1990

Intelligent scheduling with machine learning capabilities : the induction of scheduling knowledge

Michael J. Shaw
Carnegie Mellon University

Sang Chan. Park

Narayan Raman

Intelligent Scheduling with Machine Learning Capabilities: The Induction of Scheduling Knowledge§

Michael J. Shaw*†
Sang Chan Park**
Narayan Raman*
CMU-RI-TR-90-25

† Robotics Institute, Carnegie Mellon University; on leave from The Beckman Institute, University of Illinois at Urbana-Champaign
* The Department of Business Administration, University of Illinois at Urbana-Champaign
** School of Business, University of Wiscosin at Madison
§ Forthcoming in *IIE Transactions;* an earlier version appeared in the BEBR Working Paper Series No. 90-1639.

Revised November, 1990

Contents

List of Figures

List of Tables

Abstract

Dynamic scheduling of manufacturing systems has primarily involved the use of dispatching rules. In the context of conventional job shops, the relative performance of these rules has been found to depend upon the system attributes, and no single rule is dominant across all possible scenarios. This indicates the need for developing a scheduling approach which adopts a state-dependent dispatching rule selection policy. The importance of adapting the dispatching rule employed to the current state of the system is even more critical in a flexible manufacturing system because of alternative machine routing possibilities and the need for increased coordination among various machines.

This study develops a framework for incorporating machine learning capabilities in intelligent scheduling. A pattern- directed method, with a built-in inductive learning module, is developed for heuristic acquisition and refinement. This method enables the scheduler to classify distinct manufacturing patterns and to generate a decision tree consisting of heuristic policies for dynamically selecting the dispatching rule appropriate for a given set of system attributes.

Computational experience indicates that the learning-augmented approach leads to improved system performance. In addition, the process of generating the decision tree shows the efficacy of inductive learning in extracting and ranking the various system attributes relevant for deciding upon the appropriate dispatching rule to employ.

1 Introduction

Scheduling forms a part of the operational control process in a manufacturing system. The need for scheduling arises whenever a common set of resources in the manufacturing system must be shared to make a variety of different products during the same period of time. The objective of manufacturing scheduling is the efficient allocation of machines and other resources to jobs, or operations within jobs, and the subsequent time-phasing of these jobs on individual machines.

The needs of research in new approaches to manufacturing scheduling have been stimulated by a variety of pragmatic and theoretical considerations. On one hand, scheduling is a notoriously difficult problem to solve computationally; on the other hand, it is also a problem encountered in every manufacturing system and there are a great deal of financial incentives for factories to improve their scheduling practices. Global competition has enhanced the significance of manufacturing effectiveness. Better manufacturing schedules provide competitive advantage through reduced production cost and increased productivity. Moreover, global competition in the last decade has forced U.S. companies to invest in automated, capital- intensive new manufacturing systems, such as flexible manufacturing systems (FMSs). These new systems have created a range of new operational problems, making the development of new methods for scheduling these sophisticated systems increasingly important (Raman and Talbot 1985; Shaw 1986-89).

The maturation of artificial intelligence (AI) has redirected the body of scheduling research (Rodammer and White 1988; Stockey 1989). There are several capabilities of AI that make this technology particularly suitable for scheduling; these include (1) the richer, more structured, knowledge representation schemes capable of fully incorporating manufacturing knowledge, constraints, state information, and heuristics; (2) the reasoning ability enabling the scheduling systems to perform more reactive scheduling in addition to predictive scheduling; (3) the ease to integrate AI-based scheduler with other decision support systems in the manufacturing environment, such as diagnostic systems, process controllers, sensor monitors, and process planning systems; and (4) the ability to incorporate descriptive, organizationally specific scheduling knowledge usually possessed only by human expert schedulers. The

adoption of AI for factory automation is the general trend in the industry; for example, a recent survey showed that in the near future manufacturing process controllers will be mostly rule-based (Booker 1989).

However, the development of AI systems for intelligent scheduling is now at a critical junction, very much in need of new advancements to resolve a number of common difficulties encountered in applying the technology. The proposed research project is aimed at developing new methods for intelligent scheduling to address some of these issues, such as (1) how to automate the acquisition of scheduling knowledge in a given manufacturing environment? (2) how to perform dynamic, adaptive scheduling? (3) what would be the most relevant information for making such scheduling decisions. (4) how to improve the robustness of the AI-based scheduling process? (5) how best to integrate the simulation and scheduling systems for reactive- based scheduling? These research questions will be addressed in this paper by developing a new methodology using machine learning for intelligent scheduling. This methodology points to a new direction for scheduling research—the development of intelligent schedulers with machine learning capabilities. As a first step, this paper focuses on the use of inductive learning in a pattern-directed scheduling process (Shaw 1989).

Previous scheduling research has indicated that the relative effectiveness of a given scheduling rule is dependent upon the system characteristics. In a dynamic manufacturing system, these characteristics continue to change over tim. It appears conceptually appealing, therefore, to adopt an approach which employs appropriate and possibly different scheduling at various points in time. In order to do so, however, we need a mechanism which can distinguish different system characteristics, upon the rule appropriate for a given combination. This paper presents an approach to achieve these objectives by integrating pattern- directed scheduling with inductive learning.

The integration of inductive learning with pattern-directed scheduling results in an interesting scheduling approach capable of performing adaptive scheduling by selecting scheduling heuristics opportunistically; moreover, it also help identify the relative importance of a variety of manufacturing attributes in dynamic scheduling. Empirical results from simulation studies showed that this learning-augmented approach generates better scheduling performance than the traditional methods.

This paper is organized as follows. §2 discusses how machine learning can be applied in solving scheduling problems and the advantages of doing so. In §3 we describe the inductive learning process which is illustrated in §4 in the context of machine scheduling. §5 describes the generation of decision trees for selecting the appropriate scheduling rules in an FMS environment. We present an experimental study in §6 for evaluating the relative merit of this method over the single scheduling rule approach adopted in most of the previous research on dynamic scheduling. We conclude in §7 with a summary discussion of the major results.

2 Intelligent Scheduling and Machine Learning

In the recent past, several researchers have applied artificial intelligence (AI) based methods for solving scheduling problems. This body of research can best be reviewed by highlighting the focus of the AI techniques used as done below.

Scheduling as Search: Scheduling can be viewed as a process search through the state space of all possible partial and complete schedules. Search is ubiquitous in AI problems, but it is more significant an issue in scheduling problems. Several methods have been suggested in literature to alleviate the computational complexity incurred by the search process. The ISIS system (Fox and Smith 1984, Fox 1987) uses several types of constraints to reduce the state space; constraint satisfaction is used as an index to direct the search. Shaw (1986a, 1988a), Shaw and Whinston (1989a) use the combination of A^* procedure and scheduling heuristics to facilitate the search for the final schedule. The OPIS system (Ow et al. 1988) employs an opportunistic approach to improve upon ISIS. It selects the most appropriate strategy for scheduling opportunistically; the resulting flexibility achieved in problem solving results in better performance. Ow (1984) describes the beam search method for scheduling problems.

Scheduling as Planning/Replanning: One of the goals of intelligent scheduling is to be able to generate schedules more flexibly whenever alternative machine routing is possible while simultaneously taking the dynamically changing system state information into account. Thus, the scheduler not only has to decide the time sequences for performing the operations, it also has to allow for dynamic machine assignments as well. Shaw (1986a, 1988a) uses a

nonlinear planning method for deciding machine assignments and the temporal relationships among various operations. This method integrates scheduling with process planning by utilizing a two-phase procedure. In phase 1, the various machine and resource assignments for achieving the required manufacturing goals are selected. Subsequently, phase 2 works to resolve conflicts while maintaining progressive performance improvement. This approach originated with the robot planning method (Fikes and Nilsson 1971; Georgeff and Lansky 1986), and it is especially suitable for dynamic scheduling which is treated as the problem of *replanning* with a changed goal.

Scheduling as Rule-Based Inference: This method attempts to incorporate scheduling knowledge into an IF-THEN rule form which is implemented by an expert system. Wysk et al. (1986) use a multipass expert system to decide the appropriate scheduling rules based on information such as the current system status, scheduling objective and management goals. Other examples in this line of work include Raghavan (1988), Kusiak and Chen (1988) and Kusiak (1987). Bruno et al. (1986) use an expert system for knowledge representation and heuristic problem solving in the scheduling domain. In their study, the expert system is coupled with an activity-scanning scheduler adapted from discrete event simulation and a closed queueing network based algorithm for schedule analysis and performance evaluation. Another example is the ISA (Intelligent Scheduling Assistant) system developed at Digital Equipment Corporation (Kanet and Adelsberger 1987) in which approximately 300 rules were used to construct the evolving schedules.

