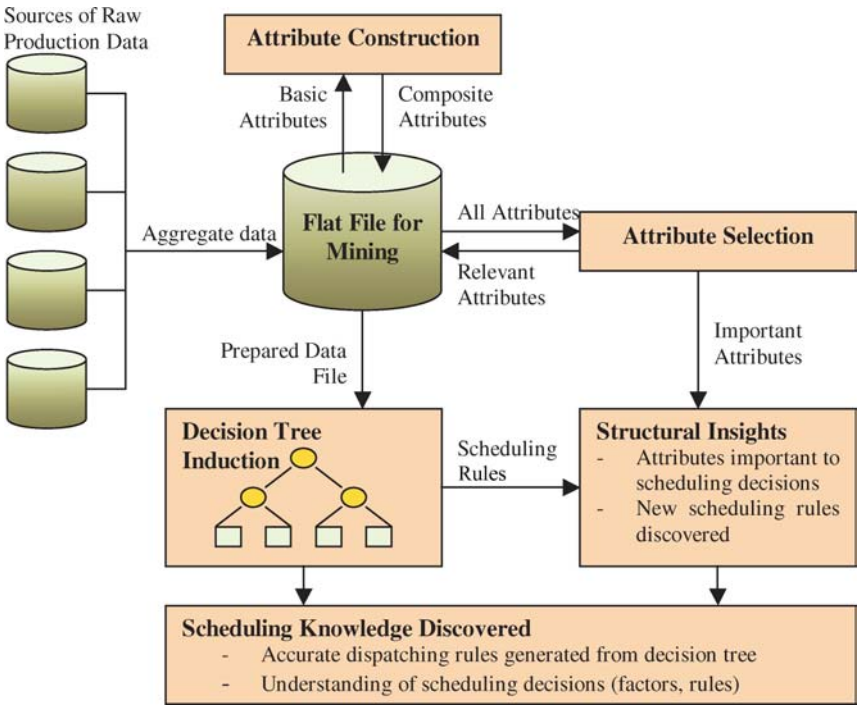

Figure 1. High-level framework for discovering scheduling knowledge.

the target concept to be learned to be a dispatching rule. In particular, given two jobs we want to learn to determine which job should be dispatched first. This knowledge would allow us to dispatch the next job at any given time and create dispatching lists for any set of jobs. Note that we do not need to predict the starting time of each job but simply when we compare two jobs, which job should be processed first. Given this target concept the framework has two main phases: (a) data preparation including aggregation, attribute construction, and attribute selection, and (b) model induction and interpretation (see Figure 1).

The initial preparation of data is highly significant since to mine any useful knowledge from the raw production data it must typically be transformed considerably. For data mining, the database should be represented as a flat data file where the columns represent the attributes of the data and each row of the file represents one piece of information independent of the other rows: an example from which we can learn. We refer to each such example or a row in the file as an instance. The raw production data that is available is likely to be in various formats. This data, could for example, include job dispatch list, work schedules, bill of materials, and so forth. All of these data sources must be combined into a single flat file.

Beyond the initial combination of various data sources into a single database, the engineering of this database plays a critical role in the usefulness of the knowledge discovered. It is indeed unlikely that the attributes that are recorded as part of the raw production data are the attributes that are the most useful for data mining. Thus, new attributes creation must be considered using both intuitive processes and automated learning. On the other hand, using attribute selection to eliminate certain

redundant and irrelevant attributes is also critical to the effectiveness of the subsequent model induction. Attribute selection also has an inherent value in that insightful structural knowledge may be obtained by selecting which attributes are important, that is, which factors most influence the scheduling decision.

After the data file has been constructed a learning algorithm is applied to induce a predictive model for the target concept. Such learning can be achieved in many ways, including using decision tree induction, statistical learning, neural networks, and support vector machines. For our framework, we focus on transparent methods, as we believe that black-box models that are difficult or impossible for the user to comprehend are not likely to be used effectively in an actual production environment. In particular, we focus on using decision trees and decision rules derived from such trees.

The decision tree induced using the learning algorithm can be applied directly as a predictive model to predict the target concept, which in our framework is a dispatching rule. Thus, the tree will, given any two jobs, predict which job should be dispatched first and can be thought of as a new, previously unknown, dispatching rule. In addition to the prediction, decision trees and decision rules often also reveal insightful structural knowledge that can be used to further enhance the scheduling decision.

