International Journal of Computer Integrated Manufacturing

Publication details, including instructions for authors and subscription information:
http://www.tandfonline.com/loi/tcim20

An enhanced knowledge representation for decision-tree based learning adaptive scheduling

Yeou-Ren Shiue & Chao-Ton Su

Published online: 08 Nov 2010.

To cite this article: Yeou-Ren Shiue & Chao-Ton Su (2003) An enhanced knowledge representation for decision-tree based learning adaptive scheduling, International Journal of Computer Integrated Manufacturing, 16:1, 48-60, DOI: 10.1080/713804978

To link to this article: http://dx.doi.org/10.1080/713804978

INT. J. COMPUTER INTEGRATED MANUFACTURING, 2003, VOL. 16, NO. 1, 48–60

An enhanced knowledge representation for decision-tree based learning adaptive scheduling

YEOU-REN SHIUE and CHAO-TON SU

Abstract. The classical decision tree (DT) learning approach to constructing DT knowledge bases is usually not considered if there exist some irrelevant and redundant attributes in the problem domain. Since the essential attributes are uncertain in manufacturing systems, how to select important manufacturing attributes to improve the generalization ability of knowledge bases and avoid overfitting training data in DT-based learning is a crucial research issue for the adaptive scheduling problem domain. In this study, we will first develop an attribute selection algorithm based on the weights of artificial neural networks (ANNs) to identify the importance of system attributes. Next, we will use the C4.5 DT learning algorithm to learn the whole set of training examples with important attributes in order to enhance knowledge representation. This hybrid ANN/DT approach is called an attribute selection DT (ASDT) based learning adaptive scheduling system. The results from the case study show that the use of an attribute selection algorithm to build scheduling knowledge bases delivers better generalization ability than in the absence of the attribute selection procedure in terms of the size of DTs under various performance criteria. Consistent conclusions are drawn from the resulting prediction accuracy of unseen data. The resulting prediction accuracy of unseen data also reveals that scheduling knowledge bases by the proposed attribute selection approach to constructing DTs can avoid overfitting the training data compared with the classical DT learning approach.

1. Introduction

In the dynamic manufacturing environment, scheduling decisions are usually implemented through dispatching rules that assign priority indices to various jobs waiting at a machine (or buffer) where the job with the highest priority is performed next. Many researchers (Blackstone *et al.* 1982, Baker 1984, Montazeri and Van Wassenhove 1990, Sabuncuoglu 1998) have been studying dispatching rules in a variety of configurations since the 1960s. They have come to the major conclusion that no single dispatching rule has been shown to consistently produce better results than other rules under a variety of shop configuration conditions and performance criteria.

Since the values of manufacturing attributes change continually in a dynamic system, Baker (1984) suggested that it is possible to improve system performance by implementing a scheduling policy rather than a single dispatching rule. Such a scheduling policy should be enabled with adaptively scheduling heuristics at various time points. This technique is called the adaptive scheduling approach because it should be able to discover the current status of the manufacturing system, and then determine the most appropriate dispatching rule to be used for the next scheduling period.

Using the decision tree (DT) learning approach in an adaptive scheduling mechanism to improve the production performance of a manufacturing system has displayed outstanding outcomes in recent research (Shaw *et al.* 1992, Park *et al.* 1992, Kim *et al.* 1998, Arzi and Iaroslavitz 2000). DT learning is one of the most widely used and practical methods for inductive inference. It is a method to approximate the numeric or symbolic valued target function that is robust to noisy data and capable of learning disjunctive expressions. The major advantages of the DT learning approach to constructing an adaptive scheduling mechanism can be stated as follows.

(1) The concept (knowledge) learned from training examples can not only classify the given

Authors: Yeou-Ren Shiue, Department of Industrial Management, Van Nung Institute of Technology, Chungli, Taiwan. Chao-Ton Su, Department of Industrial Engineering and Management, National Chiao Tung University, Hsinchu, Taiwan.

International Journal of Computer Integrated Manufacturing
ISSN 0951-192X print/ISSN 1362-3052 online © 2003 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/0951192021000025689

examples accurately but can also predict the unseen examples very well.

(2) The learned function can be represented by either a DT or sets of if–then rules in order to improve human readability. It can also be easily coded into a control mechanism.

(3) DT learning methods are robust to errors in classification of the training examples and errors in the attribute values that describe these examples.

(4) DT learning methods can be used even when the values of some training examples are unknown.

These capabilities of DT learning methods satisfy the needs generated by the dynamic nature of the manufacturing environment. Therefore, the DT-based learning approach used for online scheduling is applicable to decision problems in the manufacturing environment. Such problems, which classify training examples into one of a discrete set of possible categories (scheduling rules), are often referred to as classification problems.

Most algorithms that have been developed for learning DT are variations of a core algorithm that employs a top-down, greedy search through the space of possible DTs. This approach, exemplified by the ID3 algorithm (Quinlan 1986) and its successor C4.5 (Quinlan 1993), forms the primary focus of research in DT learning.

Generalization is an important ability specific to DT learning that predicts unseen data with a high accuracy, based on learned concepts from training examples. Like other inductive learning methods, ID3 employs a top-down, divide-and-conquer strategy that partitions the given set of training examples into smaller and smaller subsets, in steps, as the tree grows to the depth just enough for perfect classification of the training examples. This is a fairly reasonable strategy in some cases. However, it can cause difficulties when there is noise in the data (or even when the training examples are noise-free), or when the number of training examples is too small to produce a representative sample of the true target function. In either of these cases, the DT algorithm can produce trees that overfit the training examples. This phenomenon is usually called *overfitting*.

In order to avoid overfitting and improve the generalization ability in DT learning, one quite successful method for finding high accuracy hypotheses uses a technique called rule post-pruning. However, due to the existence of irrelevant and redundant attributes in manufacturing systems, overfitting is still a problem that needs to be resolved in the selection of proper attributes to describe the training examples that represent hypothesis space in an early stage. Even though ID3 uses information gain to measure the effectiveness of an attribute in classifying the training data (Quinlan 1986), it still creates DTs that are too long to fit training examples. Mitchell (1997) indicated that the ID3 algorithm could characterize its inductive bias approximately by its preference for short DTs to complex trees. The term *inductive bias* here is the set of assumptions that, together with the training data, deductively justify the classifications assigned by the learner to future instances. From the above discussion, since the essential attributes are uncertain in manufacturing systems, how to select proper attributes for adaptive scheduling so as to enhance knowledge representation in the DT learning approach is worth studying in the adaptive scheduling problem domain.