Scheduling as Cooperative Problem Solving: Scheduling in manufacturing environment is typically performed by a group of scheduling agents. As computer integrated manufacturing makes scheduling progressively more complex because of the large number of resources, information requirements and decisions as well as a larger variety of jobs involved, the scheduling of manufacturing processes will increasingly require team effort. In such an environment, the scheduling agents can be flexible cells, machine centers, or human schedulers (Parunak 1987, Ow et al. 1988). Shaw and Whinston (1985, 1989b) and Shaw (1986b, 1988b, 1988c) apply distributed artificial intelligence to the scheduling of manufacturing cells. Using cooperative problem solving, the scheduling problem can be decomposed into several subproblems to be solved by individual agents through task sharing and parallel processing. Moreover,

this approach fits naturally into the distributed manufacturing environment in which various subsystems are interconnected through communication networks.

Machine Learning is a rapidly emerging research area for studying methods for developing artificial intelligence systems which are capable of learning (Michalski et al. 1983). The ability to learn and improve is essential for an *intelligent* system; however, little work has been done in applying machine learning to intelligent scheduling. Shaw (1989b) and Park et al. (1989) apply machine learning to identify the combination of system attributes which would lead to the use of a given scheduling rule. This knowledge can then be exploited by a pattern-directed scheduler in an adaptive fashion. In addition to heuristic learning, machine learning results in the identification of manufacturing attributes critical to the scheduling decision, and it generates an adaptive mechanism for applying the scheduling rules.

Incorporating machine learning capabilities into intelligent scheduling systems can be quite useful in enhancing scheduling performance. The potential enhancements are in the following areas: 1) Machine learning can accelerate the search process by accumulating heuristics (Shaw 1989b), 2) machine learning can facilitate the planning/replanning process by learning schemata (Shaw et al. 1988), 3) machine learning can enhance rule-based inference by automating the acquisition and the refinement of rules (Shaw 1987), and 4) machine learning can help cooperative problem solving by improving the coordination among the multiple scheduling agents (Shaw and Whinston 1989b).

3 Inductive Learning

Inductive learning can be defined as the precess of inferring the description (i. e., the *concept*) of a class from the description of individual objects of the class (Shaw 1987). A concept is a symbolic description which is true if it describes the class correctly when applied to a data case, and false otherwise. The concept to be learned in scheduling, for example, can be the identification of the most appropriate dispatching rule (a class) for a given set of manufacturing attributes.

A set of training examples is provided as input for learning the concept representing each class. A training example consists of a vector of attribute values and the corresponding

class. A learned concept can be described by a rule which is determined by the inductiv learning process. If a new data case satisfies the conditions of this rule, then it belongs t the corresponding class. For example, a rule defining a concept can be the following:

IF $(b_{i1} \geq a_{i1} \geq c_{i1})$ AND ... $(b_{im} \geq a_{im} \geq c_{im})$

THEN τ.

where a_{ij} represents the jth. attribute, b_{ij} and c_{ij} define the range for a_{ij}, and τ denotes th class.

Shaw (1989b) and Park et al. (1989) employ inductive learning to derive heuristic for selecting the appropriate dispatching rules in a flexible manufacturing system. In thi instance, the IF-THEN rule is treated as a *selection heuristic* which is a conjunction o attribute conditions collectively defining the *pattern*, and τ represents the best schedulin rule for that pattern.

An instance that satisfies the definition of a given concept is called a *positive* example of that concept; an instance which does not do so is a *negative* example. In the dynamic scheduling problem, because there are several scheduling rules which can potentially be selected, multiple concepts need to be learned. In this situation, the training examples supporting the use of a given scheduling rule are treated as the positive examples of that rule; training examples supporting any other rule would be treated as negative examples.

Generalization and specialization are essential steps for the inductive learning process. A generalization of an example is a concept definition which describes a set containing that example. In other words, if a concept description Q is more general that the concept description P, then the transformation from P to Q is called generalization; a transformation from Q to P would be specialization. For a set of training examples, the generalization process identifies the common features of these examples and formulates a concept definition describing these features; the specialization process, on the other hand, helps restrict the coverage of features for a concept description. Thus, inductive learning can be viewed as the process of making successive iterations of generalizations and specializations on concept descriptions as observed from examples. This process would continue until an inductive concept description which is consistent with all the training examples is found. Thus the

generalization/specialization relations between concept descriptions provide the basic structure to guide the search the inductive learning process. For a given problem, applying the inductive learning process can contribute to one's understanding of the decision process on the following three dimensions (Shaw and Gentry 1990):

- *Predictive validity:* the ability to predict the decision outcome for a given data base.
- *Structural validity:* the ability to capture the underlying structure of the decision process.
- *Identifying validity:* the ability to infer the most critical attributes in the decision process.

These features of inductive learning make it useful in dealing with the scheduling problem. If we can make an inductive learning system *observe* the effects of various scheduling decisions on the manufacturing processes and the resulting scheduling performance, then it can 1) predict the outcome of any schedule for a given manufacturing process in a specified set of manufacturing conditions (predictive validity), 2) capture the underlying decision structure of the scheduling process (structural validity), and 3) identify the critical manufacturing attributes for the scheduling decision process (identifying validity).

The input to an inductive learning algorithm consists of three steps: 1) A set of positive and negative examples, 2) a set of generalization and other transformation rules, and 3) criteria for successful inference. Each training example consists of two components — a data case consisting of a set of attributes, each with an assigned value; and the classification decision made by a domain expert according to the given data case. The output generated by this inductive learning algorithm is a set of decision rules consisting of inductive concept definition for each of the classes. Learning programs falling into this category include AQ15 (Michalski 1983), PLS (Rendell 1983) and ID3 (Quinlan 1986). These programs are referred to as similarity- based learning methods.

Shaw et al. (1990) compare the above three inductive learning programs in terms of their algorithmic designs and classification accuracy. They find that, in general, ID3 and PLS produce more accurate classifications than AQ15. They are also more efficient computationally and are better able to handle noisy data. For these reasons, we use ID3 in this research. The

learning process in ID3 follows a sequence of specialization steps guided by an *information entropy function* for evaluating class membership. The concept description generated by a learning process can be represented by a decision tree, which is a special case of disjunctive normal form expressions.

4 Heuristic Scheduling and Inductive Learning

4.1 Heuristic Scheduling

One of the major applications of machine learning to scheduling is in dynamic job shops and flexible manufacturing systems (FMSs) in which jobs arrive randomly over time and the system behaves like a network of queues. Because of the combinatorial nature of the underlying optimization problem, scheduling decisions in such systems are specified in terms of *dispatching* rules; whenever a machine becomes idle, the scheduler must decide which job should next be processed on the machine. This selection is based on assigning priority indices to various jobs competing for the given machine; the job with the highest priority is selected next. Dispatching rules differ in how they assign these priority indices.

Because it is difficult to evaluate most of these dispatching rules analytically, computer simulation methods are used generally to study their behavior and compare their relative performance. Prior research in this area (see, for example, Conway et al. 1967, and Baker 1974, 1984 for a survey of this research) deals primarily with conventional job shops for the scheduling objectives of minimizing mean job flow time and mean job tardiness. The conclusions reached by the various studies are, however, at variance with one another.

Baker (1984) suggests that the fundamental reason underlying these conflicting results is the fact that they address different systems, and the relative performance of a given dispatching rule depends upon the system and job characteristics. In the context of a job shop under balanced machine workloads, Baker studies the impact of one such attribute, namely, the tightness of job due dates. He shows that, depending upon due date tightness, there are crossovers between dispatching rules. In particular, at the extremes, EDD is superior when due dates are set loosely, and SPT performs well when they are tightly set.

MOD performs the best in the intermediate range (which is, nevertheless, quite wide in his study).

Raman et al. (1989) extend Baker's investigation to also understand the impact of imbalance of machine workloads (which leads to one or more bottlenecks in the system) as well as variability in due date assignment on the performance of dispatching rule. Their study shows that while MOD retains its effectiveness under balanced workloads, there are crossovers between MOD and MDD when significant imbalance in machine workloads exists. In particular, MDD is superior when due date tightness is low to moderately high, and when there is greater variability in the due date assignment. They also study the benefits of using an adaptive scheduling procedure in which the selection of the dispatching rule is machine workload dependent. In particular, the shop performance improves significantly if the dispatching rule used at the bottleneck machine(s) is MDD, and MOD is used at other machines.