3.2. Numerical example

To illustrate the framework described in the last subsection we now consider a simple numerical example. We assume that the dispatching list shown in Table 1 has been used for processing five jobs on a certain single machine system.

Now assume that we do not know how jobs were scheduled but still wish to construct an automatic system to dispatch new jobs according to the same logic. We thus need to induce from the data above some rule that can be used to dispatch jobs and therefore our target concept to be learned will be a simple dispatching rule. (Note that the actual rule used to create the data in the example is to dispatch the jobs with the longest processing time first for all jobs that have been released, but this is considered unknown.)

The first step of the framework is data file construction. Given the target concept, any such data file should have a class attribute called *Job1First*, which can take two values: yes or no. Thus, when all the attributes are specified for two jobs, job 1 and job 2, the objective is to predict if job 1 comes first (class value 'yes') or not (class value 'no'). In addition to making the prediction, the model should ideally reveal some structural knowledge about the production system that reveals why one job is dispatched ahead of another. In this example, that knowledge should be that a job is

Table 1. Dispatching list

Job ID	Release time	Start time	Processing time	Completion time
J5	0	0	17	17
J1	10	17	15	32
J3	18	32	20	52
J4	0	52	7	59
J2	30	59	5	64

Table 2. Data set constructed for data mining

Job1	*ProcessingTime1*	*Release1*	*Job2*	*ProcessingTime2*	*Release2*	*Job1ScheduledFirst*
J1	15	10	J2	5	30	Yes
J1	15	10	J3	20	18	Yes
J1	15	10	J4	7	0	Yes
J1	15	10	J5	17	0	No
J2	5	30	J3	20	18	No
J2	5	30	J4	7	0	No
J2	5	30	J5	17	0	No
J3	20	18	J4	7	0	Yes
J3	20	18	J5	17	0	No
J4	7	0	J5	17	0	No

dispatched if it is the longest job of those that have been released. With this in mind, we construct the initial data set as shown in Table 2.

As is usual for knowledge discovery data files, due to the requirement that each line, or instance, be an example of the concept to be learned that is independent of all other instances, this file contains some inefficiency. However, it is clear that once it has been designed, it is not difficult to construct this data automatically from the dispatching list in Table 1.

As discussed above, attribute construction and selection is an essential element of being able to successfully mine scheduling patterns from the data. However, for illustration purposes we first apply the well-known C4.5 decision tree algorithm of Quinlan (1993) to the data in Table 2 directly. For brevity, we omit the actual tree, but it corresponds to the following classification rules, which in this case is a set of dispatching rules:

If *ProcessingTime1* ≤ 7 **then** dispatch Job 2 first.
If *ProcessingTime1* > 7 **and** *ProcessingTime2* ≤ 7 **then** dispatch Job 1 first.
If *ProcessingTime1* > 7 **and** *ProcessingTime2* > 7 **then** dispatch Job 1 first

These scheduling rules dispatch all but one of the training instances correctly, that is, it would replicate the dispatching list in Table 1 almost exactly (the order of Job 1 and Job 3 is reversed) for a dispatch order of J5-J3-J1-J4-J2. Since there are 10 instances, this corresponds the model having 90% accuracy. However, it is also quite limited in the amount of structural knowledge or insights that can be obtained.

In order to obtain more meaningful decision tree, and hence dispatching rules, a better data file must be constructed before the decision tree induction. To illustrate the potential of the approach, we deliberately add attributes that are believed to be helpful. Specifically, we add two new attributes, *ProcessingTimeDifference* and *ReleaseDifference*, which are defined as follows: *ProcessingTimeDifference* = *ProcessingTime1* – *ProcessingTime2* and *ReleaseDifference* = *Release1* – *Release2*. We also add two attributes that indicate which job is longer and which is released first: *Job1Longer* and *Job1ReleasedFirst*, that both take values yes or no. We then apply the same decision tree algorithm again, resulting in the decision tree shown in Figure 2. Note that a 'yes' in a leaf node implies that Job 1 should be dispatched first, and vice versa. The decision tree, or alternatively the corresponding decision rules, can be used to directly dispatch any job. Given any two jobs a

