Data Mining for Production Scheduling

Sigurdur Olafsson
Department of Industrial and Manufacturing Systems Engineering

Iowa State University,
Ames, IA 50011

Abstract

Data mining is a fast growing field and many industrial engineering applications generate large amounts of data to which data mining techniques can be applied. In this paper we develop a data mining framework for production scheduling. This involves preprocessing of historic schedules 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.

Keywords:

Data mining, scheduling, inductive learning, decision trees, dispatching rules

1. Introduction

Production scheduling provides an important function in any manufacturing facility, and systems for automating this function have received considerable attention. In the past, the schedule generation of such scheduling systems has primarily been based on either operations research models, expert systems, or a hybrid method. These approaches have achieved some successes, but each is limited by the requirement that the designer must either model the system or derive the basic scheduling rules beforehand based on expert knowledge of the system. This is not always possible. Production scheduling in many facilities is done on an ad-hoc basis that does not rely solely on well-defined rules or principles, but rather on the intuition and experience of the scheduler. When such knowledge is not explicitly captured, any model-based approach may fail to capture important considerations.

Data mining and knowledge discovery is an emerging area of research and applications that uses machine learning and statistical methods to learn previously unknown and useful knowledge from examples in large databases. Data mining has already made a significant impact on numerous industries but its function in manufacturing, and in production scheduling in particular, have only received moderate attention to date. The strengths of data mining lie in areas 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. Both situations commonly exist in production environments. These environments 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.

We suggest that data mining can be used to capture both explicit and implicit knowledge that is used to create production schedules. This enables a) automating decision support for scheduling procedures that are currently dependent on the intuition and experience of the schedulers, b) discovering hidden patterns in the schedule generation that may add insights not realized by the schedulers themselves, and c) looking for ways in which the current scheduling can be improved. Accordingly, in this paper we describe a framework that can be used to discover previously unknown dispatching rules directly from production data.

2. Relation to Other Research

Motivated partially by the limitations of traditional operations research model for practical scheduling [5,15,20], artificial intelligence for production scheduling has received considerable attention over the past two decades [1,7,9,13,16,19]. This includes for example the use of neural networks [6,12], induction [18], hybrid approaches [8,10], and unsupervised learning [2,11].

Our approach relates to prior research on machine learning for production scheduling in that much of this research has focused on dynamic scheduling and intelligent selection of dispatching rules [4,14]. For example, it has been suggested to use simulated data to generate training data that is then generalized using the learning algorithm to