4.2 Induction of Heuristic Knowledge

The aforementioned studies on heuristic scheduling raise two important questions: 1) What are the job and system attributes that affect the relative performance of a given dispatching rule, and 2) How should the scheduler take these into account while making the dynamic scheduling decisions? Inductive learning can help resolve these issues by first identifying the relevant system and job attributes, and subsequently, developing scheduling systems capable of selecting the dispatching rule most appropriate for a given state of the manufacturing system as characterized by the combination of these attributes (i. e., the *manufacturing patterns*). Because the selection of dispatching rules is determined by the dynamically changing manufacturing patterns, we refer to this scheduling approach as **Pattern-Directed Scheduling (PDS)**.

The suggested approach consists of two basic stages: 1) The learning stage, and 2) the scheduling stage. The **learning stage** extracts relevant manufacturing patterns, which are conjunctions of attribute-value pairs, from simulation experiments. The output of this stage is a set of heuristic rules which describe the relationships among manufacturing patterns and the appropriate dispatching rule in the form

pattern (i,j) $\rightarrow$ *dispatching rule i*

An example of such a heuristic is

IF: $(TBF < 14)$ AND $(S \geq 63)$ AND $(NSDRL \geq 16)$

THEN: MDD rule.

This rule states that when the Total buffer size is less than 14, the system utilization is greater than or equal to 0.63, and the normalized standard deviation of relative machine loading is greater than or equal to 1.6, then the dispatching rule to be applied is the Modified Job Due Date rule. Index j is used in the above expression to denote the various patterns for which rule i would be applicable.

The **pattern-directed scheduling** stage applies the dispatching rules selected in stage 1 adaptively for performing dynamic scheduling based on the manufacturing pattern manifested by the system.

The learning stage starts with conducting a series of simulation experiments to generate training examples to study the performance of various dispatching rules under a variety of manufacturing environments which are modeled in the simulation program. These training examples are input into the inductive learning process, which generates the heuristic rules describing the dependence between manufacturing patterns and the dispatching rules. This integration of inductive learning and pattern directed scheduling is depicted in Figure 1.

INSERT FIGURE 1 HERE

Figure 1 also shows an additional stage, that of **rule refinement.** This stage collects the performance results of the pattern-directed stage, and uses a critic program to evaluate the performance of a given rule. If the heuristic rule is found to yield unfavorable results, then the rule-refinement module generates additional training examples with a view to revise the rule (possibly through attribute conjunction). Thus, the rule- refinement program iterates through the steps of obtaining performance results of various rules, analyzing rules which are not satisfactory, and appropriately modifying these rules. [Politakis and Weiss (1984),

Bundy and Silver (1982) and Wilkins (1989) describe some methods for refining rules.] Meta-rules are used to suggest further experimentation involving training examples. The 'IF' part of the meta-rule contains a conjunction of predicate clauses that look for certain features of the unsatisfactory rules as well as the training examples which generated these rules. The 'THEN' part of the meta-rule suggests further experiments with modified attributes for generating additional training examples and refining the rules.

The proposed pattern-directed scheduling approach can be viewed as an AI planning process (Georgeff and Lansky 1986), in which a sequence of operators, i.e., the dispatching rules, are selected on the basis of the current manufacturing state. In terms of the problem solving strategy, the selection of rules in pattern directed scheduling is driven by the changes in manufacturing pattern, rather than the goal. Moreover, rule selection is *opportunistic* based on the dynamics of the manufacturing process. It is not preplanned as is the case for most planning problems.

We are now in a position to formally state the difference in the approach taken by this paper relative to the previous work on dynamic scheduling which has primarily adopted the use of a single dispatching rule.

Definition 1 *The single rule scheduling problem is represented by* $(J, S_0, \mathcal{H}, D)$ *where* J *denotes the scheduling objective,* S_0 *is the initial state of the manufacturing system,* $\mathcal{H}$ *represents the set of candidate dispatching rules, and* D *denotes the decision that selects a dispatching rule from* $\mathcal{H}$ *based on* S_0*. This rule is used until* J *is achieved.*

Definition 2 *The pattern directed scheduling problem is represented by* $(J, \mathcal{M}, \mathcal{H}, \mathcal{R}, E)$ *where* J *and* $\mathcal{H}$ *are defined as above.* $\mathcal{M}$ *is the set of manufacturing states, and* $\mathcal{R}$ *is the set of heuristics for selecting dispatching rules. Each rule* r *in* $\mathcal{R}$ *is of the form* $p \rightarrow h, p \in \mathcal{M}, h \in \mathcal{H}$*.* E *is a list of events,* $e_{t1}, e_{t2}, \ldots, e_{tk}, \ldots$*, which trigger the application of selection heuristics at time* $t_1, t_2, \ldots$*. At time* t_{tk}*, dispatching rule* h *selected by* $\mathcal{R}$ *is activated, and the manufacturing system transits from state* m_{tk} *to state* m_{tk+1}*, i. e.,* $h(m_{tk}) = m_{tk+1}$*.*

In our approach, inductive learning is used to determine the set R from simulation experiments. Because the pattern-directed approach is more adaptive to the environmental changes

of the manufacturing system, it should result in better scheduling performance. As discussed later in §6, this conjecture is verified by empirical studies.

The fact that the pattern-directed approach can select dispatching rules dynamically implies that it should be especially useful in manufacturing systems with more dynamic processes. This feature makes the pattern-directed approach a good candidate for scheduling flexible manufacturing systems which are characterized by the versatility of their machines and routing flexibility, and therefore, require a more dynamic approach to scheduling.

5 FMS Scheduling

5.1 Problem Characteristics

FMS scheduling differs from conventional job shop scheduling in that FMSs offer many more alternatives to the scheduler. For example, an operation could be processed at any one of several machines. In addition, the various machines in the system are linked more tightly because of limited available buffer space and the use of common material transporters. This results in a more difficult scheduling problem as it leads to more manufacturing patterns that can be manifest in an FMS. Consequently, the ability of PDS to discern these patterns and to apply the appropriate dispatching rules becomes even more appealing in an FMS environment.

In addition to due date tightness and relative workload imbalance, which have been discussed in the previous section, the attributes which are likely to be significant for selecting the appropriate dispatching rule in an FMS include: 1) The scope for alternative job routing, 2) the limitation on local buffers available at individual machines, and 3) the increased versatility of machines. It is possible in an FMS to configure the machines in such a way that a given operation can be done on one or more machines. Elementary queueing theory suggests that the *routing flexibility* provided by such parallel servers can lead to significant reductions in the job flow time, which in turn, should reduce job tardiness. However, it is not clear how this flexibility will impact the relative effectiveness of various dispatching rules. The system performance will also be affected by the constraints on the available

buffer space. When this constraint is binding, various machines in the system are likely to go through phases of blocking and starving which adversely affect the system performance. In such a case, the overall performance of a scheduling rule also depends upon its ability to minimize such instances. Finally, the increased processing capability of some machines in the system would lead to varying number of operations being processed at different machines. This would affect, among other factors, the coefficient of variation of the machine service times which would, in turn, impact the flow times and tardiness values.

5.2 Implementation of PDS

We now discuss the application of PDS in an FMS which permits random, job shop like material flows. The scheduling objective considered is minimizing mean tardiness. This objective was selected primarily because it has been studied extensively. As mentioned in §3, a variety of dispatching rules have been found to be effective under different situations. This not only provides with a larger set $\mathcal{H}$ of effective dispatching rules, but more importantly, we have a strong benchmark for testing the relative effectiveness of the PDS approach. The set of dispatching rules which have been found to be dominant in previous research includes the Earliest Due Date (EDD) rule [Baker and Bertrand 1981, 1982], the Shortest Processing Time (SPT) rule [Baker and Bertrand 1981, 1982; Conway 1965], the Modified Job Due Date (MDD) rule [Baker and Kanet 1983; Raman et al. 1989] and the Modified Operation Due date (MOD) rule [Baker 1984; Raman et al. 1989]. Consequently, these four dispatching rules were selected in this study.

The control attributes selected for capturing the relevant manufacturing patterns were:

1. Number of machines in the system (NMAC).

2. Total buffer size (TBF).

3. Maximum relative machine workload (BOTTLE) which checks whether any one of the machines in the system is a bottleneck at a given point in time. It is expressed as the ratio $W_{max}/\overline{W}$, where W_{max} is the maximum workload, measured in terms of remaining processing time, in front of any machine in the system currently, and $\overline{W}$ is the current average machine workload.