In order to capture the scheduling knowledge concept, we need to determine the mapping function between system attributes and the indices of dispatching rules under various performance criteria. The artificial neural networks (ANNs) as learning tools have demonstrated the ability to capture the general relationships among variables that are difficult or impossible to relate to each other analytically by learning and generalizing from training patterns or data. Hence, using ANNs as the mapping function in scheduling knowledge provides a reasonable solution tool to abstract proper attributes.

This study aims to develop a hybrid ANN/DT approach to enhance the knowledge representation of the DT learning approach in the adaptive scheduling problem domain, which is called an attribute selection decision tree (ASDT)-based learning adaptive scheduling system. The proposed approach requires less computation efforts and possesses the enhanced generalization ability of inductive bias. In addition, it can avoid overfitting training examples. In the long run, the ASDT-based adaptive scheduling system can be expected to outperform classical DT-based learning adaptive scheduling in the absence of an attribute selection procedure under various production performance criteria.

2. Literature review

2.1. *The strategies to enhance knowledge representation in DT learning*

For a detailed description of the ID3 algorithm and C4.5, see Quinlan's (1986, 1987, 1993) works. In brief, the approach begins with a set of training examples consisting of an attribute-value list. Each example

belongs to a particular known class. The goal is to develop a series of rules that will further classify the examples correctly into one of these classes, when only the values of the attributes of the examples are known. The algorithm examines each attribute in turn and measures by information gain how attributes differentiate between classes. The best attribute is chosen, and the data are partitioned into subsets according to the values of that attribute. This process is recursively applied to each subset until all the examples are correctly classified. The result is a tree in which nodes represent attributes and branches represent all possible attribute values or ranges of values. Terminal nodes (leaves) of the tree correspond to the sets of examples in the same class.

One important advantage of inductive learning from examples is its ability to generalize and apply to new situations. Generalization begins where learning ends. The concept learned from a number of examples can induce a complete function that often works well on unseen data. That is, the concept learned from examples can not only explain the given examples, but also predict unseen data with high accuracy.

There are two important criteria on generalization ability for evaluating decision learning (Mingers 1989a, b):

(1) Size. It is generally accepted that shorter trees are preferred to longer trees. This is particularly true in the case with statistical models, in which complexity usually improves explanatory power on the training data, but deteriorates the predictive ability of the model on independent test data.
(2) Accuracy. This refers to a DT's predictive ability to classify an independent set of test data. It is measured by the error rate, i.e. the proportion of incorrect predictions that a tree makes on the test data.

There are three phases to rule induction in DT learning: first, create an initial, large rule tree from the sets of examples based on attribute selection measures; second, prune this tree by removing the branches with little statistical validity; and third, process the pruned tree to improve its understandability. The generalization ability in DT learning can be enhanced through the first two phases even if there may be errors in the classifications of the training examples and errors in the attribute values that describe these examples.

In the tree creation phase, the central task in the ID3 algorithm is selecting the most useful attributes for classifying examples to test at each node. What is a good quantitative measure of the worth of an attribute? Quinlan (1986) proposed an evaluation function called *information gain* that measures how well a given attribute separates the training examples according to the target classification. ID3 uses this information gain to select among the candidate attributes at each step while growing the tree.

There is a natural bias in the information gain measure in that it favours the attributes with many values over those with few values. One alternative measure that has been used successfully is the gain ratio (Quinlan 1986). The gain ratio measure penalizes attributes by incorporating the normalizing factor, called *split information*, which is sensitive to how broadly and uniformly the attribute splits the data.

Mingers (1989a) provided an experimental analysis of the relative effectiveness of several measures for selecting attributes over a variety of problems. He reported significant differences in the size of the unpruned trees produced by the different selection measures. However, in his experimental domains, the choice of attribute selection measure appears to have a smaller impact on final accuracy than the post-pruning method.

Owing to its underlying characteristics, the ID3 algorithm can lead to difficulties when there is noise in the data. When ID3 classifies such data, the resulting tree tends to be very large. However, many branches reflect the chance of occurrence of particular data rather than representing underlying relationships. These relationships are very unlikely to occur in further examples. In this situation, the ID3 algorithm will produce trees that overfit the training examples.

In the pruning trees phase, the least reliable branches are identified and removed. Pruning a tree will increase the number of classification errors made on the training data, but should decrease the error rate on independent test data.

One quite popular tree pruning method is called *rule post-pruning*. A variant of this pruning method is used by C4.5. Rule post-pruning involves the following steps.

(1) Infer the DT from the training set, grow the tree until the training data are fit as well as possible, and allow overfitting to occur.
(2) Convert the learning tree into an equivalent set of rules by creating one rule for each path from the root node to a leaf node.
(3) Prune each rule by removing any preconditions that will improve its estimated accuracy.
(4) Sort the pruned rules by their estimated accuracy, and consider them in this sequence when classifying subsequent instances.

One method to estimate rule accuracy is to use a test set of examples disjoint from the training set. This DT pruning technique comes from Quinlan's (1987) reduced-error pruning approach that produces a series of pruned trees by using the test data directly, rather than merely selecting the best tree. It can produce the smallest version of the most accurate tree with respect to the test set.

Another method used by C4.5 is to evaluate performance based on the training set itself, using a pessimistic error pruning method proposed by Quinlan (1986). It aims to avoid the requirement for a separate test data set. In this approach, C4.5 calculates its pessimistic estimate by calculating the rule accuracy over the training examples to which it applies and then calculates the standard deviation in this estimated accuracy, assuming a binomial distribution. For a given confidence level, the lower-bound estimate is then taken as the measure of rule performance. When the presence of a leaf node leads to a greater number of predictable errors it is pruned from the tree.