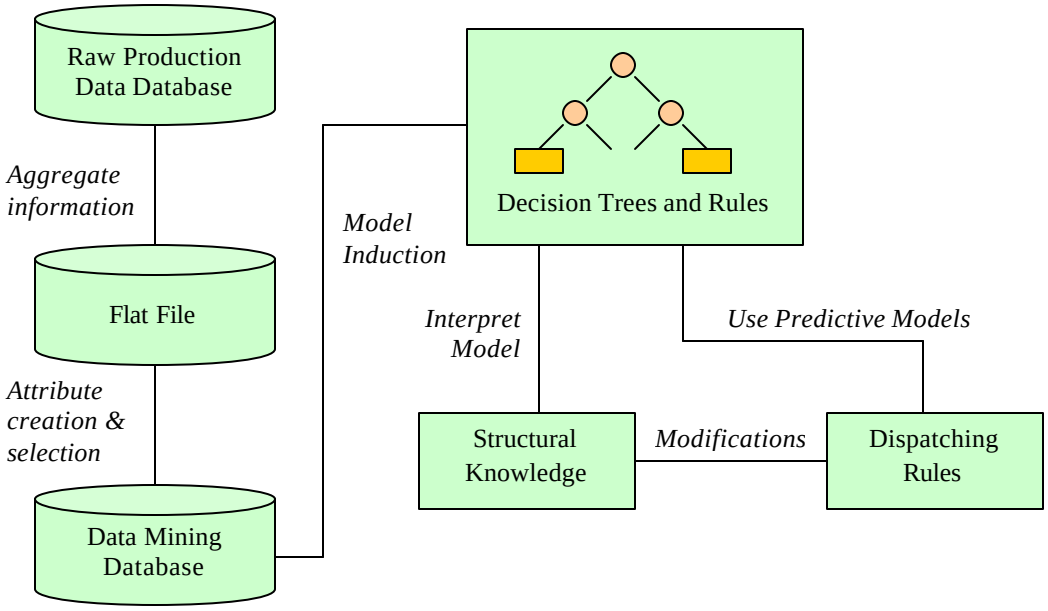

Figure 1: High-level overview of the framework for learning dispatching rules

learn which dispatching rule to use at which time. This previous work differs from the framework presented here in that we propose to use data mining to learn directly from production data. For example, instead of learning to select between a priori specified dispatching rules, our approach learns the dispatching rules themselves. Thus, in our framework, new previously unknown dispatching rules may be discovered. Furthermore, in this framework we focus on extracting structural knowledge from the rules discovered. We contend that such knowledge can lead to new insights into both the production system itself and the current scheduling practices that can then be exploited to improve scheduling performance.

3. Learning Dispatching Rules

As mentioned above, several researchers have considered using machine learning for dynamically selecting the best dispatching rule, often using simulated data. What has, to our knowledge, not been addressed before is how data mining can be used to learn new dispatching rules by applying it directly to existing scheduling data. The benefits of this approach include:

- The implicit knowledge of expert schedulers is discovered and can be automated in a system that can create future schedules without the direct involvement of such experts.
- Existing scheduling practices are generalized into explicit scheduling rules. These rules can then be applied to both situations that have occurred before and to new situations.
- 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 describe the general framework for data mining of dispatching rules from existing production data. From the data mining perspective we specify the target concept to be learned to be a dispatching rule. What this implies is that given two jobs we want to learn which job should be dispatched first, which 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 as in the dispatching list above, 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 and b) model induction and interpretation (see Figure 1).

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 an example or a row in the file as an instance. The production data that is available is likely in different formats. This data could for example include job dispatch list, work schedules, bill of materials, and so forth. 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 [3]. This can be done using both intuitive processes, and semi-automated learning such as using association rule discovery to uncover important temporal, spatial, and statistical characteristic that should be represented as separate attributes. Association rule discovery has been found to be effective in other contexts for such attribute creation.

On the other hand, using attribute selection to eliminate certain attributes is also critical to the effectiveness of the subsequent model induction. Attribute selection 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 (that is, 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.

After the data file has been constructed a learning algorithm is applied to induce a predictive model for the target concept. There are many ways in which such learning can be achieved, 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.

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 Numeric Example

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

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

Now assume that we do not know how jobs are scheduled but still wish to construct an automatic system that dispatches the jobs. 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 is 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. Note that all of these data files have a common 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 make the prediction, the model should ideally reveal some structural knowledge about the system that reveals why one job is dispatched ahead of another. In this case, that knowledge should be that jobs are dispatched if it is the longest job of those that have been released already. We construct the initial data set as follows:

Table 2: Data set constructed for data mining

Job1	ProcessingTime1	Release1		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

As is usually the case for the construction of data files for knowledge discovery, this file contains some inefficiency. In particular, each line, or *instance*, is an example of the concept to be learned that is independent of all other instances. 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 [17] to the data in Table 2 directly. The following classification rules, which in this case is a set of dispatching rules, were obtained by reading them from the C4.5 tree:

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. However, it is 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 is started. Hence, 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 selection can be made which job should be dispatched first and hence, given any set of jobs that has been released into the system

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 decision as to which job is dispatched first are determined by the difference in processing time between the two jobs. 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:

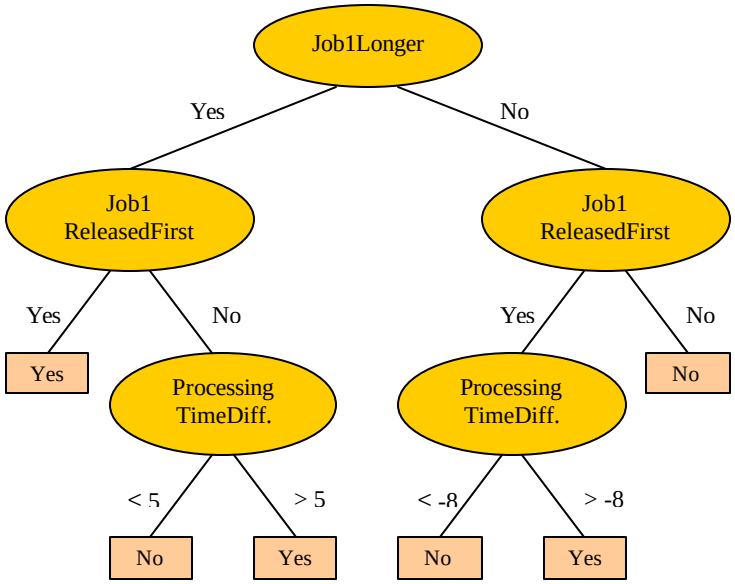

Figure 2: Decision tree for dispatching jobs

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 to 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 content 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 formal 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.

