# A LEARNING-BASED SCHEDULING SYSTEM WITH CONTINUOUS CONTROL AND UPDATE STRUCTURE

**A THESIS**

**SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING**

**AND THE INSTITUTE OF ENGINEERING AND SCIENCE**

**OF BILKENT UNIVERSITY**

**IN PARTIAL FULFILLMENT OF THE REQUIREMENTS**

**FOR THE DEGREE OF**

**MASTER OF SCIENCE**

**By**

**Gökhan Metan**

**January, 2005**

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. İhsan Sabuncuoğlu (Supervisor)

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Prof. Dr. Erdal Erel

I certify that I have read this thesis and that in my opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

Asst. Prof. Dr. Mehmet Taner

Approved for the Institute of Engineering and Sciences:

Prof. Dr. Mehmet Baray
Director of the Institute of Engineering and Science

# ABSTRACT

## A LEARNING-BASED SCHEDULING SYSTEM WITH CONTINUOUS CONTROL AND UPDATE STRUCTURE

Gökhan Metan

M.S. in Industrial Engineering

Supervisor: Prof. Dr. İhsan Sabuncuoğlu

January, 2005

In today's highly competitive business environment, the product varieties of firms tend to increase and the demand patterns of commodities change rapidly. Especially for high tech industries, the product life cycles become very short and the customer demand can change drastically due to the introduction of new technologies in the market (i.e., introduction by the competitors). These factors increase the need for more efficient scheduling strategies. In this thesis, a learning-based scheduling system for a classical job shop problem with the average tardiness objective is developed. The system learns on the manufacturing environment by constructing a learning tree and selects a dispatching rule from the tree for each scheduling period to schedule the operations. The system also utilizes the process control charts to monitor the performance of the learning tree and the tree as well as the control charts is updated when necessary. Therefore, the system adapts itself for the changes in the manufacturing environment and survives in time. Also, extensive simulation experiments are performed for the system parameters such as monitoring (MPL) and scheduling period lengths (SPL). Our results indicate that the system performance is significantly affected by the parameters (i.e., MPL and SPL). Moreover, simulation results show that the performance of the proposed system is considerably better than the simulation-based single-pass and multi-pass scheduling algorithms available in the literature.

**Keywords**: Scheduling, Machine Learning, Data Mining, Control Charts, Job Shop Scheduling, AI, Dispatching Rules.

# ÖZET

## SÜREKLİ KONTROL VE GÜNCELLEŞTİRME YAPILI ÖĞRENME TEMELLİ ÇİZELGELEME SİSTEMİ

Gökhan Metan

Endüstri Mühendisliği, Yüksek Lisans

Tez Yöneticisi: Prof. Dr. İhsan Sabuncuoğlu

Ocak, 2005

Günümüzün rekabetçi iş dünyasında firmaların ürün çeşitleri artmakta ve malların talep düzeni hızlı bir şekilde değişmektedir. Özellikle yüksek teknoloji endüstrilerinde yeni teknolojilerin pazara tanıtımlarıyla ürün ömür çevrimleri kısalmakta ve müşteri talebi şiddetli şekilde değişmektedir. Bu etmenler verimli çizelgeleme gengüdümlerine olan ihtiyacı artırmaktadır. Bu tezde, geleneksel atelye problemine ortalama gecikme amacına yönelik öğrenme temelli çizelgeleme sistemi geliştirilmiştir. Önerilen sistem öğrenme ağacı kurmak yoluyla üretim ortamı üzerinde öğrenmekte ve bu ağaçtan herbir çizelgeleme dönemi için dağıtım kuralı seçerek işlemleri çizelgelemektedir. Sistem aynı zamanda süreç denetim çizeneklerinden faydalanarak öğrenme ağacının başarımını gözetlemekte ve gerekli bulduğunda ağacı ve denetim çizeneklerini güncellemektedir. Bu sayede, önerilen sistem kendi kendini üretim ortamındaki değişikliklere uyarlamakta ve zaman içinde hayatta kalabilmektedir. Bunun yanı sıra, çizelgeleme dönem uzunluğu ve gözetleme dönem uzunlukları gibi sistem parametreleri üzerinde kapsamlı benzetim deneyleri gerçekleştirilmiştir. Sonuçların gösterdiğine göre bu parametreler sistem başarımını (ortalama gecikme) önemli şekilde etkilemektedir. Bundan başka, benzetim sonuçları önerilen sistemin başarımının benzetim-temelli tek-geçişli ve çok-geçişli çizelgeleme algoritmalarından daha iyi olduğunu göstermektedir.