So far, most research works have been focusing on the impact of generalization ability in DT learning where the hypothetical attributes are limited and given in advance. However, due to the existence of irrelevant and redundant attributes in manufacturing systems, how to select the appropriate attributes to describe the training examples and represent knowledge bases in early stage is still a problem that needs to be resolved. If too many irrelevant attributes are used in describing the training examples, longer DTs will be created to fit the training examples and thus the generalization ability of knowledge bases will be hurt. Due to the fact that the essential attributes are uncertain in manufacturing systems, how to select important manufacturing attributes to enhance the generalization ability of a knowledge base's generalization ability and avoid overfitting in DT learning is a crucial research issue for an adaptive scheduling problem domain.

2.2. *Attribute selection approaches*

Siedlecki and Sklansky (1988) gave an overview of combinatorial feature selection methods, described the limitations of the methods such as artificial intelligence (AI) methods using graph searching techniques or branch-and-bound search algorithms, and indicated these methods are not feasible for large-scale problems (they considered a 20-element selection problem to be a large-scale domain). To handle large-scale problems, they described the potential benefits of Monte Carlo approaches, such as simulated annealing and genetic algorithms (GAs). In data mining of inputs for the neural networks approach, several researchers (Garson 1991, Wong *et al.* 1995) examined the use of the weight matrix of the trained neural network itself to determine which inputs are significant.

In the adaptive scheduling problem domain, Chen and Yih (1996) proposed a neural network based approach to identify the essential attributes for a knowledge-based scheduling system. In their approach, a penalty function was developed to measure how much the performance of the network will degrade from the upper bound when the information of an attribute is omitted. The major conclusion of their experiment is that the scheduling knowledge bases using the set of selected attributes are superior in choosing desired dispatching rules under unknown production conditions, compared with the knowledge bases built by other sets of attributes. A weakness of Chen and Yih's (1996) approach is that the attribute reduction process requires extensive computational effort and each dispatching rule has equal weight in the significant score function when chosen for the next control period.

Chen and Yih (1999) proposed a FSSNCA (feature subset selection based on nonlinear correlation analysis) procedure not only to select essential attributes, but also to generate important attributes to facilitate the development of knowledge bases and enhance the generalization ability of resulting knowledge bases. However, their experimental results did not indicate whether the FSSNCA procedure could avoid overfitting the training data. Moreover, their approach did not verify whether FSSNCA is robust under various performance criteria.

3. The architecture of the ASDT-based adaptive scheduling system

Figure 1 illustrates the ASDT-based adaptive scheduling system. The proposed system includes three phases: building the simulation model to obtain training examples, developing the attribute selection algorithm to select important attributes, and generating adaptive scheduling knowledge for the online scheduling control mechanism.

In this section, we will first give an overview of the basic concept of the ASDT-based adaptive scheduling. Next, we will describe the specifications of the training examples. Finally, we will discuss the attribute selection algorithm as the core mechanism of ASDT-based adaptive scheduling.

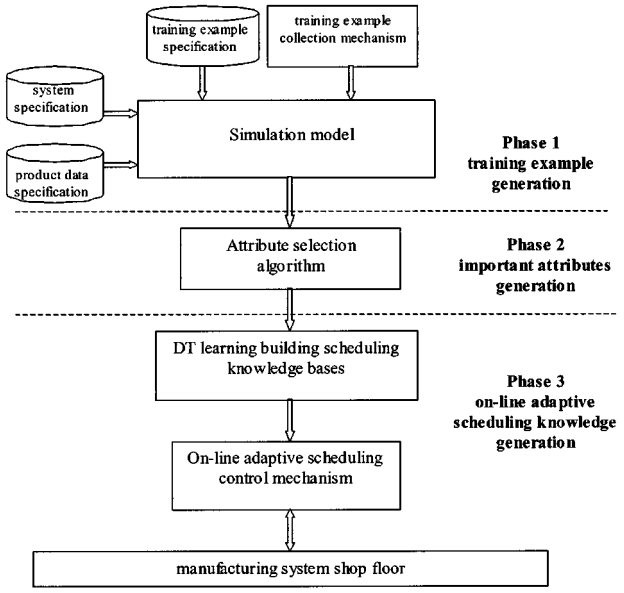

Figure 1. The architecture of the ASDT-based adaptive scheduling system.

3.1. *Overview of the ASDT-based adaptive scheduling system*

The first phase is to collect a set of training examples in the ASDT-based adaptive scheduling system. The input data of this stage include the specifications of the manufacturing system, production data, and training examples (to be described in section 3.2), and the training example collection mechanism to build the simulation model for offline learning training examples. In this phase, the training example collection mechanism must provide a comprehensive initial knowledge base, which represents a wide range of possible system states. In order to reach this goal, we use a multi-pass simulation (Wu and Wysk 1989) approach as a tool of the training example collection mechanism.

The second phase is an important attribute generation phase. In this phase, the set of training examples is put into the BP neural network for offline learning. The task of the training phase is to determine the weights of neural networks so that the input/output (system attributes/dispatching rules) mapping functions can be captured by neural networks. When the values of network weights are generated, we then calculate the weights of neural networks (to be described in section 3.3) to identify important system attributes. This stage concludes with the set of training examples with important attributes.

In the third phase, the DT learning algorithm (e.g. C4.5) learns the whole set of training examples with important attributes obtained from the second stage in order to generate scheduling knowledge bases. When DT learning is completed, the manufacturing system will receive a scheduling control period signal from the online adaptive scheduling control mechanism and then input the current system status into a DT rule-base for online scheduling control. Under this architecture, the ASDT-based adaptive scheduling system can easily identify important attributes to build sound adaptive scheduling knowledge bases and enhance the generalization ability of inductive bias. Furthermore, the ASDT-based adaptive scheduling system can significantly improve the performance of the manufacturing system compared with the classical DT-based learning adaptive scheduling in the absence of an attribute selection procedure under various criteria in the long run.