4. Variability in machine workload (NSDRL). This is expressed as the ratio of the standard deviation of individual machine utilizations to the average machine utilization.

5. Contention factor (CFACT) which indicates the average number of alternative machines available for processing a given operation.

6. Contention factor ratio (CFRATIO) which is the ratio of CFACT to NMAC.

7. Machine homogeneity (MH) which measures the variability in the number of operations that individual machines can process. It is expressed as the ratio of the standard deviation of the number of operations that each machine can process to the average number of operations that a machine can process.

8. Flow allowance factor (F) which measures due date tightness. Following Baker (1984), we used the Total Work Content (TWK) rule for assigning job due dates. Under TWK, the due date d_j of job j is determined as follows:

$$d_j = a_j + Fp_j$$

where a_j is the arrival time and p_j is the total processing time of job j.

9. Overall system utilization (S).

Note that NMAC, in conjunction with MH, is a measure of the versatility of the machines in the system. BOTTLE and NSDRL together describe the relative machine workloads. S impacts job flow times directly. As Baker (1984) notes, the due date tightness induced by a given value of F depends upon system utilization as well. However, when significant imbalances in machine workloads exist, S merits independent consideration.

In the learning stage, we generated 130 training examples which covered various combinations of the nine attributes discussed above. For each example, steady state statistics pertaining to the mean tardiness values under each of the four dispatching rules were recorded. Those instances in which a given rule performed the best were selected as positive examples pertaining to that rule. The ID3 algorithm was used for rule induction based on these positive examples. The resulting decision tree was translated into a set of pattern directed heuristics which is depicted in Figure 2.

INSERT FIGURE 2 HERE

Note that an important contribution of this decision tree is in its ability to highlight the relative importance of the control attributes in influencing the selection of dispatching rules. Thus, from Figure 2, these attributes can be ranked in the following order of decreasing importance: TBF, (S, BOTTLE), (F, NSDRL), CFACT, and CFRATIO. Also note that MH is screened out because of its marginal impact.

Preliminary studies with PDS indicated the need to curtail excessive nervousness of the scheduling system. We observed that PDS performs poorly in many instances if switching between dispatching rules was affected immediately upon a pattern change. In order to mitigate overreaction to patterns which are only transitory, we developed a smoothing approach in which the scheduling mechanism retains a cumulative score of the number of occasions a given dispatching rule is favored. Whenever a scheduling decision is to be made, the dispatching rule with the maximum cumulative score is selected provided it is above a prespecified threshold. Suppose that the dispatching rule being used currently is i with a cumulative score of S_i. Whenever a scheduling decision is required, this scheme will select rule j, if it exists, where

$$j = arg \max_{h \in \mathcal{H}} (S_h), \text{ and } S_j \geq \theta S_i,$$

and θ is a measure of the required threshold; otherwise it will continue using rule i. θ is a smoothing coefficient; higher θ values lead to increased damping of system responsiveness. Further experimentation revealed that varying degrees of smoothing are required to respond to different manufacturing patterns. This resulted in a scheme in which θ was allowed to vary, and the rule induction process was applied once again to yield another decision tree; the corresponding set of heuristic selection rules is shown in Figure 3.

INSERT FIGURE 3 HERE

The overall PDS module, therefore, consists of two functional components: the heuristic selection module (HSM) and the pattern smoothing module (PSM). These two modules and their interactions are depicted in Figure 4. Whenever a scheduling decision is required, PSM selects the θ value based upon the current manufacturing pattern observed and the

induction tree for θ. This value of θ determines the required smoothing threshold. HSM uses this threshold in conjunction with the observed pattern and the heuristic knowledge generated through inductive learning to select the appropriate dispatching rule to use.

INSERT FIGURE 4 HERE

We now discuss the experiments conducted to evaluate the relative merit of the PDS approach.

6 Experimental Study

This section describes an experimental study conducted to understand the performance characteristics of PDS. *A priori*, we expect the PDS approach to be superior to the conventional single dispatching rule approach for two reasons. First, as depicted in Figure 2, PDS identifies the best dispatching rule for a given manufacturing scenario. Because of this *selecting ability*, PDS should perform at least as well as the best from among the candidate dispatching rules considered in $\mathcal{H}$. Second, PDS is able to adapt its selection of the best dispatching rule dynamically to the changing patterns. Such an ***adapting ability*** should result in a schedule quality which is even superior to that of the best dispatching rule. Simulation studies were designed specifically to verify these two expectations.

As noted earlier, the PDS module consists of the HMS and PSM components. This expermiental study focused on the performance of HSM in selecting the best dispatching rule through inductive learning. Thus, the selection function performed by PSM is emulated by employing the value of θ which is the best of three predetermined values - 0.0, 0.7 and 1.0, so that the impact of HSM can be isolated and better understood. [The smoothing effect of PSM is, however, indispensable for achieving the adapting ability of PDS. As an extension to this work, we are currently conducting another study which focuses on the adapting behavior of PDS in response to such disturbances as machine breakdowns and sudden changes of job loading patterns.]

The simulation experiments addressed an FMS at which jobs arrive following a Poisson process. Upon its arrival, each job was assigned the number of required operations randomly

from a uniform distribution which ranged between 1 and the total number of machines in the system. Operation processing times were sampled from an exponential distribution with a mean of 400. A range of due date tightness was achieved by allowing the flow allowance factor to vary between 2 and 15.

Three different system sizes comprising 4, 6 and 8 machines were considered. The relative machine workload was allowed to vary between 0.6 and 1.2. Machine homogeneity was allowed to vary between 0 and 0.4. [A value of 0 implies that the same number of operations are assigned to all machines.] The contention factor ranged between 2 and 4. Job interarrival times were varied to yield overall system utilizations between 50% and 95%.

The experiments addressed 69 different combinations of these parameters. In each case, the method of batching was used to determine the steady state mean tardiness values resulting from employing SPT, EDD, MOD and MDD dispatching rules individually, as well as the values obtained by using PDS which incorporated the selection heuristics depicted in Figures 2 and 3.

The mean tardiness values obtained by using different scheduling rules are shown in Table 1. In this table, BEST refers to the mean tardiness value obtained by the dispatching rule which performs the best for the corresponding scenario. This dispatching rule is labeled the *Best Rule*.

Overall, PDS resulted in an improvement of 11.5% over BEST. It produced lower mean tardiness values in 33 cases and the same values in 26 cases. It was worse in the remaining 10 cases; however, the tardiness values obtained in 8 of these cases were quite low (less than 19) under both BEST and PDS.

INSERT TABLE 1 HERE

These experiments indicate that the performance of PDS depends upon two factors. First is the impact of the number of machines in the system. Table 2 gives a breakdown of the number of instances in which PDS was better, the same and worse for the three system sizes studied. PDS is distinctly superior for 4-machine systems. However, as the system size increases, the number of instances in which PDS and BEST are equally effective increases as well.

TABLE 2

Impact of System Size

No. of Machines	*No. of Cases in which PDS is*		
	Better	Same	Worse
4	18	2	4
6	10	8	3
8	5	16	3

Second, as depicted in Figure 5, the relative improvement achieved by PDS depends upon the frequency of pattern changes realized in the system. Recall that a *pattern change* signals a potential switch in the applicable dispatching rule as determined by the PDS decision tree. However, depending upon the appropriate value of θ, a change may not be affected in practice. Therefore, the number of actual dispatching *rule switches* would be smaller than the number of pattern changes. [Also note that the number of pattern changes is dependent upon the PDS tree generated. If this tree is different, possibly because of the use of a different rule induction algorithm, the number of pattern changes for the same experiment is also likely to be different.]

INSERT FIGURE 5 HERE

In order to understand these results, it is important to first note that the relative performance of PDS will depend, among other factors, upon i) the difference in the performance of individual dispatching rules under a given pattern, and ii) whether the system characteristics result in a *dominant* pattern. If the system attributes are such that they yield a sequence of patterns in which the various dispatching rules yield similar results, PDS is not likely to be significantly superior to BEST. This argument partially explains the apparent dependence of PDS upon system size. An increase in the number of machines, in general, leads to an increase in contention factor as well. As reported in Wayson's (1965) and Stecke and Raman's (1990) studies, the difference in mean flow times and mean tardiness values obtained under different dispatching rules decreases with an increase in the number of alternative machines available for processing any operation. Consequently, higher contention factors lead

to smaller differences between PDS and BEST. Therefore, conceptually, we would expect PDS and BEST to behave identically in the limit.