**Anahtar Sözcükler:** Çizelgeleme, Makina Öğrenmesi, Veri Madenciliği, Denetim Çizeneği, Atelye Çizelgelemesi, Yapay Zeka, Dağıtım Kuralları

To my family and my wife...

# Acknowledgement

I would like to express my deepest gratitude to my supervisor Prof. Dr. İhsan Sabuncuoğlu for his instructive comments in the supervision of the thesis and also for all the encouragement and trust during my graduate study.

I would like to express my special thanks and gratitude to Prof. Dr. Erdal Erel and Asst. Prof. Dr. Mehmet Taner for showing keen interest to the subject matter, for their remarks, recommendations and accepting to read and review the thesis.

I would like to express my deepest thanks to Kürşad Derinkuyu, Emrah Zarifoğlu, Ali Koç, M. Oğuz Atan, Halil Şekerci, Arda Alp and Mustafa R. Kılınç for all their encouragements and supports.

I would like to extend my sincere thanks to Orçun Ergün, Esra Büyüktahtakın and Banu Yüksel for their endless morale support and friendship during all my desperate times, makes me to face with all the troubles.

I would also like expressing my greatest thanks to Şermin Kamberli, Şengül Deveci and Uğur Deveci for being like my second family and showing all their supports.

Finally, I would like to express my gratitude to my family and my wife for their love, understanding, suggestions and their endless support. I owe so much to my family and my wife. I love you both…

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

In today's highly competitive business environment, customer satisfaction plays the key role for the success of any firm. Customers not only care about the cost of a product, but they also give special importance to the quality of the products and the reliability of the manufacturers in terms of meeting their agreements such as the promised due dates. Moreover, the product variety of a firm tends to increase due to the demand for highly customized goods, which in turn increases the complexity of operating a manufacturing system. In addition to these, the demand patterns of commodities may also change too rapidly. Especially for high tech industries, the product life cycles become very short and the customer demand can change drastically due to the introduction of new technologies in the market (i.e., introduction by the competitors). These factors increase the need for more efficient manufacturing strategies and approaches.

One of the key elements for the success of any manufacturing firm is efficient scheduling of its limited resources. However, even for a small sized company with a few number of equipments, it can become a very difficult problem to deal with.

In addition to this, scheduling problems should be solved frequently since it is the lowest level tactical decision for a firm. Therefore, development of efficient scheduling algorithms is vitally important and there is a vast amount of literature on this issue.

When the scheduling problem is stochastic and dynamic (i.e., the jobs arrive dynamically to the system and the arrival and processing times are stochastic) in nature, scheduling via the dispatching rules are commonly preferred. Dispatching rules are myopic decision rules that schedule the jobs on the machines one at a time based on the simple calculations that utilizing the information such as processing times, due dates etc. There are many such rules defined in the literature and we can simply pick one of them and perform the scheduling activities. However, the problem with these dispatching rules is that none of them is superior to the others in every manufacturing condition. Therefore, the appropriate rule/s should be determined prior to the use. In addition to that, even if a particular dispatching rule is found to perform better for a specific manufacturing system, switching to the other rules in certain periods may result in additional benefits. For this reason, there are also some simulation-based scheduling approaches in the literature. For such studies see, for example, Kim and Kim (1994), Jeong and Kim (1998), Kutanoglu and Sabuncuoglu (2001). In this approach, simulation-based scheduling, a set of candidate dispatching rules are simulated for a planning period and the rule with the best performance value is used in that period. One of the shortcomings of this approach is that it requires too much computer time to simulate the performance of each candidate dispatching rule. Also, the procedure depends on the assumption that we know the probability distribution functions and the parameters of the processing and arrival times. However, this may not be the case if the demand patterns in the market and/or product

types change rapidly, which is the situation for high tech industries. Also, the processing times may change due to the machines' depreciations in time. Hence, the simulation models constructed to evaluate the performance of the rules might become invalid after some time.