3.2. *Presentation of training examples*

A set of training examples is provided as system information to learn the concept representing each class. A given training example consists of a vector of attribute values and the corresponding class. A concept learned can be described by a rule determined by the machine learning approach, such as DT learning. If a new set of input attributes satisfies the conditions of this rule, then it belongs to the corresponding class. Baker (1984) indicated that the relative effectiveness of a scheduling rule depends on the state of the system, given performance criteria. Hence, in order to build the scheduling knowledge bases, training examples must have enough information to reveal this property.

In adaptive scheduling knowledge bases, a set of training examples can be represented by triplet {P, S, D}. P denotes the user-defined management performance criteria; S is the set of system status; D represents the best dispatching rule under such performance criteria and system status.

Three kinds of performance criteria are usually studied in adaptive scheduling research: throughput based, flow-time based, and due-date based. In order to compare the efficiency of the adaptive scheduling system with that of other dispatching rules with respect to different performance criteria, four performance criteria are used in this study as illustrated in table 2 (to be described later in section 4).

In the manufacturing environment, jobs arrive randomly and system attributes change over time. Because of the exponentially growing complexity of the underlying optimization problem, scheduling decisions in such systems are usually specified in terms of dispatching rules. Whenever a machine becomes idle, which job should be processed next on the machine must be determined. This selection is done by assigning a priority index to various jobs competing for the given

machine. The job with the highest priority will be selected. Dispatching rules differ in how they assign these priority indices. The need for adaptive dispatching rules arises from the fact that no single dispatching rule has been proven to be optimal for a manufacturing environment (Blackstone *et al.* 1982, Baker 1984, Montazeri and Van Wassenhove 1990, Sabuncuoglu 1998). That is, no dispatching rule has shown to consistently cost less in total than all other rules under a variety of shop configurations and operating conditions. Therefore, it is not necessary to spend excessive efforts in studying the best dispatching heuristics in different environments. We select nine dispatching rules that have been found to be effective for the objective in terms of the four performance criteria in this study. Table 3 (to be described later in section 4) lists these nine dispatching rules selected in this study.

This study aims to identify important system attributes under different performance criteria. Therefore, we have exhaustively examined all possible system attributes. The 30 candidate attributes examined in this study are listed in table 4 (to be described in section 4). The selection criteria for system attributes are based on previous research (Cho and Wysk 1993, Chen and Yih 1996, Arzi and Iaroslavitz 2000), which used machine learning methodology to develop scheduling knowledge bases, and the case study in this research.

3.3. *Attribute selection algorithm*

The Back Propagation (BP) neural network model excels in function approximation (Rumelhart *et al.* 1986); hence, it is used in this study to capture the mapping function of adaptive scheduling knowledge bases. The architecture of BP neural network is shown in figure 2. The model can establish the relationships between the system state attributes and the dispatching rules under various performance criteria.

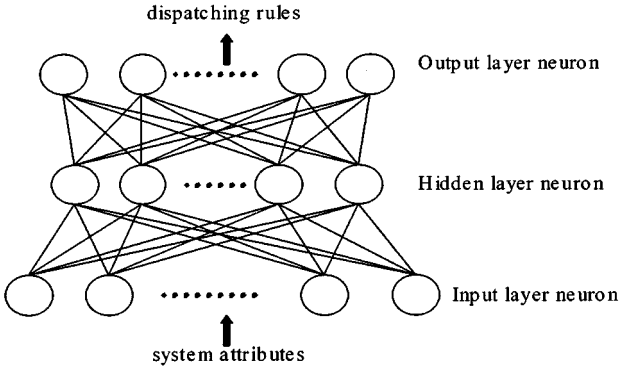

Figure 2. The BP neural network model.

When the BP neural network training process is completed, the values of network weights are generated. The interconnections of all the neurons provide essential information on the BP network architecture. The weight w_{ij} represents the strength of the synapse (called the connection or link) connecting neuron i (source) to neuron j (destination). Positive weights have an excitatory influence whereas negative values of weight have an inhibitory influence. A zero value in weights means no connection between the two neurons. In the BP network, the output of the network depends on both the weights from input to hidden and the weights from hidden to output. Therefore, combining these two sets of weights into a measure of the importance of input neurons is reasonable.

Based on the above viewpoint, we use the following measure, proposed by several researchers (Garson 1991, Wong *et al.* 1995), on the proportional contribution of an input to a particular output:

$$Q_{jk} = \sum_j \left(\frac{|w_{ij}|}{\sum_i |w_{ij}|} \times \frac{|w_{jk}|}{\sum_j |w_{jk}|} \right), \tag{1}$$

where i is the input layer neuron index; j is the hidden layer neuron index; k is the output layer neuron index; w_{ij} is the weight from the input layer of neuron i to the hidden layer of neuron j; w_{jk} is the weight from the hidden layer of neuron j to the output layer of neuron k.

The measure of the input neuron introduced here is an extension of equation (1). We can define the attribute selection score of input neuron i by:

$$AS_i = \frac{\sum_k Q_{jk}}{k} \tag{2}$$

where k is the output layer neuron index and also represents the number of dispatching rules used in this study.

In equation (2), a higher score means that the attribute is more important to this specific input neuron. A major problem is how many attributes should be brought in to the BP network model for training. The relevant research offers no definite answer. In this study, we set a threshold value that is equal to the reciprocal of the value of studied attributes. If the attribute selection score is below this threshold value, the corresponding input neuron can be deleted from the BP model.

Based on the assumptions discussed above, the attribute selection algorithm selects important attributes by the procedures summarized below.

Step 1. According to the specifications of training examples from phase 1, build the BP neural

network configuration for each performance criterion.

Step 2. Conduct BP neural network training to capture the mapping function of adaptive scheduling knowledge bases.

Step 3. For each BP neural network under various performance criteria, calculate the attribute selection score based on equations (1) and (2).

Step 4. If $AS_i \geqslant 1/N$ then select attribute i into the subset of important attributes, where N is the number of studied attributes.

4. Study case description

The study case is a modification of the model used by Montazeri and Van Wassenhove (1990). The FMS case in the study consists of three machine families (F1, F2 and F3), three load/unload stations, three AGVs, an input buffer and a central WIP buffer with limited capacity, and a computer-controlled local area network. The first two machine families have two machines and the third family has one machine. There are 11 different types of parts to be produced in this model and their processing times are identical to those used by Montazeri and Van Wassenhove (1990). In order to achieve different conditions in terms of machine load and shifting bottleneck, we design five types of product mix ratios as shown in table 1, which will be continuously changed over constant time periods.