However, in most real FMSs, providing high contention factors, which amounts to creating more redundancies in the system, may not be economically viable because it leads to a reduction in the number of parts which can be manufactured concurrently and/or a reduction in system utilization. In most practical situations, contention factor is likely to be low to moderately high, a range in which PDS is significantly superior to BEST.

We observed from our experiments that the relative difference between various dispatching rules decreases also when mean tardiness values are small. [This result is reported in earlier simulation studies of job shops as well; see, for example, Baker 1984.] In such cases, the benefit in using PDS is small *even if there are large number of rule switches.* [As shown in Figure 5, the experiments *do* show that small tardiness values are accompanied by a large number of pattern changes. This would indicate the existence of several crossovers among the dispatching rules when tardiness is small even though they differ only marginally from each other.] This argument partially explains why, in Figure 5, we find that the improvement ratio decreases with a decrease in mean tardiness and an increase in the number of pattern changes.

At the other extreme, fewer pattern changes imply that there is a small number of dominant patterns that characterize the system. In such cases, PDS would frequently select the dispatching rule(s) most appropriate for the dominant pattern(s); as a result, the difference between PDS and BEST would be marginal. As shown in Figure 5, we observed that dominant patterns are accompanied by high tardiness values. This would happen, for example, if system utilization levels are high and buffer sizes are small. In such instances, MDD would be the most appropriate dispatching rule across a wide range of other system attributes. Consequently, the manufacturing pattern which favors MDD will be dominant resulting in a small difference between PDS and MDD (which would be the BEST rule). Of course, it is unlikely that most real systems would operate at high tardiness values on a sustained basis.

In summary, the maximum effectiveness of PDS is realized when the number of pattern changes is medium to reasonably high, and no one pattern is clearly dominant. This range also corresponds to tardiness values ranging from low to moderately high — a range which

would be applicable to most viable manufacturing systems.

Finally, we note three possible reasons why PDS results in higher tardiness values compared to BEST in some instances. First, because the set of training examples used for generating PDS is a subset of the universe of the possible scheduling environments, the learned heuristics are likely to be *overgeneralized* to some extent leading to a few prediction errors. Specifically, PDS may not perform well in a situation which is not explicitly addressed by the training examples. Second, the performance of PDS is limited by the number of control attributes considered for designing the training examples. While, on the basis of the available scheduling literature, the nine attributes considered in this study are fairly comprehensive, it is possible that for a given system, the selection of appropriate dispatching rules is affected significantly by some other attributes. Third, because the training examples are driven by simulation experiments, the appropriateness of a dispatching rule for a given pattern is determined by its *steady state average* performance over the length of the simulation run. Its implementation during real time scheduling is, however, based on the pattern which is observed at the instant a scheduling decision is to be made. While a dispatching rule may perform well in the long run for a given set of attributes, it need not necessarily be effective when it is applied on a rolling basis on transient patterns.

The adverse impact of overgeneralization can be mitigated in practice by employing a feedback mechanism to monitor scenarios in which PDS does not perform well, and to update the set of training examples accordingly. The suggested rule induction process can then be used to appropriately refine the selection heuristics. In a similar vein, if empirical evidence suggests that one or more control attributes, which were not previously considered for generating training examples, have significant impact on the selection of dispatching rules, then they can be included in new training examples (and, if necessary, selection heuristics can be modified). In a real system, therefore, the relative performance of PDS should continuously improve with the incorporation of a feedback mechanism.

However, the use of steady state average values for determining the appropriateness of any dispatching rule is intrinsic to the overall PDS approach in the context of dynamic scheduling. The adverse impact, if any, of doing so is partially mitigated by the use of θ which helps in smoothing out the transient patterns. Nevertheless, it must be understood

that PDS may not perform very well if the pattern changes are extremely frequent. This is another factor which contributes to a reduction in improvement ratio as the number of pattern changes increases. As the experimental results indicate, however, in most such instances there is only a minor degradation in the quality of the PDS schedule.

7 Conclusion

This study develops a scheduling approach which employs inductive learning to generate pattern-directed heuristics for making dynamic scheduling decisions in an FMS. This approach comprises the three steps of: 1) Generation of training examples through experimentation involving various dispatching rules, 2) determination of selection heuristics through inductive learning, and 3) executing pattern-directed scheduling adaptively.

The PDS approach performs better than the conventional single dispatching rule approach because of its capabilities of selecting the best rule (the selecting ability), and of switching between different rules in real time with changes in the state of the system (the adapting ability). The decision trees generated in this study to depict the selection heuristics clearly show the efficacy of inductive learning in extracting and ranking the system attributes relevant for deciding upon the appropriate dispatching rule to employ. In this process, those attributes which have only a marginal impact are screened out. The experimental results show that this approach results in significantly improved system performance. This is especially so in systems which exhibit dynamically changing patterns.

We believe that the most appealing characteristic of PDS, in the presence of a feedback mechanism, is its ability to dynamically refine the set of selection heuristics in response to manufacturing scenarios in which it does not preform very well. Consequently, in a real system, the relative performance of PDS should improve continuously.

References

1. Baker, K. R. (1974), *Introduction to Sequencing and Scheduling*, John Wiley, New York.

2. Baker, K. R. (1984), "Sequencing Rules and Due-Date Assignments in a Job Shop," *Management Science*, Vol. 30, No. 9, 1093–1104.

3. Booker, E. (1989), "Outlook: Manufacturing and Design," *Computerworld*, Vol. 24, No. 1.

4. Bruno, G., A. Elia, and P. Laface (1986), "A Rule-based System to Schedule Production," *Computer*, Vol. C-19, No. 7, 32–40.

5. Bundy, A., and B. Silver (1982), "A Critical Survey of Rule Learning Programs," in *Proceedings of the European Conference on Artificial Intelligence*, Orsay, France.

6. Conway, R. W., W. L. Maxwell and L. W. Miller (1967), *Theory of Scheduling*, Addison-Wesley, Reading, MA.

7. Fikes, R. E., and N. J. Nilsson (1971), "STRIPS: A new Approach to the application of Theorem Providing to Problem Solving," *Artificial Intelligence*, Vol. 2, 189–208.

8. Fox, M. S. (1987), *Constraint-Directed Search: A Case Study of Job-Shop Scheduling*, Morgan Kaufmann, Los Altos, CA.

9. Fox, M. S., and S. F. Smith (1984), "ISIS: A Knowledge- Based System for Factory Scheduling," *Expert Systems*, Vol. 1, No. 1, 25–49.

10. Georgeff, M. P. and Lansky, A. L. (1986), *Reasoning about Actions and Plans*, Morgan Kaufmann, Los Altos, CA.

11. Kanet, J. J., and H. H. Adelsberger (1987), "Expert Systems in Production Scheduling," *European Journal of Operational Research*, Vol. 29, 51–57.

12. Kusiak, A. (1987), "Designing Expert Systems for Scheduling Automated Manufacturing," *Industrial Engineering*, 42–46.

13. Kusiak, A., and M. Chen (1988), "Expert Systems for Planning and Scheduling Manufacturing Systems, " *European Journal of Operational Research*, Vol. 34, No. 2, 113–130.

14. Michalski, R. S. (1983), "A Theory and Methodology of Inductive Learning," in *Machine Learning*, edited by R. Michalski, J. Carbonell and T. Mitchell, Tioga, Palo Alto, CA.

15. Ow, P. S. (1984), "Heuristic Knowledge and Search for Scheduling," *Unpublished Ph.D. Dissertation*, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA.

16. Ow. P. S., S. F. Smith, and R. Howie (1988), "A Cooperative Scheduling System," in *Expert Systems and Intelligent Manufacturing*, edited by M. D. Oliff, North-Holland, 70–89.

17. Park, S. C., N. Raman, and M. J. Shaw (1989), "Heuristic Learning in Pattern-Directed Scheduling," in *Proceedings of the Third ORSA/TIMS Conference on Flexible Manufacturing Systems*, Elsevier Science, Amsterdam, The Netherlands.

18. Parunak, H. (1987), "Manufacturing Experience with Contract Net," in *Distributed Artificial Intelligence*, edited by M. Hughes, Pitman, London, U. K.

19. Politakis, P., and S. Weiss (1984), "Using Empirical Analysis to Refine System Knowledge Bases," *Artificial Intelligence*, Vol. 22, 23–48.

20. Quinlan, J. R. (1986), "Induction of Decision Trees," *Machine Learning*, Vol. 1, 81–106.

21. Raghavan, V. (1988), "An Expert System Framework for the Management of Due-Dates in Flexible Manufacturing Systems," in *Expert Systems and Intelligent Manufacturing*, edited by M. D. Oliff, North-Holland, 235–247.