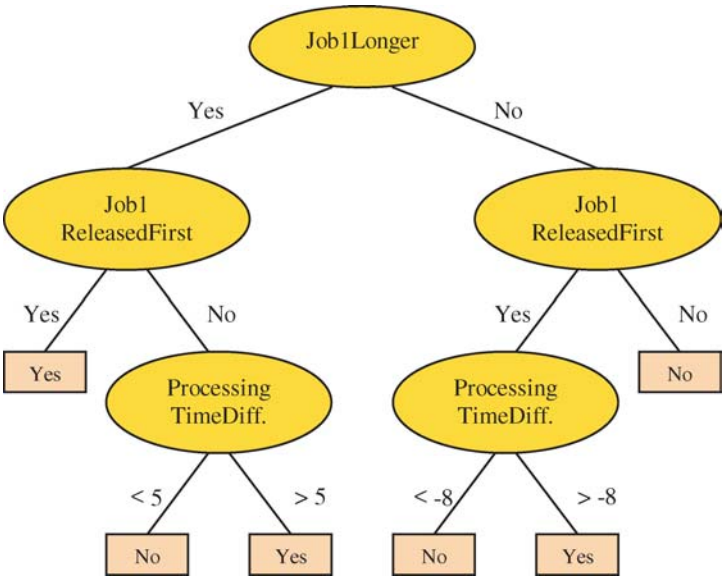

Figure 2. Decision tree for dispatching jobs.

selection can be made about which job should be dispatched first and hence, given any set of jobs that has been released into the system, a selection can be made.

In addition to being used for prediction, in our framework we also look at what previously unknown structural information can be discovered (see Figure 1). Looking at the decision tree we note that there are two easy classification cases. If Job 1 is both longer and is released first, then it is dispatched first (leaf node furthest to the left). Vice versa if neither holds then Job 2 is dispatched first (leaf node furthest to the right). The ambiguous cases occur in between and in those cases the decisions as to which job is dispatched first are determined by the difference in processing time between the two jobs. When compared to the rules obtained earlier without using the new attributes, these rules clearly reveal more structural knowledge and get to the point by focusing on the processing time difference. In particular, note that the first rule says that if the processing time of Job 1 is much smaller than that of Job 2, then Job 2 should be dispatched first even if Job 1 is released first. What counts as being 'much smaller' is determined by the data, and in particular how much delay a late release date can possibly cause for this particular system. Thus, we have discovered the following nuggets of information:

– If a job is both longer and released ahead of another job it should be dispatched first.
– A job that is released first into the system should wait for an anticipated longer job to be available if that job is much longer, specifically if its processing time is 5–8 units longer.

Such observations can then be assessed to see if scheduling practices should be changed. Should longer jobs be favored? Is it justified to idle machines to wait for those longer jobs? How should a 'much longer' job be defined?

We contend that in an actual production situation, such observations could result in significant improvements. Indeed, more may be discovered than could be found out even if the dispatcher

could accurately describe the scheduling process. Here the dispatcher would simply state that jobs are dispatched according to LPT for all jobs that have been released. The induction algorithm is learning from examples that use this rule and the processing time and release time data. Thus, the induced model can take into account the possible range of processing times and the largest possible delay that can be caused by a job not being released, and thus discovering new structural patterns that may not be explicitly known by the dispatcher.

Despite its simplicity, this example lends itself to numerous observations. It is clear that data mining algorithms, and in particular decision trees, can be used to extract knowledge concerning scheduling concepts from production data such as dispatch lists. Such knowledge can be both predictive, as in determining which of two jobs should be dispatched first, and descriptive, for example by revealing why a job should be dispatched first. However, discovering descriptive knowledge that may be used further to improve schedule performance is not a trivial matter, and as illustrated by the example depends on the representation of the data. Thus, we infer that to mine for information in scheduling data, the data should be represented in ways that are significantly different from traditional scheduling data outputs. Furthermore, a careful construction of new attributes to represent the scheduling data can greatly enhance the value of the models obtained, in particular with respect to the structural knowledge that is gained.