Several related operating assumptions are listed below.

(1) It is assumed that the raw materials for each type of part are readily available.
(2) Each job order arrives at random at the FMS and consists of only one part with an individual due date.
(3) A part with a pallet travels to each machine or load/unload station in order to achieve operation flexibility, and the part–type match for one specific pallet problem is not considered.
(4) Each machine can execute only one job order at a time.
(5) Each machine is subject to random seed failures.
(6) Processing times are assumed to be pre-determined.
(7) An idle machine in a family has a higher priority than other machines to process a part. If there is no idle machine in the family, then the part goes to the machine of the lowest utilization.
(8) When the part finishes each step of the process, it must return to one of the available load/unload stations for reorientation. Otherwise, it will go to the central WIP buffer to wait for the next operation (part reorientation in load/unload stations).
(9) An AGV can carry only one piece of a part at a time and move in the counter-clockwise direction only.
(10) All material movements not using the AGV system are assumed to be negligible.

Table 1. Part mix ratio used in this study.

Part ID	Part mix ratio (%)				
	Type 1	Type 2	Type 3	Type 4	Type 5
1	11.00	14.00	6.00	9.00	14.00
2	11.00	14.00	6.00	9.00	14.00
3	11.00	15.00	6.00	9.00	14.00
4	12.00	10.00	15.00	8.00	15.00
5	6.00	12.00	15.00	13.00	7.00
6	8.00	8.00	9.00	12.00	5.00
7	8.00	5.00	8.00	3.00	5.00
8	7.00	3.00	8.00	9.00	4.00
9	7.00	3.00	7.00	8.00	4.00
10	2.50	1.00	4.00	1.00	6.00
11	16.50	15.00	16.00	19.00	12.00

Based on our case study, the following notation will be useful for defining training examples. The performance criteria, system attributes, and dispatching rules from this study are summarized in tables 2, 3 and 4, respectively.

Notation	Definition
t	Time when a decision is to be made.
i	Job index.
j	Operation index.
k	Machine index (K=number of machines).
a	AGV index (A=number of AGVs).
l	Load/unload station index (L=number of load/unload stations).
p	Pallet buffer index (P=number of pallet buffers).

Table 2. Performance criteria used in this study.

Performance criteria	Description	Mathematical definition
TP	Throughput	$\|SF\|$
MF	Mean Flow Time	$\frac{\sum_{i \in SF} F_i}{\|SF\|}$
MT	Mean Tardiness	$\bar{T} = \frac{\sum_{i \in SF} T_i}{\|SF\|}$
NT	Number of the tardy parts	$\sum_{T_i > 0} 1$

Table 3. FMS system attributes used in this study.

System attribute	Description	Mathematical definition
NJ	Number of the jobs in the system	$\|SJ\|$
MeUM	The mean utilization of machines	$\frac{\sum_k W_k}{K}$
SdUM	The standard deviation of machine utilization	$\sqrt{\sum_k \frac{[W_k - \text{MeUM}]^2}{K-1}}$
MeUL	The mean utilization of load/unload stations	$\frac{\sum_l W_l}{L}$
MeUB	The mean utilization of pallet buffers	$\frac{\sum_p W_p}{P}$
MeUA	The mean utilization of AGVs	$\frac{\sum_a B_a}{A}$
MiOT	The minimum imminent operation time of candidate jobs within the system	$\min_{i \in SJ}\{P_{ij}\}$
MaOT	The maximum imminent operation time of candidate jobs within the system	$\max_{i \in SJ}\{P_{ij}\}$
MeOT	The mean imminent operation time of candidate jobs within the system	$\frac{\sum_{i \in SJ} P_{ij}}{\|SJ\|}$
SdOT	The standard deviation of the imminent operation time of candidate jobs within the system	$\sqrt{\sum_{i \in SJ} \frac{[P_{ij} - \text{MeOT}]^2}{\|SJ\|-1}}$
MiPT	The minimum total processing time of candidate jobs within the system	$\min_{i \in SJ}\{\sum_j P_{ij}\}$
MaPT	The maximum total processing time of candidate jobs within the system	$\max_{i \in SJ}\{\sum_j P_{ij}\}$
MePT	The mean total processing time of candidate jobs within the system	$\frac{\sum_{i \in SJ}\sum_j P_{ij}}{\|SJ\|}$
SdPT	The standard deviation of the total processing time of candidate jobs within the system	$\sqrt{\sum_{i \in SJ} \frac{[(\sum_j P_{ij}) - \text{MePT}]^2}{\|SJ\|-1}}$
MiRT	The minimum remaining processing time of candidate jobs within the system	$\min_{i \in SJ}\{\sum_{j \in SR_i} P_{ij}\}$
MaRT	The maximum remaining processing time of candidate jobs within the system	$\max_{i \in SJ}\{\sum_{j \in SR_i} P_{ij}\}$
MeRT	The mean remaining processing time of candidate jobs within the system	$\frac{\sum_{i \in SJ}\sum_{j \in SR_i} P_{ij}}{\|SJ\|}$
SdRT	The standard deviation of the remaining processing time of candidate jobs within the system	$\sqrt{\sum_{i \in SJ} \frac{[(\sum_{j \in SR_i} P_{ij}) - \text{MeRT}]^2}{\|SJ\|-1}}$
MiST	The minimum slack time of candidate jobs within the system	$\min_{i \in SJ}\{SL_i\}$
MeST	The mean slack time of candidate jobs within the system	$\frac{\sum_{i \in SJ}\{SL_i\}}{\|SJ\|}$
SdST	The standard deviation of the slack time of candidate jobs within the system	$\sqrt{\sum_{i \in SJ} \frac{[(SL_i) - \text{MeST}]^2}{\|SJ\|-1}}$
MaTA	The maximum tardiness of candidate jobs within the system	$\max_{i \in SJ}\{T_i\}$
MeTA	The mean tardiness of candidate jobs within the system	$\frac{\sum_{i \in SJ}\{T_i\}}{\|SJ\|}$

Continued

Table 3. (Continued).