22. Raman, N. and F. B. Talbot (1985), "A Survey of Literature on Production Scheduling as It Pertains to Flexible Manufacturing Systems," Working Paper # NBS-GCR-85-499, National Bureau of Standards, Gaithersburg, MD.

23. Raman, N., F. B. Talbot, and R. V. Rachamadugu (1989), "Due Date Based Scheduling in a General Flexible Manufacturing System," *Journal of Operations Management*, Vol. 8, No. 2, 115–132.

24. Rendell, L. (1983), "A New Basis for State-Space Learning Systems and a Successful Implementation," *Artificial Intelligence*, Vol. 1, No. 2, 177–226.

25. Rodammer, F. A. and K. P. White (1988), "A Recent Survey of Production Scheduling," *IEEE Transactions on Man, Machine and Cybernetics*, Vol. 18, No. 6, 841–851.

26. Shaw, M. J. (1986a), "A Pattern-Directed Approach to FMS Planning and Scheduling," in *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems*, edited by K. E. Stecke and R. Suri, Elsevier Science, Amsterdam, The Netherlands.

27. Shaw, M. J. (1986b), "A Two-Level Planning and Scheduling Approach to Computer Integrated Manufacturing," in *Proceedings of the Symposium on Real-Time Optimization in Automated Manufacturing Facilities*, National Bureau of Standards, Gaithersburg, MD.

28. Shaw, M. J. (1987), "Inductive Learning for Enhancing Knowledge-Based Expert Systems," *Decision Support Systems*, Vol. 3, 319–332.

29. Shaw, M. J. (1988a), "A Knowledge-Based Scheduling System for Flexible Manufacturing," *International Journal of Production Research*, Vol. 26, No. 5, 821–844.

30. Shaw, M. J. (1988b), "Dynamic Scheduling in Cellular Manufacturing Systems: A Framework for Networked Decision Making," *Journal of Manufacturing Systems*, Vol. 7, No. 2, 83–94.

31. Shaw, M. J. (1988c), "FMS Scheduling as Cooperative Problem Solving," *Annals of Operations Research*, Vol. 17, 323–346.

32. Shaw, M. J. (1989), "A Pattern-Directed Inference Approach to FMS Scheduling," *International Journal of Flexible Manufacturing Systems*, Vol. 2, No. 2, 121–144.

33. Shaw, M. J., and J. Gentry (1990), "Inductive Learning for Risk Analysis," *IEEE Experts*, Vol. 5, No. 1, 47–53.

34. Shaw, M. J., J. Gentry, and S. Piramuthu (1990), "Inductive Leaning Systems for Decision Support: A Comparative Study," *Computer Science in Economics and Management*, forthcoming.

35. Shaw, M. J., U. Menon, and S. C. Park (1988), "Machine Learning Methods for Computer-aided Process Planning," in *Expert Systems and Manufacturing Designs*, edited by A. Kusiak, Society of Manufacturing Engineers Press, Dearborn, MI.

36. Shaw, M. J., and A. Whinston (1985), "Task Bidding, Distributed Planning, and Flexible Manufacturing," in *Proceedings of the IEEE Conference on Artificial Intelligence Applications*, Miami, FL.

37. Shaw, M. J., and A. Whinston (1989a), "An Artificial Intelligence Approach to Scheduling in Flexible Manufacturing Systems," *IIE Transactions*, Vol. 21, No. 2, 170–183.

38. Shaw, M. J., and A. Whinston (1989b), "Learning and Adaptation in a Distributed Artificial Intelligence System," in *Distributed Artificial Intelligence*, Vol. II, edited by M. Huhns, Morgan Kaufmann, Los Altos, CA.

39. Stecke, K. E. and N. Raman (1990), "Pre-Production Planning Decisions in Flexible Manufacturing Systems with Random Material Flows," *in preparation.*

40. Stokey, R. J. (1989), "AI Factory Scheduling: Multiple Problem Formulations," *SIGART Newsletter*, No. 110, 27–30.

41. Wayson, R. D. (1965), "The Effects of Alternate Machines on Two Priority Dispatching Disciplines in General Job Shops," *Master's Thesis*, Cornell University, Ithaca, NY.

42. Wysk, R. A., S. D. Wu, and N. Yang (1986), "A Multi-pass Expert Control System for Manufacturing Systems," in *Proceedings of the Symposium on Real-Time Optimization in Automated Manufacturing Facilities*, National Bureau of Standards, Gaithersburg, MD.