In this research, we consider the stochastic and dynamic job shop scheduling problem with the average tardiness (mean tardiness) objective and develop a system to select the right dispatching rule among a set of candidate rules. The proposed system utilizes the intelligent machine learning techniques from computer science (i.e., data mining) as well as the process control charts from the statistical quality control. The objective of our system is to learn about the characteristics of the manufacturing system by constructing a learning tree and then selecting a dispatching rule for a scheduling period from this tree on-line. Therefore, we eliminate the extensive simulation experiments that should be carried out before every scheduling period as it is in simulation-based scheduling approaches. Moreover, we use the control charts to monitor the actual performance of the learning tree. If these charts signal that the current learning tree begins to perform poorly, a new tree is constructed based on the recent information gathered from the manufacturing system. The reason for the current tree to have a poor performance might be a result of change in the demand patterns, processing time distributions and so on. Thus, by updating the current learning tree, we are targeting to capture these changes in the manufacturing system and select the right dispatching rules for the future periods. In this sense, the proposed system has the ability to survive in time. In other words, we propose a system that corrects itself whenever necessary (without an external manipulation) and continues to make the scheduling decisions (i.e., selecting the dispatching rules) as long as the manufacturing system exists.

In this study, we also address two important questions and conduct extensive experiments to answer them. One of these questions is “how frequently should we update the dispatching rule used in the manufacturing system?” This is a critical question since frequent selection of a new rule might result in system nervousness and, on the other hand, infrequent update of the rules most probably result in the loss of additional benefits that can be achieved by switching between the rules. The second question is “how frequently should we monitor the performance of the manufacturing system that operates under a rule and how should we decide to update or continue with this rule at these monitoring points?” Both of these questions are also important for the performance of our proposed system and experimented extensively.

In the next chapter, a review of the relevant literature is presented. In Chapter 3, we propose the intelligent scheduling system and discuss its key features in detail. Experimental designs and the results of these experiments are given in Chapter 4. Finally, in Chapter 5, we present the conclusion of this study along with the contributions and give future research directions.

# CHAPTER 2

# Literature Review

In the scheduling literature, there is a vast amount of studies that deal with various issues in scheduling. In this section, we will briefly review the relevant studies that employ iterative simulation and artificial intelligence (AI) concepts in manufacturing systems. In addition, we consider some studies related to process control charts as we use them as the tool in our research.

Wu and Wysk (1988) develop an expert system called *multi-pass expert control system* (MPECS) for flexible manufacturing cells. This system takes advantage of both expert system technology and discrete-event simulation. Simulation is employed as a prediction mechanism and evaluates the performance of the dispatching rules that are suggested by the expert system. Then, the dispatching rule that results the best performance value in simulation runs is used to schedule the jobs. This system also contains a simplified and restricted learning mechanism. This learning module uses training instances that relate the dispatching rules, the performance measures and the system characteristics together. By using this restricted learning mechanism, the system provides the user a learning aid, which collects information of the user interested factors (e.g., number of times a rule is selected, etc.)

to help the user learn from the system and modify the knowledge base if possible. In this sense, the system does not learn automatically by itself, but guides the user by providing significantly found information about the manufacturing system.

In another study by Wu and Wysk (1989), a simulation-based scheduling algorithm is proposed for flexible manufacturing system. In this research, a dispatching rule among a set of candidate rules is selected for each short period via simulation just before the implementation time occurs. The experiments on this candidate rule set are carried out by deterministic simulation and performance of each rule is estimated. Then the rule with the best performance estimate is used in that short period of time to schedule the operations. Since all the candidate dispatching rules are evaluated at each short scheduling period and the best performer is selected to be used in that interval, the proposed scheduling approach is termed as a multi-pass scheduling algorithm. Thus, in the long run, this process results in a combination of different dispatching rules. Their results show that the multi-pass scheduling algorithm performs better than the single-pass scheduling algorithm, which uses a single dispatching rule for the entire manufacturing period.

Another simulation-based scheduling study is due to Ishii and Talavage (1991). In this research, a transient-based real-time scheduling algorithm that selects a dispatching rule dynamically for a next short time period to react to changes of system state is proposed. In this study, as opposed to the work of Wu and Wysk (1989), the scheduling interval length, where each candidate dispatching rule is evaluated, is not held fixed and four different strategies are defined accordingly. In the first strategy, the simulation window (length of time used to evaluate the performance of candidate rules) is defined of equal length to the next scheduling interval as it is in the study of Wu and Wysk (1989). In the second strategy, simulation window is defined from the

current time to the time until all parts that exist in the system during the next scheduling interval depart from the system. For the third strategy, they define simulation window as from current time to the end of the entire manufacturing period. Finally, the last strategy assumes simulation window as the two consecutive scheduling intervals and selects the best rule for the first scheduling interval based on the performances measured at the end of the second scheduling interval. In this sense, the last strategy employs a single period look-ahead mechanism. It is reported in the paper that in most of the experiments strategy 4 results in better schedules than the other strategies as well as the single-pass scheduling algorithm.