System attribute	Description	Mathematical definition
SdTA	The standard deviation of the tardiness of candidate jobs within the system	$\sqrt{\sum_{i\in SJ}\frac{[(T_i)-\mathrm{MeTA}]^2}{\lvert SJ\rvert-1}}$
MaWL	The maximum workload in front of any machine/station within the system	$\max_{k\cup l}\{\sum_{i\in SJ}\sum_{j\in SR_i}P_{ij}^k \cup \sum_{i\in SJ}\sum_{j\in SR_i}P_{ij}^l\}$
ToWL	The total workload in front of any machine/station within the system	$\sum_{i\in SJ}\sum_{j\in SR_i}P_{ij}$
MeSO	The mean sojourn time of candidate jobs within the system	$\sum_{i\in SJ}\frac{t-AR_i}{\lvert SJ\rvert}$
SdSO	The standard deviation of the sojourn time of candidate jobs within the system	$\sqrt{\sum_{i\in SJ}\frac{[(t-AR_i)-\mathrm{MeSO}]^2}{\lvert SJ-1\rvert}}$
MeTD	The mean time now until due date of candidate jobs within the system	$\sum_{i\in SJ}\frac{D_i-t}{\lvert SJ\rvert}$
SdTD	The standard deviation of the time now until due date of candidate jobs within the system	$\sqrt{\sum_{i\in SJ}\frac{[(D_i-t)-\mathrm{MeTD}]^2}{\lvert SJ-1\rvert}}$

Table 4. Dispatching rules used in this study.

Dispatching rule	Description	Mathematical definition
FIFO	Select the job according to the first-in first-out rule	$\min_{i\in SJ}\{AR_i\}$
SPT	Select the job with the shortest processing time	$\min_{i\in SJ}\{\sum_j P_{ij}\}$
SIO	Select the job with the shortest imminent operation time	$\min_{i\in SJ}\{P_{ij}\}$
SRPT	Select the job with the shortest remaining processing time	$\min_{i\in SJ}\{\sum_{j\in SR_i}Pij\}$
CR	Select the job with the minimum ratio between time now until due-date and its remaining processing time	$\min_{i\in SJ}\left\{\frac{D_i-t}{\sum_{j\in SR_i}P_{ij}}\right\}$
DS	Select the job with minimum slack time	$\min_{i\in SJ}\{D_i-t-\sum_{j\in SR_i}P_{ij}\}$
EDD	Select the job with the earliest due-date	$\min_{i\in SJ}\{D_i\}$
MDD	Select the job with the minimum modified due-date	$\min_{i\in SJ}\{\max(D_i,\ t+\sum_{j\in SR_i}P_{ij})\}$
MOD	Select the job with the minimum modified operation due-date	$\min_{i\in SJ}\{\max(D_i-\sum_{j\in SR_i}P_{ij}, t+\sum_{j\in SR_i}P_{ij})\}$

W_k Percentage of work time on machine k in the simulation period.
W_a Percentage of work time on AGV a in the simulation period.
W_l Percentage of work time on load/unload station l in the simulation period.
W_p Percentage of work time on pallet buffer p in the simulation period.
P_{ij} Processing time of the jth operation of job i.
P_{ij}^k Processing time of the jth operation of job i in machine k.
P_{ij}^l Processing time of the jth operation of job i in load/unload station l.
AR_i Time when job i arrives at the system.
D_i Due date of job i.
C_i Time when job i is completed and leaves the system.
SJ Set of jobs within the system.
SF Set of finished jobs.

SR_i Set of remaining operations of job i.
$|SJ|$ The cardinality of SJ.
$|SF|$ The cardinality of SF.
F_i Flowtime of the job i (F_i=C_i-AR_i).
T_i Tardy time of the job i (T_i=max(0,C_i-D_i)).
SL_i Slack time of the job i ($SL_i = D_i - t - \sum_{j \in SR_i} P_{ij}$).

5. Experiment

5.1. *Simulation model construction and training example generation*

To verify the proposed methodology, a discrete event simulation model is used to generate training examples. The simulation model was built and executed using SIMPLE++ (2000) object-oriented simulation language and was run on a Pentium III PC with Windows 2000 system.

Several parameters are determined by a preliminary simulation run. The time between job arrivals is exponentially distributed with a mean of 31 minutes. The due date of each job is randomly assigned from 6 to 10 times the total processing time and is uniformly distributed. The maximum number of pallets (jobs) that are allowed to be within the FMS system is limited to 100 pallets. The proportions of part types vary continuously every 20 000 minutes in this study.

The training examples are generated by executing a simulation run for every dispatching rule given the same initial state of the system attribute and arriving job stream, according to the declaration by Arzi and Iaroslavitz (2000). In addition, in order to provide comprehensive training examples, which will represent the wide breadth of possible system states, the technique of multi-pass simulation is utilized to collect the training examples, including the state variable of the system attribute recorded at the decision point and the performance measure of each dispatching rule recorded at the end of the scheduling point.

To generate a large number of various training examples, we used 40 different random seeds and chose, from the simulation clock, 1000 to 5000 minutes (1000 minutes for one unit) to generate 200 different job arriving patterns. The warm-up period for each run is 10 000 minutes followed by 10 multi-pass scheduling periods, each of which is 10 000 minutes (after the warm-up period) depending on a trial-and-error process for each performance criterion. There are 2000 training examples collected in total, which are then divided into the training set and test set arbitrarily. Each set contains 1000 training examples.

5.2. *Mapping function formation for BP network learning and important attribute selection*

In this phase, neural networks are used to capture the mapping function between the attributes of the system state and the dispatching rules under various performance criteria. The BP neural network model used in this study is implemented by NeuralWorks Professional II Plus (2000) software. Some important experimental parameters in the BP network model used in this study are described in the following:

The initial network connection weights: [−0.5, +0.5]
Learning rate: 0.3, 0.4, and 0.5
Momentum: 0.4 and 0.9
Initial bias: 0.5
Learning rule: generalized delta-rule
Transfer function: sigmoid
Scaled input neuron network range: [−3, +3]
Scaled output neuron network range: [0, +1]
Hidden layer neuron range: [3, 25]
Maximum iterations: 100 000

Based on the above, the training process is implemented 138 times in total for each performance criterion. Next, the training examples are put into BP network models for offline learning. Table 5 shows the topology and learning parameters of BP network models after a training process is chosen to provide the mapping function of adaptive scheduling knowledge bases.