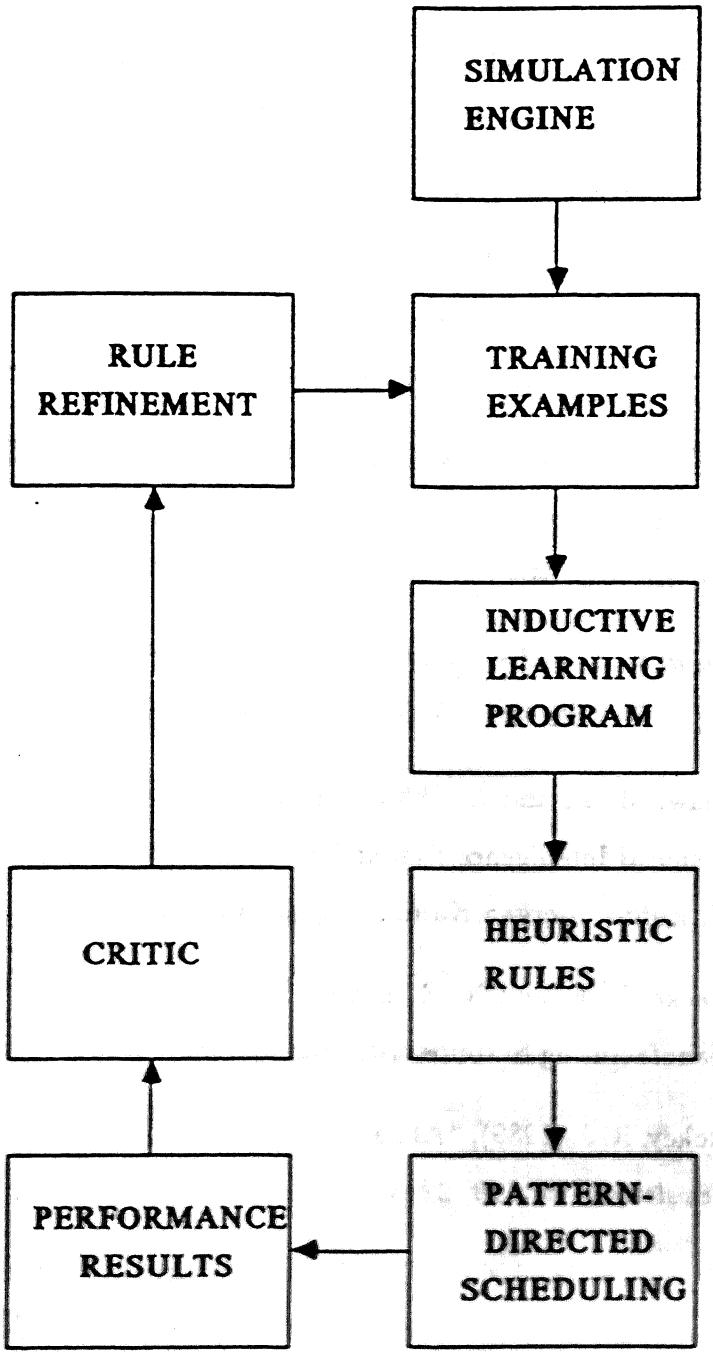

Figure 1: The Inductive Learning Process

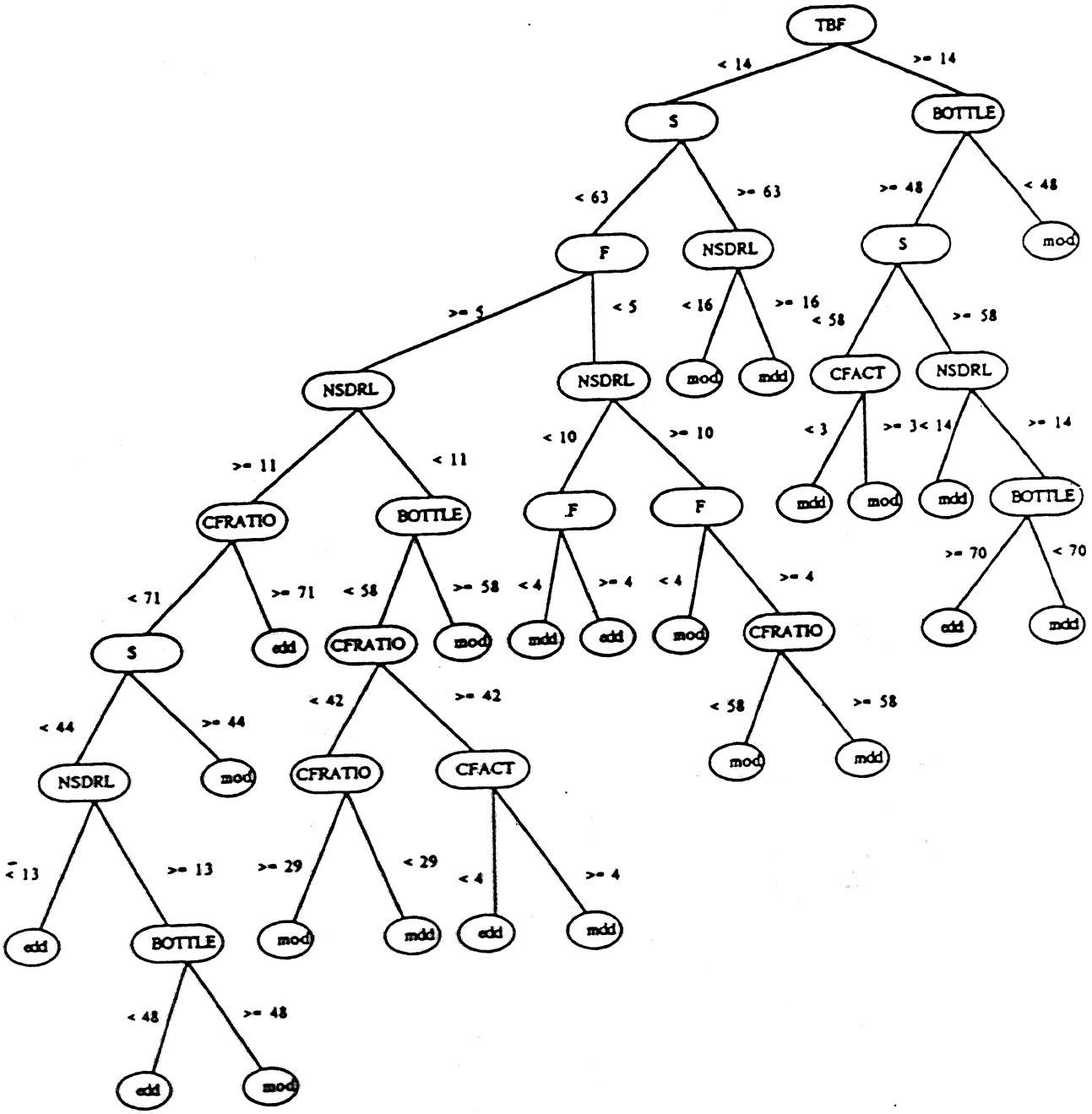

Figure 2: Decision Tree for Selecting Dispatching Rules

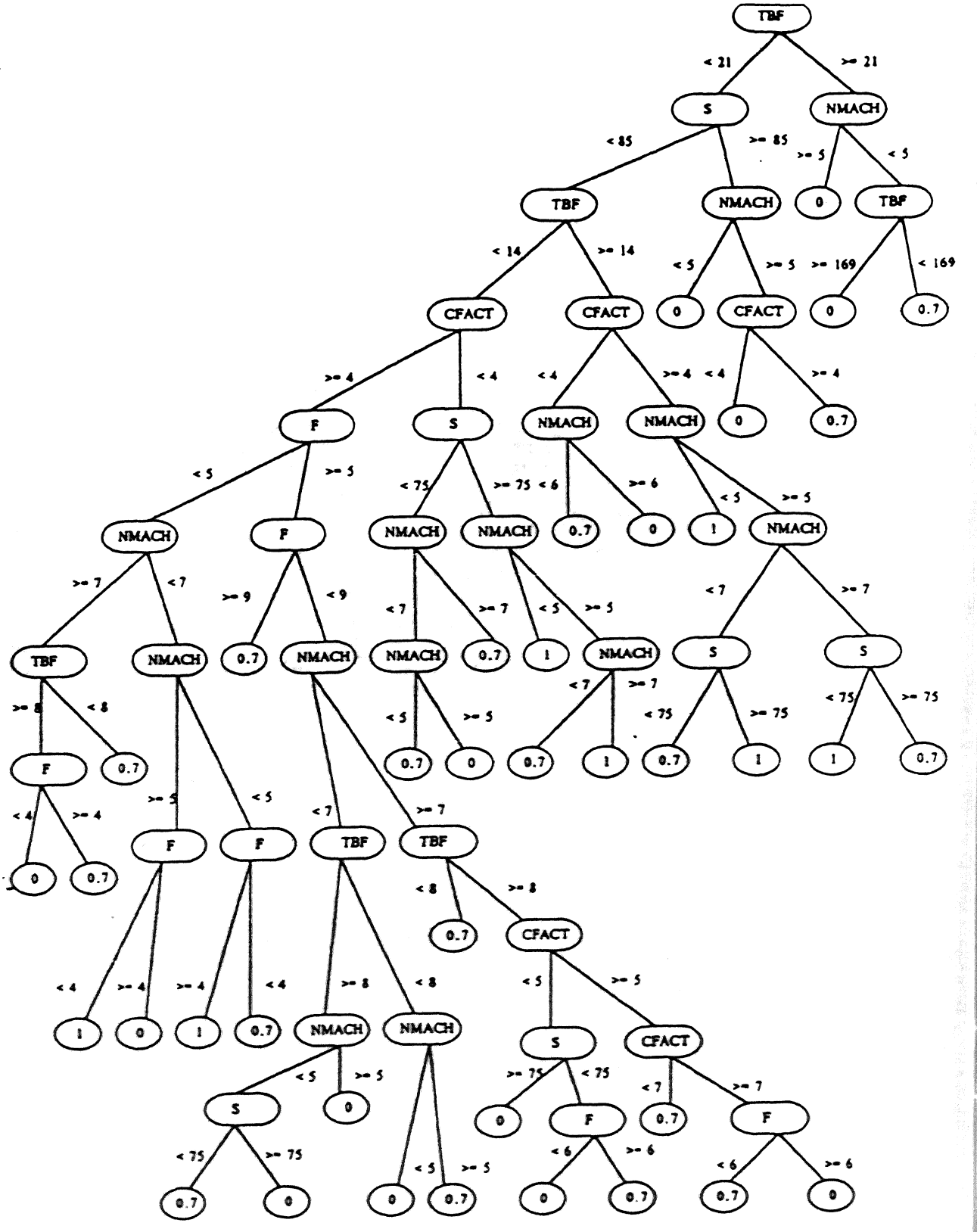

Figure 3: Decision Tree for Selecting θ

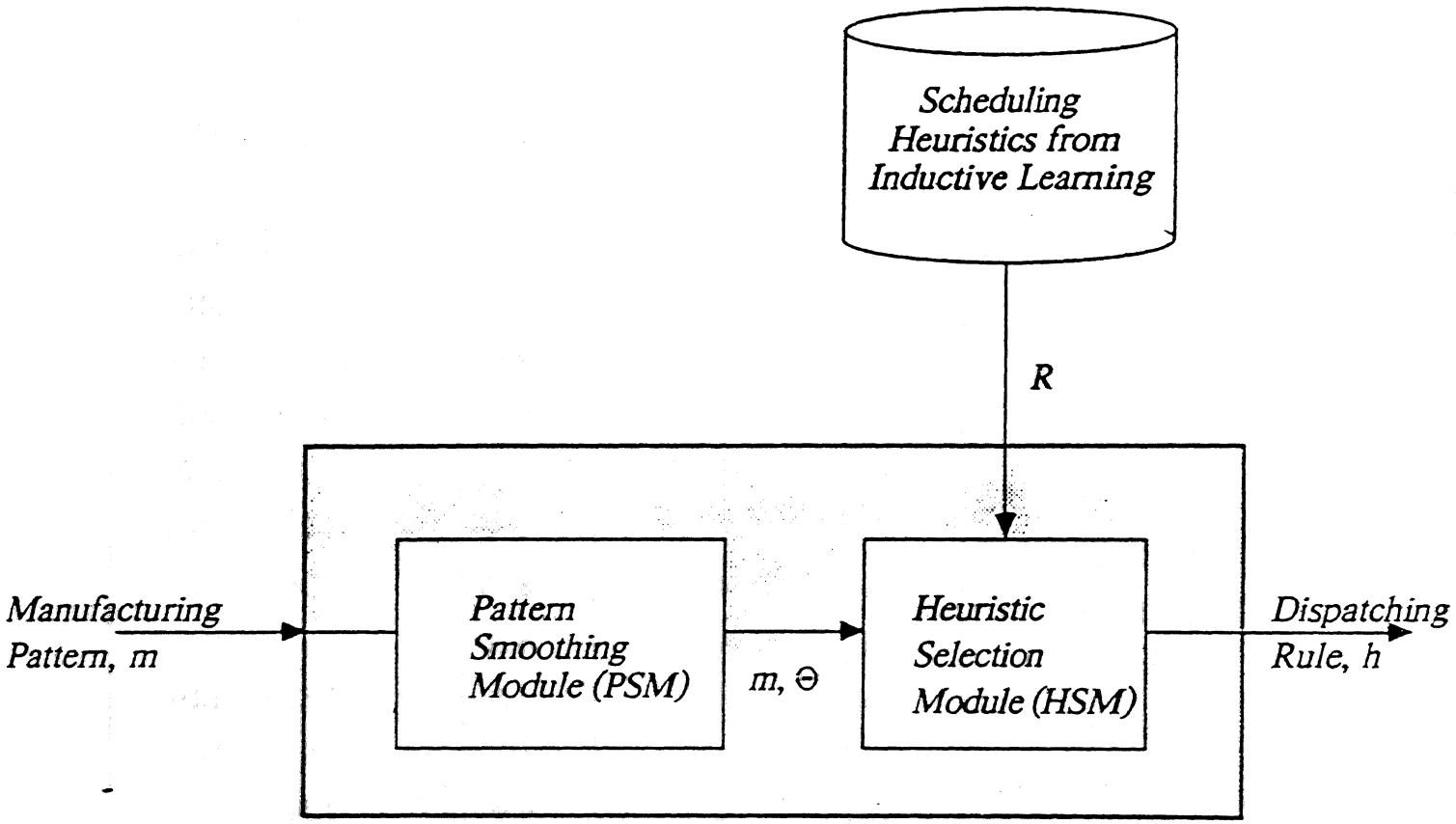

Figure 4: PDS Module Execution through its Functional Components

TABLE 1

Comparative Mean Tardiness Values

S. No.	*No. of Machines*	*Mean Tardiness under*		*Best Rule*
		BEST	PDS	
1	4	1991.00	1422.00	MDD
2	4	1834.00	1710.00	MDD
3	4	1063.00	915.00	MDD
4	4	983.20	983.20	MDD
5	4	660.80	660.80	MOD
6	4	97.94	87.90	MDD
7	4	58.34	54.37	MDD
8	4	56.28	40.27	MOD
9	4	51.73	28.48	MOD
10	4	38.62	25.61	MOD
11	4	16.88	16.50	MOD
12	4	11.31	18.34	MDD
13	4	11.05	9.23	MOD
14	4	8.13	4.96	EDD
15	4	5.26	6.36	MDD
16	4	4.58	3.62	EDD