The first study that applies machine learning techniques to the scheduling problems is the work of Shaw *et. al.* (1992). In this paper machine learning capabilities for an FMS scheduling problem is investigated. Their machine learning approach is used to select the best dispatching rule based on a number of manufacturing system characteristics (the overall system utilization, total buffer size and number of machines). This selected rule is then used to schedule the jobs on the machines, and the rule is never questioned again as long as the shop floor configuration is stable (e.g., number of machines in the facility doesn't change). Therefore, the decision given in this study can be thought of as a strategic decision rather than a tactical one. Training examples are generated for different attribute-value combinations. These examples are supplied for the learning algorithm as a learning data set after being tested via simulation. After the learning algorithm processes the learning data, a learning tree is constructed. Whenever one or more of the system characteristics changes (takes a different value than its current value), the algorithm selects a new dispatching rule (DR) from the learning tree based on the new values of the attributes. It does not implement the new DR immediately, but rather it makes a

new decision about changing the current DR with this new one or not. This is because of the fact that some attribute changes may be temporary and changing the DR of the manufacturing system may be destructive when compared with the expected performance of the current DR. This decision is done in such a way that if the cumulative score (number of times a DR is favored to the others) of the new DR is greater than the cumulative score of the current DR multiplied by a smoothing factor, the new DR is selected for use. Otherwise, the system continues its operation with the current DR. Here, the smoothing factor is a decision variable between 0 and 1. Also, since smoothing factor is a decision variable, experimentation on this variable is performed with different attribute values for three smoothing factor values (0, 0.7 and 1) and another learning tree is constructed for the selection of this variable. In other words, the value of the smoothing factor is not a fixed value but its value is determined based on the system attributes from the second learning tree whenever a new DR is to be selected from the first learning tree. By using this machine learning strategy, Shaw *et. al.* test their algorithm on different FMS problems. The results indicate that the proposed approach outperforms the approach of using the single best DR from a set of candidate DRs in most of the cases.

In another series of studies by Tayanithi, Manivannan, and Banks (1993a, 1993b), an integrated scheduling and control system that combines simulation and knowledge-based concepts to perform an analysis of interruptions in the form of machine breakdowns and rush orders in a flexible manufacturing system is proposed. In this system, when a control decision cannot be obtained readily from the knowledge base, the alternative actions are evaluated by using the simulation mechanism.

Cho and Wysk (1993) propose a neural network based scheduling algorithm for FMS. Their system mainly composed of three parts: a preprocessor, a neural network and a multi-pass simulator. Preprocessor generates input based on the current workstation status and supplies it to the neural network. In turn, neural network produces a set of promising part dispatching strategies (i.e., dispatching rules) to guide the future scheduling activities. These strategies are then evaluated by the multi-pass simulator and the best strategy to use is determined. Then the selected strategy is used in the shop floor until a new rule update is required. The performance of the algorithm is compared with the single-pass strategies and found to be superior.

Ishii and Talavage (1994) propose another simulation-based scheduling system for flexible manufacturing systems. In this research, they develop a mixed dispatching rule approach in which each individual machine in an FMS are allowed to have a different dispatching rule to perform the scheduling of jobs. It is assumed in the paper that the candidate dispatching rule set is predetermined and a search strategy to select the best combination from these candidate rules is employed. The effectiveness of the mixed dispatching rule approach is demonstrated by comparing the experimental results with the conventional approach, where a single dispatching rule is assigned for all machines in a system for a given scheduling interval.