The values of network weights are generated after the training process. We use the proposed attribute selection algorithm to identify important system attri-

Table 5. The design parameters of selected BP network models in the attribute selection process.

Performance criteria	Topology	Learning rate	Momentum	RMS error of testing data
TP	30-21-9	0.4	0.9	0.0756
MF	30-8-9	0.5	0.4	0.0577
MT	30-25-9	0.4	0.9	0.0380
NT	30-15-9	0.4	0.9	0.0567

butes. The scores of system attributes for each performance criterion are presented in table 6 and the results of the selected attributes are shown in table 7.

5.3. *DT knowledge base development and generalization ability verification*

In order to verify whether the selected attributes for a case study can provide greater generalization ability in building scheduling knowledge bases, we design two groups of attribute subsets for the training examples, one of which includes all 30 attributes whereas the other one includes only the important attributes selected from the attribute selection algorithm. In this study, we use C4.5 program code (Quinlan 1993) as the DT learning tool to build scheduling knowledge bases. To achieve greater generalization ability by means of the post-pruning technique in knowledge bases for each DT learning procedure concerned in this study, C4.5 can be tuned by two parameters: confidence level of pruning (*C* parameter) and the minimal number of examples represented at any branch of any feature-value test (*M* parameter). We use default settings for the two parameters under this study and the two parameters are described in the following:

(1) The *C* parameter denotes the confidence level of pruning, which ranges between 0% and 100%. This parameter is used in a heuristic function that estimates the predicted number of misclassifications of unseen instances at the leaf nodes, by computing the binomial probability (i.e. the confidence limits for the binomial distribution) of misclassifications within the set of instances represented at those nodes. If the presence of a leaf node leads to a higher predicted number of errors than its absence, then it is pruned from the tree. By default, *C*=25%; set at 100%, no pruning occurs. The more pruning is performed, the less information about the individual example is remembered in the abstracted decision tree.

(2) The *M* parameter determines the minimum number of instances represented by a node. By setting $M > 1$, C4.5 can avoid the creation of long paths involving the minority of obscure individual instances that most likely represent noise. By default, *M*=2. With *M*=1, C4.5 builds a path for every single instance not yet disambiguated. A higher value of *M* leads to an increasing level of abstraction and therefore less recoverable information about individual instances.

To verify the generalization ability in the ASDT-based adaptive scheduling system, we divide the 2000

Table 6. The attribute selection score for each performance criterion

System attribute	Attribute selection score TP	MF	MT	NT
NJ	0.0408	0.0651	0.0897	0.0939
MeUM	0.0361	0.0198	0.0260	0.0361
SdUM	0.0494	0.0309	0.0365	0.0185
MeUL	0.0282	0.0135	0.0174	0.0256
MeUB	0.0241	0.0204	0.0258	0.0231
MeUA	0.0142	0.0198	0.0179	0.0197
MiOT	0.0257	0.0323	0.0315	0.0211
MaOT	0.0262	0.0176	0.0201	0.0308
MeOT	0.0335	0.0342	0.0178	0.0171
SdOT	0.0364	0.0388	0.0460	0.0471
MiPT	0.0294	0.0198	0.0331	0.0275
MaPT	0.0369	0.0316	0.0336	0.0269
MePT	0.0264	0.0773	0.0381	0.0686
SdPT	0.0478	0.0544	0.0580	0.0271
MiRT	0.0308	0.0241	0.0315	0.0339
MaRT	0.0269	0.0253	0.0232	0.0230
MeRT	0.0821	0.0450	0.0665	0.0634
SdRT	0.0688	0.0691	0.0266	0.0721
MiST	0.0344	0.0191	0.0321	0.0247
MeST	0.0195	0.0293	0.0222	0.0261
SdST	0.0226	0.0218	0.0286	0.0237
MaTA	0.0362	0.0256	0.0294	0.0287
MeTA	0.0230	0.0322	0.0256	0.0178
SdTA	0.0217	0.0214	0.0272	0.0210
MaWL	0.0189	0.0445	0.0233	0.0109
ToWL	0.0897	0.0582	0.0613	0.0783
MeSO	0.0169	0.0317	0.0254	0.0227
SdSO	0.0223	0.0194	0.0311	0.0257
MeTD	0.0155	0.0305	0.0248	0.0230
SdTD	0.0177	0.0263	0.0298	0.0216

Table 7. The results of selected attributes for each performance criterion.

Performance criteria	Important attribute subset	No. of attributes selected
TP	{NJ, MeUM, SdUM, MeOT, SdOT, MaPT, SdPT, MeRT, SdRT, MiST, MaTA, ToWL}	12
MF	{ NJ, MeOT, SdOT, MePT, SdPT, MeRT, SdRT, MaWL, ToWL}	9
MT	{ NJ, SdUM, SdOT, MaPT, MePT, SdPT, MeRT, ToWL}	8
NT	{ NJ, MeUM, SdOT, MePT, MiRT, MeRT, SdRT, ToWL}	8

training examples into a learning set and an unseen (test) set arbitrarily. Each set contains 1000 training examples. The learning set is used to build scheduling knowledge bases. The unseen set is only used to estimate the generalization ability after the attribute has been selected and the scheduling knowledge base has been constructed. Table 8 displays the size of DTs for learning (or unseen) data with or without using the attribute selection algorithm under various performance criteria. The results show that even if the post-pruning procedure is applied to each DT learning procedure, an even smaller DT can be created by using the attribute selection algorithm to select important attributes.