4. DECISION TREES AS DISPATCHING RULES

The example in Section 3 motivates that decision tree induction can be used to learn novel dispatching rules. In this section, we evaluate the quality of such dispatching rules more extensively through a series of simulation experiments. This evaluation is based on data created using four simple and well-known dispatching rules for a single machine problem, namely, weighted earliest due date (EDD), weighted shortest processing time first (WSPT), minimum slack (MS), and earliest release date (ERD). Corresponding to each of these rules, there is a priority index $I_j^{\mathrm{RULE}}(p_j, r_j, d_j, w_j)$ that is a function of the basic characteristics of the job, that is, the processing time (p_j), release time (r_j), due date (d_j), and weight (w_j).

As in Section 3, the basic data p_j, r_j, d_j, and w_j is first generated for a set of jobs $j = 1, 2, \ldots, m$. In these experiments the total number of jobs used ranges from $m = 120$ to $m = 200$. Given the basic data, the jobs are ordered according to the appropriate priority index $I_j^{\mathrm{RULE}}(p_j, r_j, d_j, w_j)$, with the job with the lowest index being first, and so forth. This results in a dispatching list similar to Table 1. This dispatching list is then transformed into a flat file similar to the one in Table 2, except that due dates and weights of each job are also included. For this and all subsequent experiments, there are eight basic attributes plus the class attribute, and between 7,140 ($m = 120$) and 19,900 ($m = 200$) instances. As before the concept to be learned is which of the two jobs is dispatched first. All experiments are replicated four times, which was found to be sufficient due to relatively low variability in the knowledge discovery process.

We start by addressing the question of whether the decision tree algorithm can accurately replicate the dispatching list generated by the rule. Table 3 shows the results when the C4.5 decision tree algorithm is applied to each of the four flat files.

It is clear that in each case the decision tree algorithm accurately discovers how the jobs were dispatched by the particular rule, although the EDD and WSPT rules appear to be slightly more difficult to discover. This supports the claim made based on the example in Section 3, but as in that example, further improvements can be made by engineering the data.

Table 3. Accuracy of decision trees in replicating dispatching lists

Rule	Average accuracy (%)	Standard deviation	Minimum accuracy (%)	Maximum accuracy (%)
EDD	96.18	0.28	95.9	96.5
WSPT	96.30	0.36	95.8	96.3
MS	98.20	1.05	97.2	99.3
ERD	99.20	0.24	98.9	99.1

Data engineering plays a critical role in constructing more accurate models, building more interpretable models, and in providing insights into which factors are most important in making the scheduling decisions. Specifically, the attributes that are recorded as part of the raw production data may not be the attributes that are the most useful for the data mining itself. Thus, new attributes creation must be considered (Chen and Yih, 1996). In some cases, this could be done manually using intuitive processes such as those illustrated in the example in Section 3.2 above, where the best possible attributes were constructed manually. However, this is generally not possible in practice, and now we show how valuable new attributes can be discovered automatically.

Attribute selection using frequent itemset generation was introduced in the seminal paper by Agrawal, Imielinski, and Swami (1993) and this approach has been found to be useful in many areas of knowledge discovery. In this context, an item is an attribute value pair, that is an attribute along with one of its possible values, and an itemset is simply a set of items. An itemset is called frequent if it occurs some minimum number of times in the database, that is, meets the minimum support. It is possible to use such frequent itemset generation to construct new attributes that are helpful for classification learning (Lesh, Zaki, and Ogihara, 1999; Deshpande and Karypis, 2002). With this approach, new composite attributes are added based on attributes that occur frequently together, which allows the subsequent learning algorithm to use conjunction of attribute values. For example, if itemsets involving processing time of job 1, p_1, and processing time of job 2, p_2, occur frequently together then we attempt to construct new composite attribute using this pair with four basic arithmetic operations: $p_1 - p_2$, $p_1 + p_2$, p_1/p_2, and $p_1 \cdot p_2$.

In addition to creating new attributes, using attribute selection to eliminate certain attributes is critical to the effectiveness of the subsequent model induction. Attribute selection in general is an important part of the knowledge discovery for numerous reasons. It can be used to eliminate redundant and irrelevant attributes from a data set, resulting in a dimensionality reduction that reduces the learning time needed for induction algorithms that are applied to the data set, and in many cases also results in better (i.e., more accurate) predictive models. Careful attribute selection can improve the scalability of a data mining system as the induction is usually much faster with fewer attributes, and finally, attribute selection also has an inherent value in that insightful structural knowledge may be obtained by selecting which attributes are important.