One of the simulation-based studies for scheduling problems is due to Kim and Kim (1994), there is a candidate DR set and the rules in this set are evaluated at the beginning of each planning horizon by deterministic simulation. The best performer is then selected for use for that planning horizon. There are also monitoring points defined within the planning horizon and the actual performance of the DR (based on the stochastic simulation which represents the real life situation) is compared with the estimated one (from deterministic simulation at the beginning of

the planning horizon). If this difference exceeds the limit then a new DR is selected from the candidate set by the same procedure mentioned as above. In the follow up study, Jeong and Kim (1998) have extended the previous study. The dispatching rule selection approach is the same as the first study, where a set of candidate DRs are evaluated via simulation and the best one is selected for implementation. The major development in the later study is that the results from the different policies are defined for the question of "when to select a new rule?" Specifically, four different alternative policies are defined and compared in this study. The first policy is called as BEGIN and only selects a new DR at the beginning of each planning horizon. The second policy, MAJOR, considers selecting a new DR at the beginning of each planning horizon and at times within the planning horizon whenever a major breakdown of a machine occurs. The third one, MAJOR and PERIODIC (M&P), selects a new DR as MAJOR and additionally at monitoring points. And the final policy, so called ALL, selects a new DR at the beginning of each planning horizon, when a major breakdown occurs and a minor breakdown occurs, as well. The concept of major and minor breakdowns is a subjective issue and is defined by the authors in the paper with some parameters. The results of the experiments in the paper show that M&P and ALL perform best. Moreover, while evaluating the candidate DRs in the previous paper (Kim and Kim, 1994), authors used deterministic simulation. In this paper, authors also test the effect of using deterministic and stochastic simulations in the decision phase, that is, the point where we will select the best performing DR via simulation. Results show that the deterministic simulation, where the machine breakdowns are not considered, resulted in better selections of DRs.

Pierreval and Mebarki (1997) propose a system by which dispatching rules are selected dynamically. Their aim is to monitor the system continuously and select the

most suitable dispatching rule for each work-center to optimize the system performance. Actually, this research includes the following developments:

i- Allows for more than one performance criteria to be considered simultaneously (both primary and a secondary criteria are considered at the same time).

ii- Based on the *specified performance criteria*, dynamic selection of the DRs seems to be a good policy for the *operating conditions* and *current shop status*.

iii- The capability of tracking the triggering events such as new job arrivals, resource availabilities etc.

In the light of these developments, a new heuristic technique called SFSR (shift from standard rules) is proposed. This mechanism has a default dispatching strategy based on the specified performance criteria. These default dispatching strategies are called as the 'Standard Rules'. For example, R1, which is defined as the standard rule that applies whenever the primary objective is to reduce the mean flow time of jobs dictates the system the SPT rule and it is active if there is no anomalies (no triggering events) in the system. These standard rules are obtained from the literature based on their performances in the previous studies. There is also a second class of rules called as the 'Diagnosis Rules' that accounts for a major development. These rules work according to the symptoms detected in the system by continuously monitoring. A defined set of symptoms (new job arrivals, resource availability etc.) and their corresponding actions, so called the Diagnosis rules, aim at achieving better performances. However, the generation of these diagnosis rules is not based on a data mining approach but rather they are common sense rules that are based on experiences of humans. In this sense, this research cannot be classified as a machine

learning approach to scheduling problems, but it can be classified as a scheduling by using heuristic rules.

In another study by Kutanoglu and Sabuncuoglu (2001), an iterative simulation-based approach for the dynamic and stochastic job shops is proposed. In this study, at the beginning of each scheduling period, a set of DRs are tested via simulation under the current system conditions and the forecasts. The best performing DR is selected for the upcoming period and used until the next scheduling period. The rolling horizon technique is also employed in this study since the simulation runs are taken for longer time periods (more than one scheduling period).

Suwa and Fujii (2003) use a machine learning technique (data mining) for rule acquisition in a dynamic single machine scheduling problem. The training examples to the learning module are generated via simulation and then the learning tree is constructed. Afterwards, the learning tree is used for selecting the appropriate DR to schedule the jobs in a rolling horizon basis. The attributes used in this study to represent both the training examples and the conditions when selecting a new DR at the beginning of a new period are based on some performance measure differences between the current period and the last period. The learning tree is used forever after once it is constructed and no revision or critique of the existing rule base is performed.

A related study is the working paper by Huyet and Paris (2003). In this study, an evolutionary optimization method is used with machine learning in order to set the parameters of a Kanban system optimally. A population of 30 individuals is used in each generation of GA and at every three iterations, the machine learning is used to learn about the characteristics of promising solutions. Then a number of solutions generated randomly, but which have the characteristics found to be important by the

machine learning are embedded into the new generation and the process continues. In this research, machine learning mainly accelerates the convergence of the individuals and hence find the optimum (or near optimum) solutions more rapidly (approximately half of the iterations are found to be sufficient for the same level of convergence when compared with the GA used alone). This research is a good example to show the power of the machine learning approaches when they are employed effectively.