17	4	1.06	2.93	MOD
18	4	0.90	0.34	EDD
19	4	0.86	0.52	EDD
20	4	0.65	0.00	MOD
21	4	0.56	0.48	MOD
22	4	0.53	0.07	MOD
23	4	0.48	1.10	EDD
24	4	0.33	0.13	MDD

TABLE 1 (continued)

S. No.	*No. of Machines*	*Mean Tardiness under*		*Best Rule*
		BEST	PDS	
46	8	1457.00	1457.00	MDD
47	8	1057.00	1079.00	MDD
48	8	812.90	812.90	MDD
49	8	774.10	774.10	MDD
50	8	772.20	772.20	MOD
51	8	772.00	772.00	MDD
52	8	515.10	515.10	MDD
53	8	422.50	422.50	MDD
54	8	146.00	146.00	MOD
55	8	61.30	61.30	MOD
56	8	61.30	16.76	MOD
57	8	18.86	18.86	MOD
58	8	14.84	14.84	MOD
59	8	11.87	11.87	MOD
60	8	7.88	7.47	MOD
61	8	4.83	4.64	MOD
62	8	4.08	3.16	EDD
63	8	1.67	1.67	MOD
64	8	1.48	1.48	MOD
65	8	0.85	0.85	MDD
66	8	0.33	0.45	EDD
67	8	0.23	0.04	MOD
68	8	0.04	0.04	EDD
69	8	0.03	0.39	EDD

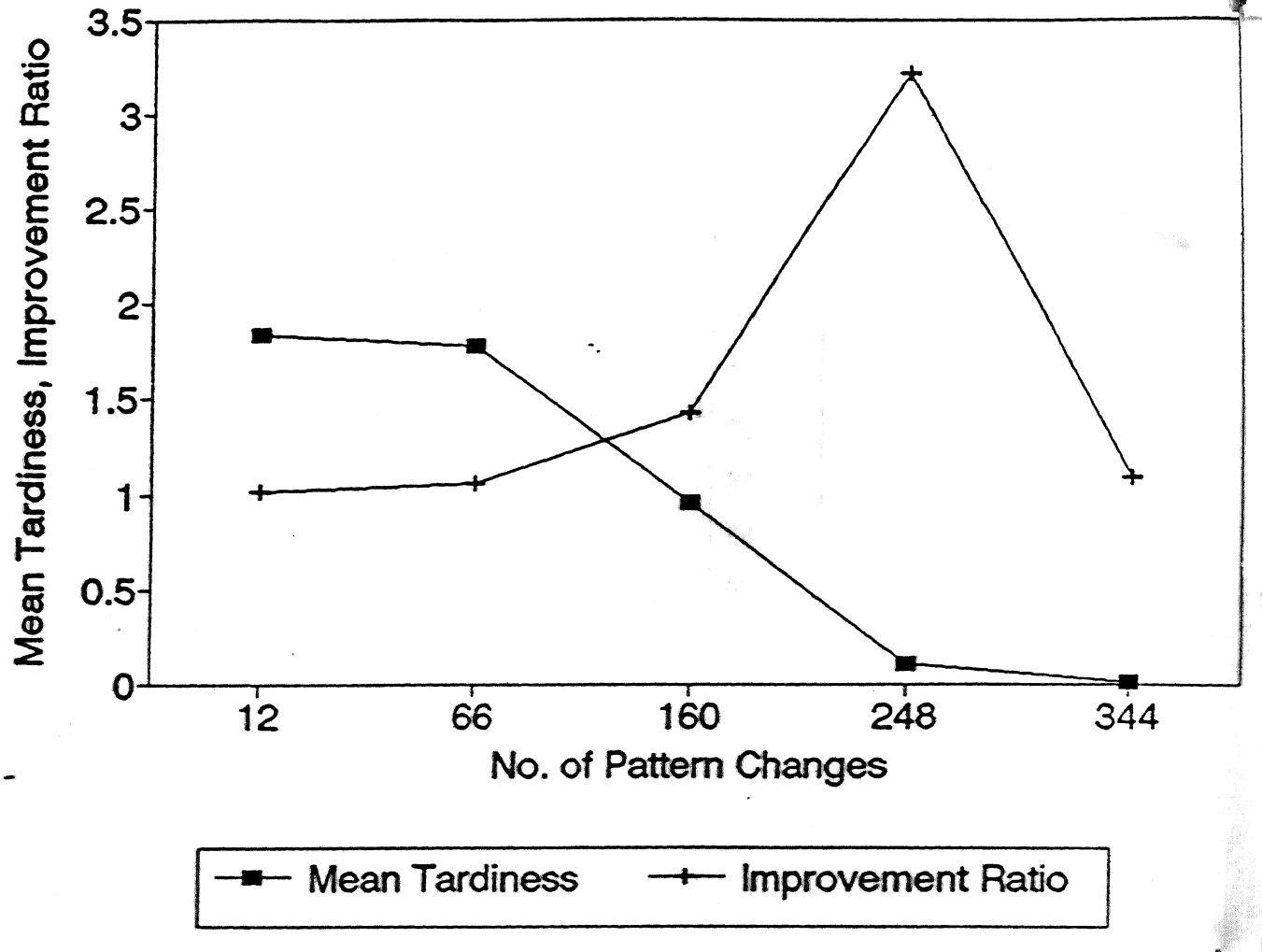

Figure 5: Impact of the Number of Pattern Changes