To illustrate how the engineering of the data improves the performance of the decision tree models, Table 4 shows the accuracy of the decision trees after attribute construction and selection for the same data sets as those reported in Table 3. It also indicates the improvement over the accuracy using the original attributes only. The actual number of derived attributes varies, but the average number is 36 attributes for the EDD data sets, and 37 attributes for the others. After attribute selection the average number of remaining attributes is 5 for the EDD and WSPT datasets, 10 for the MS datasets, and 1 for the ERD datasets.

Table 4. Accuracy of decision trees with data engineering

Rule	Average accuracy (%)	Standard deviation	Change in accuracy
EDD	99.10	0.08	2.93
WSPT	98.90	0.26	2.73
MS	97.47	2.27	−0.73
ERD	99.90	0.00	0.70

Table 5. Reduction in size of decision trees

	Number of nodes in decision tree		
Rule	Original data	Engineered data	Difference (%)
EDD	175.5	62.8	−64
WSPT	174.8	58.3	−67
MS	87.0	63.7	−27
ERD	59.0	2.0	−97

We note that in all but one of the cases, there is an average improvement in accuracy, and the improvement is particularly large for the EDD and WSPT data sets, where the accuracy was smaller before. Thus, we conclude that the data engineering is useful to improve accuracy of the models.

However, as noted before the primary benefit of attribute creation and selection may not be improved accuracy, but rather simpler models and structural insights. Thus, Table 5 compares the size of the decision tree, as measured by the number of nodes in the tree, both before and after the data engineering.

It is clear that the decision trees generated after the data engineering are much smaller, and hence typically simpler to interpret, than the trees generated using the original data. As for the accuracy, the attribute creation and selection is the least useful for the data generated using the MS rule, where there is only 27% reduction in tree size. On the other hand, for the data generated by the ERD rule there is a very dramatic reduction down to only two attributes, which is a reduction of 97% in size.

The improvements in accuracy are achieved by first creating derived attributes and then selecting a subset of attributes that is the most important. To obtain some insights into what is important it is interesting to consider what composite attributes are created for each data set. Table 6 shows the attribute pairs used as well as the number of derived attributes used by the model (after attribute selection). Some attribute pairs were used in every replication, whereas others were only used by some. For convenience, the table also shows the priority index for each rule. These indices are of course unknown by the learning algorithm, but ideally one would expect the attribute construction and selection to discover the components of the relevant index.

We note that for the EDD rule, a composite of the weight and due date of each attribute was discovered by the attribute construction in every replication. This is quite intuitive since the priority index is $I_j^{\mathrm{EDD}} = w_j d_j$, and two composite attributes $w_1 d_1$ and $w_2 d_2$ are thus sufficient

Table 6. Attributes discovered by the attribute creation and selection process

Rule	Priority index	Attribute pairs used	Number of derived attributes used
EDD	$I_j^{\mathrm{EDD}} = w_j d_j$	$(w_1, d_1)^*$, (w_2, d_2)	2
WSPT	$I_j^{\mathrm{WSPT}} = \frac{w_j}{p_j}$	(w_1, p_2), (w_2, p_1), (w_1, p_1), (w_2, p_2)	2
MS	$I_j^{\mathrm{MS}} = \max\left\{\frac{d_j - p_j - t}{w_j}, 0\right\}$	(d_1, r_1), (d_2, r_2), (d_1, d_2), (w_2, d_1), (w_2, p_2)	2–3
ERD	$I_j^{\mathrm{ERD}} = r_j$	$(r_1, r_2)^{**}$	1

* Used by every data set.

to determine which job is dispatched first, Job 1 or Job 2. We emphasize again that the EDD rule is assumed unknown a priori, but the data mining has discovered that $w_1 d_1$ and $w_2 d_2$ are most important factors in the scheduling decision, which is the essential structural insight for this particular system.

Similarly, for ERD, Job 1 goes ahead of Job 2 if and only if $r_1 < r_2$, so a composite attribute of $r_1 - r_2$ is sufficient. The attribute construction discovers this, and creates this composite attribute for every replication. For the WSPT and MS data sets, the process does not always discover the same attribute pairs. For the WSPT, two composites of processing time and weight were used every time, but not always the same combination. Finally, for the data set generated by the more complex MS rule, two or three composite attributes were used each replication, involving composites of due dates, release times, weights, and processing time. Overall, these results show that useful composite attributes are constructed and these attributes provide significant insights into what factors are important in each scheduling situation.