Another two related studies that show the applicability and usefulness of one of our tools in our proposed approach is the papers of Takahashi and Nakamura (1999, 2002). In these two papers, a reactive Kanban system is proposed, where the number of Kanban cards in the system is manipulated continuously as a response to the system parameter, the unstable changes in demand (both mean and the variance of the demand distribution is subject to change continuously). Since the demand distribution is not stable, the optimal parameters of the Kanban system (number of Kanban cards, Kanban container sizes etc.) change dynamically, as well. Therefore, appropriate actions should be taken in order to operate optimally or at least near optimally. In this paper, the Process Control Charts (EWMA) from quality control are employed in order to monitor the unstable changes in demand parameters. The demand distribution is assumed to be normally distributed and the appropriate actions are taken whenever the chart signals a change in the mean and the variance of the demand distribution.

In this chapter, we presented the relevant literature to our study. In the next chapter, we present our learning-based scheduling approach in detail and give a numeric example to illustrate the learning procedure.

# CHAPTER 3

# Proposed System: Intelligent Scheduling with Machine Learning

In this research, we propose a learning-based scheduling technique where the dispatching rules (DRs) that are used to schedule the jobs on the machines are selected by the learning tree. Moreover, the system adapts itself to the changes in manufacturing conditions. To achieve this, the performance of the learning tree is monitored against the considerable changes in manufacturing system parameter(s) via the process control charts. Whenever the control charts signal out a change in the manufacturing conditions, the learning algorithm uses the new available data gathered from the system to re-learn about the characteristics of the manufacturing environment to make better decisions in the future periods. Control charts are also updated whenever necessary.

In this Chapter, we discuss the structure of our learning-based scheduling system. First, we will start our discussion by giving important definitions that are frequently used in the rest of the Chapter. Then, we will present our proposed system in general terms. After discussing the scheduling strategies and the data structures employed, we will give a detailed explanation of our learning-based scheduling

system. Following these, we will define our system attributes and explain the internal dynamics of the learning procedure used and illustrate its steps with an example.

## 3.1. Definitions

**Scheduling period** is a time interval during which a selected DR is used to schedule jobs. The rule can be changed before the end of the scheduling period if some changes occur in system conditions. In such cases, this scheduling period is said to be **incomplete**. Otherwise, it is of type **complete**.

**Instance data** is composed of a number of attributes and a class value, where attributes take values of manufacturing conditions and class value corresponds to the DR selected for a specific condition.

**Realized scheduling period data** represents the actual events that occur in a specific scheduling period. It includes realized values of random variables such as the processing times, interarrival times and system conditions at the beginning of the scheduling period. This data set is stored in the database and is provided for the simulation module when demanded.

**System attributes** are a predefined set of variables that carry information about the state of the real manufacturing system such as queue length, total remaining processing times, etc.

**New rule selection symptoms** are the triggering events that are defined in the scheduling strategy to answer the question of "when-to-schedule".

**Scheduling strategy** determines "when-to-schedule" and "how-to-schedule" decisions (Sabuncuoglu and Goren, 2003).

**Process control chart** is a statistical chart used to monitor the quality of the decisions given by the learning tree.

## 3.2. Proposed System

The proposed scheduling system is an intelligent scheduling mechanism that employs machine learning capabilities from AI as well as the process control chart concept from quality control. The goal of the system is to select the best DR among Candidate Dispatching Rules (CDRs) for a particular scheduling period. The general structure is shown in Figure 3.1.

In the proposed system, there are five main subroutines, called modules. They operate in harmony to achieve the goal of selecting the best performing dispatching rule for each scheduling period. The operations of these five modules are as follows. The *database* provides necessary data for both the learning module and the simulation module. It holds the instance data for the learning algorithm to generate the learning tree. The realized scheduling period data is also stored in the database for assessment of DRs via simulation. *Simulation module* is used to measure the performances of the candidate dispatching rules. The simulation module is invoked by the *process controller module* whenever necessary. Simulation module's outputs (instance data) are sent to the database. These results are then used by the *learning module* to generate the learning tree. Whenever a scheduling decision is to be made according to the current scheduling strategy (e.g., hybrid approach), the learning tree selects a new dispatching rule and this decision is implemented by *on-line controller module* (i.e., it employs the selected DR in actual manufacturing conditions). It also supplies the realized scheduling period data to the database and monitors the real system for new rule selection symptoms. The *process controller module* monitors the performance of the learning tree. It takes its inputs (realized average tardiness) from the on-line controller module and monitors the performance of the learning tree. When the performance of the current learning tree is found to be insufficient, it requests from