3.3 Analysis

The example in Section 3.2 above lends itself to numerous observations. It is clear from the example 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 further used 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.

These observations serve as further motivation for the framework presented in Section 3.1 above (see Figure 1). Recall that this framework has two phases: a) data preparation and b) model induction and interpretation. As illustrated by the example of Section 3.2, although the second phase is where the "true" data mining occurs, that is the induction phase, the first phase may actually be much more critical to the success of the knowledge discovery.

4. Conclusion

We have introduced a new framework for learning dispatching rules directly 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.

Future work includes further numerical experiments, investigation of how new attributes should be constructed to most effectively mine new dispatching rules, and implementation of the results in an industry setting.

References

1. Aytug, H., Bhattacharyya, S., Koehler, G.J., Snowdon, J.L. (1994). A review of machine learning in scheduling. *IEEE Transactions on Engineering Management* 41(2), 165-171.
2. Bowden, R., Bullington, S.F. (1996). Development of manufacturing control strategies using unsupervised machine learning. *IIE Transactions* 28, 319-331.
3. Chen, C.C., Yih, Y. (1996). Identifying attributes for knowledge-based development in dynamic scheduling environments. *International Journal of Production Research* 34(6), 1739-1755.
4. Chiu, C., Yih, Y. (1995). A learning-based methodology for dynamic scheduling in distributed manufacturing systems. *International Journal of Production Research* 33(11), 3217-3232.
5. Hestermann C. and M. Wolber. (1997). 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.
6. Jain, A.S., Meeran, S. (1998). Job-shop scheduling using neural networks. *International Journal of Production Research* 36(5), 1249-1272.
7. Kanet, J.J., Adelsberger, H.H. (1987). Expert systems in production scheduling. *European Journal of Operational Research* 29, 51-59.
8. Kim, C.-O., Min, H.-S., Yih, Y. (1998). Integration of inductive learning and neural networks for multi-objective FMS scheduling. *International Journal of Production Research* 36(9), 2497-2509.
9. Kusiak, A., Chen, M. (1988). Expert systems for planning and scheduling manufacturing systems. *European Journal of Operational Research* 34, 113-130.
10. Lee, C.-Y., Piramuthu, S., Tsai, Y.-K. (1997). Job shop scheduling with a genetic algorithm and machine learning. *International Journal of Production Research* 35(4), 1171-1191.
11. Li, D.-C., She, I.-S. (1994). Using unsupervised learning technologies to induce scheduling knowledge for FMSs. *International Journal of Production Research* 32(9), 2187-2199.
12. Min, H.-S., Yih, Y., Kim, C.-O. (1998). A competitive neural network approach to multi-objective FMS scheduling. *International Journal of Production Research* 36(7), 1749-1765.
13. Noronha, S.J. and V.V.S. Sarma. (1991). Knowledge-based approaches for scheduling problems: a survey. IEEE Transactions on Knowledge and Data Engineering 3(2), 160-171
14. Pierreval, H., Mebarki, N. (1997). Dynamic selection of dispatching rules for manufacturing system scheduling. *International Journal of Production Research* 35(6), 1575-1591.
15. Pinedo, M. (1995). *Scheduling: Theory, Algorithms and Systems*. Prentice Hall.
16. Priore, P., D. De La Fuente, A. Gomez, and J. Puente. (2001). A review of machine learning in dynamic scheduling of flexible manufacturing systems. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 15, 251-264.
17. Quinlan, J.R. 1986. Induction of decision trees. *Machine Learning* 1(1), 81-106.
18. Shaw, M.J., Park, S., Raman, N. (1992). Intelligent scheduling with machine learning capabilities: the induction of scheduling knowledge. *IIE Transactions* 24(2), 156-168.
19. Shaw, M.J., Whinston, A.B. (1989). An artificial intelligence approach to the scheduling of flexible manufacturing systems. *IIE Transactions* 21(2), 170-183.
20. Wiers, V.C.S. (1997). A Review of the Applicability of OR and AI Scheduling Techniques in Practice. *OMEGA - The International Journal of Management Science* 25(2), 145-153.