So far, we have seen that decision tree induction can accurately discover the underlying structure of dispatching rules but we have not yet considered the actual scheduling performance. Table 7 does this for all of the dispatching rules and three problem sizes (120, 160, and 200 jobs). The table shows the average difference in performance as measured by the makespan ($\Delta C_{\max}$), the standard deviation (S.D.) of the difference, and the percentage difference.

Since the purpose of the decision trees is to learn the scheduling concepts contained in the original dispatching list, not to improve upon it, we do not expect the scheduling performance when the trees are used as dispatching rules to be better than the original. However, if the trees do discover the relevant concepts then the performance should be similar to the original rules. From Table 7 we note that in 9 out of 12 cases the difference is less than 2%. For all of dispatching list

Table 7. Accuracy of decision trees versus original dispatching rules

	Problem size								
	120 Jobs			160 Jobs			200 Jobs		
Rule	$\Delta C_{\max}$	S.D.	% $\Delta C_{\max}$ (%)	$\Delta C_{\max}$	S.D.	% $\Delta C_{\max}$ (%)	$\Delta C_{\max}$	S.D.	% $\Delta C_{\max}$ (%)
EDD	9.0	6.3	1.7	7.4	1.7	1.2	0.4	0.7	0.1
WSPT	8.9	6.1	1.2	43.7	13.5	8.3	29.9	17.4	6.3
MS	4.6	6.8	0.9	17.0	5.4	3.6	11.9	5.4	1.7
ERD	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

created using the ERD rule, the tree replicates the list exactly and there is no difference. The largest difference is for WSPT, where the tree is as much as 8.3% worse on the average than the original dispatching list. However, overall the dispatching rules based on the decision trees have similar performance as the original rules that we are attempting to discover.

5. DISCUSSION

We have introduced a new framework for using data mining to discover dispatching rules from production data. We show that by using decision tree models to learn from properly prepared data set, not only do we obtain a predictive model that can be used as a dispatching rule, but previously unknown structural knowledge can be obtained that provides new insights and may be used to improve scheduling performance. Furthermore, we develop methods for using frequent item set generation to construct composite attributes that improve the performance of the predictive models, and in combination with attribute selection method reveal what factors are the most influential in making the scheduling decisions.

The data-driven approach proposed in this paper implicitly assumes that mimicking a human scheduler's decision is worthwhile and it does not take into account any traditional performance measures. Thus, the methodology is directly applicable when the human scheduler's decisions are believed to be of high quality, and we want to make them explicit and perhaps incorporate them into an automated system. However, we envision two types of extensions that overcome the obvious limitation of not considering any performance measures. First, instead of learning from all of the data, it might be better to learn from a subset of data that corresponds to best practices. Preliminary results indicate that this is a viable approach, but the challenge is to determine how best practices can be identified in an automated manner. Second, the scheduling knowledge discovered using the data mining approach can be taken as the starting point to improve scheduling decisions. This could be done in an ad-hoc manner by a manual examination of the decision tree, but we feel that an important future research topic is to investigate how optimization techniques can be used to search for an improved decision tree, given the components identified during the data mining.

In order to clearly demonstrate how scheduling knowledge can be discovered using data mining, the experiments reported in this paper focused on single machine problems where simple underlying structures could be used. However, it is important to realize that the data mining techniques applied in this approach are commonly used for problems with thousands of attributes and millions of instances. In this context, possible attributes include not only job characteristics, such as processing time and due date, but the state of the system, and the values of the multiple objectives a scheduler might consider. Thus, this approach is particularly applicable for large, complex production environments, even when this complexity makes it difficult to model the system explicitly.

Finally, mimicking common dispatching rules, we have limited ourselves to looking at a pairwise comparison between two jobs as the target concept to be learned. With this approach, we can expect to learn dispatching rules of similar structure as traditional rules, such as those used as a benchmark in this paper. An interesting future research topic is to extend this inductive learning approach to consider comparisons between multiple jobs. This would allow us to learn dispatching rules of higher complexity that could potentially be more effective for scheduling than traditional rules.