Table 9 shows the accuracy of the DT tree for learning and unseen data with or without using the attribute selection algorithm under various performance criteria. Although the attribute selection algorithm deteriorates the predictive ability of the learning data, as table 9 indicates, the generalization ability of knowledge bases using the proposed attribute selection algorithm can provide higher accuracy in unseen data based on all criteria. A further implication is that the proposed ASDT-based approach to constructing DTs can avoid overfitting training data compared with the classical DT learning approach to constructing DTs. Moreover, implementation results (as shown in tables 8 and 9) show that the ASDT-based approach can generate better inductive bias in DT learning by producing DTs of smaller size compared with the classical DT learning approach. Hence, the ASDT-based approach can provide greater prediction accuracy in unseen data under various performance criteria.

In order to demonstrate the robustness of the constructed knowledge bases in adaptive scheduling in dynamic manufacturing environment for a long run, we used another 25 different random seeds to generate 25 different job arriving patterns. The warm-up period for each run was 10 000 minutes followed by 20 multi-pass simulation scheduling periods, each of which was 10 000 minutes; hence, a stream of arriving jobs each of a 200 000 minute simulation run was generated by using a different set of random seeds. Table 10 shows the accuracy of DTs for unseen data in different system scenarios using the ASDT-based approach or the classical DT-based approach to construct knowledge bases (the same knowledge bases as in tables 8 and 9) under various performance criteria. As table 10 indicates, the generalization ability of knowledge bases using the proposed attribute selection algorithm can achieve superior accuracy in unseen data compared with that realized by the generation ability of knowledge bases without using the attribute selection algorithm under all criteria.

Table 8. Size of DTs under various performance criteria.

	TP	MF	MT	NT
ASDT-based approach	122	132	155	75
Classical DT-based approach	154	161	203	179

6. Conclusions

Some classical DT learning approaches, such as ID3 and C4.5, have provided several measures for attribute selection in the tree creation phase and for rule post-pruning to enhance the generalization ability in inductive bias. However, these DT learning approaches to construct DT knowledge bases are usually not considered when there exist some irrelevant and redundant attributes in the problem domain. In this study, we have developed the ASDT-based learning approach for online dispatching in dynamic manufacturing systems. The proposed attribute selection algorithm can easily measure the important system attributes that can be utilized to enhance knowledge representation. Moreover, less effort is required to build scheduling knowledge bases by using reduced system attributes. The experiment's results show that the use of the attribute selection algorithm to build scheduling knowledge bases delivers better generalization ability than in the absence of an attribute selection procedure in terms of the size of DTs under various performance criteria. Consistent results have been obtained from the prediction accuracy of unseen data in various scenarios, which also reveals that scheduling knowledge bases through the proposed ASDT-based learning approach can avoid overfitting

Table 9. Accuracy of DTs under various performance criteria.

	Learning data				Unseen data			
	TP	MF	MT	NT	TP	MF	MT	NT
ASDT-based approach	0.874	0.856	0.838	0.750	0.6840	0.612	0.618	0.650
Classical DT-based approach	0.914	0.918	0.914	0.925	0.612	0.534	0.488	0.606

Table 10. Accuracy of DTs in unseen data (different system scenarios).

	TP	MF	MT	NT
ASDT-based approach	0.6640	0.6040	0.5580	0.6560
Classical DT-based approach	0.5760	0.5440	0.4220	0.5860

compared with the classical DT learning approach to constructing DTs.

The practicability of the attribute selection algorithm can be further studied. Moreover, how to improve the generalization ability of scheduling knowledge bases corresponding to the continuously changing environment of the product mix is another important issue for further research.

References

Arzi, Y. and Iaroslavitz, L., 2000, Operating an FMC by a decision-tree-based adaptive production control system. *International Journal of Production Research*, **38**(3), 675–697.

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

Blackstone, J. H., Philips, D. T. Jr. and Hogg, G. L., 1982, A state-of-the-art survey of dispatching rules for manufacturing job shop operations. *International Journal of Production Research*, **20**(1), 27–45.

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

Chen, C. C. and Yih, Y., 1999, Auto-bias selection for learning-based scheduling systems. *International Journal of Production Research*, **37**(9), 1987–2002.

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.

Garson, G. D., 1991, Interpreting neural network connection weights. *AI Expert*, April, 47–51.

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

Mitchell, T. M., 1997, *Machine Learning* (New York: McGraw-Hill).

Mingers, J., 1989a, An empirical comparison of selection measures for decision-tree induction. *Machine Learning*, **3**(4), 319–342.

Mingers, J., 1989b, An empirical comparison of pruning methods for decision-tree induction. *Machine Learning*, **4**(2), 227–243.

Montazeri, M., and Wassenhove, L. N. van, 1990, Analysis of scheduling rules for an FMS. *International Journal Production Research*, **28**(4), 785–802.

NeuralWorks, 2000, *Professional II/Plus Reference Guide* (Pittsburgh, PA: NeuralWare).

Park, S. C., Raman, N. and Shaw, M. J., 1997, Adaptive scheduling in dynamic flexible manufacturing systems: a dynamic rule selection approach. *IEEE Transactions on Robotics and Automation*, **13**(4), 486–502.

Quinlan, J. R., 1986, Induction of decision trees. *Machine Learning*, **1**, 81–106.

Quinlan, J. R., 1987, Simplifying decision trees. *International Journal of Man–Machine Studies*, **27**, 221–234.

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

Rumelhart, D. E., Hinton, G. E. and Williams, R. J., 1986, *Learning Internal Representations by Error Propagation. Parallel Distributed Processing* (Cambridge, MA: MIT press).

Sabuncuoglu, I., 1998, A study of scheduling rules of flexible manufacturing systems: a simulation approach. *International Journal Production Research*, **36**(2), 527–546.

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

Siedlecki, W. and Sklansky, J., 1988, On Automatic feature selection. *International Journal of Pattern Recognition and Artificial Intelligence*, **2**(2), 197–220.

SIMPLE ++, 2000, *Reference Manual Version 7.0* (Stuttgart: AESOP).

Wong, P. M., Gedeon, T. D. and Taggart, I. J., 1995, An improved technique in porosity prediction: A neural network approach. *IEEE Transactions on Geoscience and Remote Sensing*, **33**(4), 971–980.

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.