REFERENCES

Agrawal, R., T. Imielinski, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM-SIGMOD International Conference on Management of Data*, 1993.

Aytug, H., S. Bhattacharyya, G. J. Koehler, and J. L. Snowdon, "A review of machine learning in scheduling," *IEEE Transactions on Engineering Management*, **41**(2), 165–171 (1994).

Bowden, R. and S. F. Bullington, "Development of manufacturing control strategies using unsupervised machine learning," *IIE Transactions*, **28**, 319–331 (1996).

Chen, C. C. and Y. Yih, "Identifying attributes for knowledge-based development in dynamic scheduling environments," *International Journal of Production Research*, **34**(6), 1739–1755 (1996).

Deshpande, M. and G. Karypis, "Using conjunction of attribute values for classification," *CIKM'02*, Nov. 4–9, McLean, VA, 356–364 (2002).

Geiger, C. D., R. Uzsoy, and H. Aytug, "Autonomous learning of effective dispatch policies for flowshop scheduling problems," in *Proceedings of the Industrial Engineering Research Conference* (IERC'03) (2003).

Hestermann, C. and M. Wolber, "A comparison between operations research-models and real world scheduling problems," The European Conference on Intelligent Management Systems in Operations; University of Salford, U.K., 25–26 March 1997.

Jain, A. S. and S. Meeran, "Job-shop scheduling using neural networks," *International Journal of Production Research*, **36**(5), 1249–1272 (1998).

Kanet, J. J. and H. H. Adelsberger, "Expert systems in production scheduling," *European Journal of Operational Research*, **29**, 51–59 (1987).

Kim, C.-O., H.-S. Min, and Y. Yih, "Integration of inductive learning and neural networks for multi-objective FMS scheduling," *International Journal of Production Research*, **36**(9), 2497–2509 (1998).

Kusiak, A. and M. Chen, "Expert systems for planning and scheduling manufacturing systems," *European Journal of Operational Research*, **34**, 113–130 (1988).

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

Lesh, N., M. J. Zaki, and M. Ogihara, "Mining features for sequence classification," in 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (1999).

Li, D.-C. and I.-S. She, "Using unsupervised learning technologies to induce scheduling knowledge for FMSs," *International Journal of Production Research*, **32**(9), 2187–2199 (1994).

Min, H.-S., Y. Yih, and C.-O. Kim, "A competitive neural network approach to multi-objective FMS scheduling," *International Journal of Production Research*, **36**(7), 1749–1765 (1998).

Nakasuka, S. and T. Yoshida, "Dynamic scheduling system utilizing machine learning as a knowledge acquisition tool," *International Journal of Production Research*, **30**(2), 411–431 (1992).

Noronha, S. J. and V. V. S. Sarma, "Knowledge-based approaches for scheduling problems: A survey," *IEEE Transactions on Knowledge and Data Engineering*, **3**(2), 160–171 (1991).

Pinedo, M., *Scheduling: Theory, Algorithms and Systems*, Prentice Hall, 1995.

Piramuthu, S., N. Raman, and M. J. Shaw, "Learning-based scheduling in a flexible manufacturing flow line," *IEEE Transactions on Engineering Management*, **41**(2), 172–182 (1994).

Piramuthu, S., N. Raman, M. J. Shaw, and S. Park, "Integration of simulation modeling and inductive learning in an adaptive decision support system," *Decision Support Systems*, **9**, 127–142 (1993).

Priore, P., D. De La Fuente, A. Gomez, and J. Puente, "A review of machine learning in dynamic scheduling of flexible manufacturing systems," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **15**, 251–264 (2001).

Quinlan, J. R., *C4.5: Programs for Machine Learning*. Morgan Kaufmann, San Mateo, CA, 1993.

Shaw, M. J., S. Park, and N. Raman, "Intelligent scheduling with machine learning capabilities: The induction of scheduling knowledge," *IIE Transactions*, **24**(2), 156–168 (1992).

Shaw, M. J. and A. B. Whinston, "An artificial intelligence approach to the scheduling of flexible manufacturing systems," *IIE Transactions*, **21**(2), 170–183 (1989).

Wiers, V. C. S., "A Review of the Applicability of OR and AI Scheduling Techniques in Practice," *OMEGA - The International Journal of Management Science*, **25**(2), 145–153 (1997).