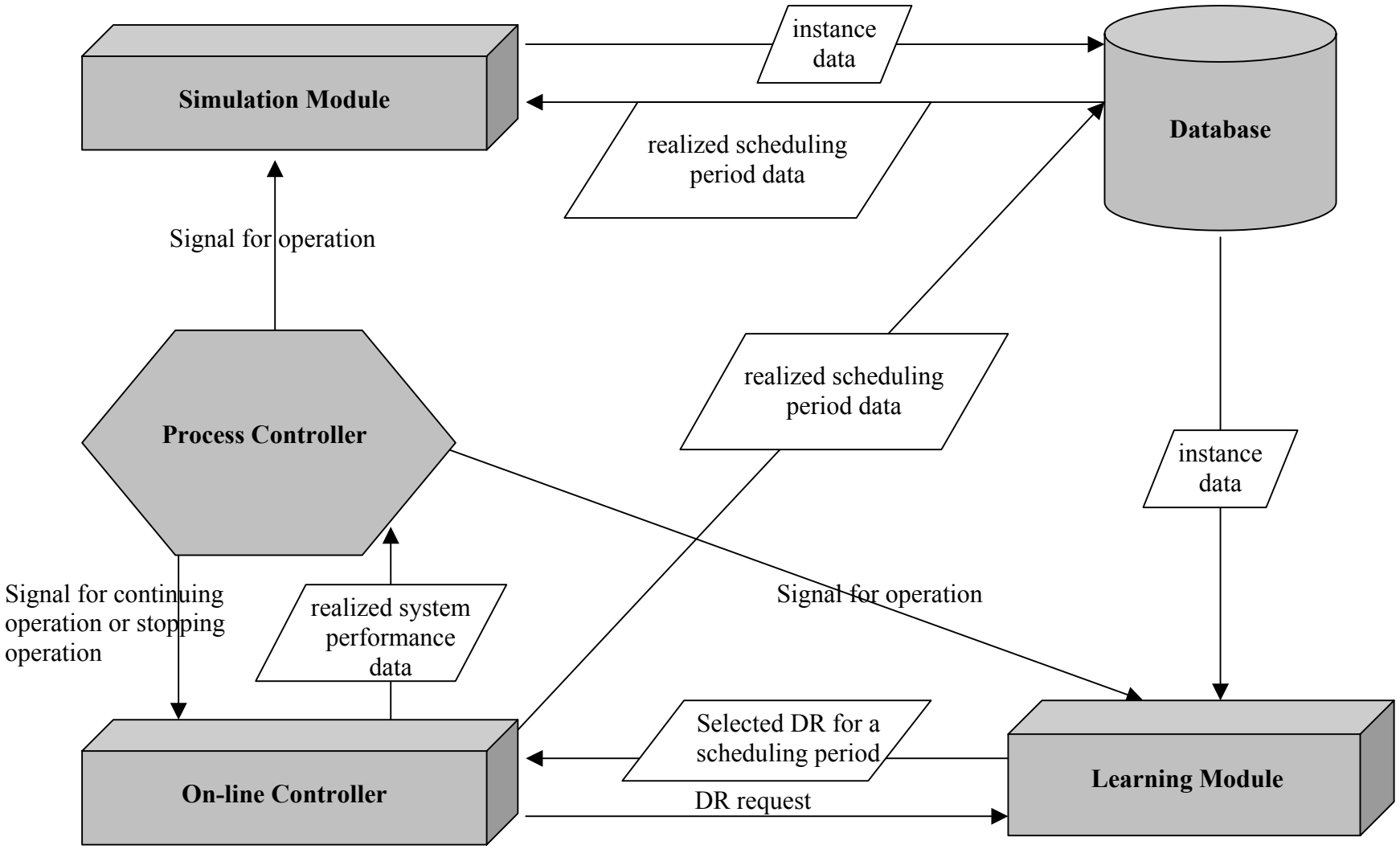



Figure 3.1: Proposed System – General Structure

the simulation module to provide new training data (instance data) for the learning module and then sends a signal to the learning module to update the current learning tree with this new data set. As a result, new dispatching rules are selected from this updated learning tree and the process continues in this manner.

## 3.3. Scheduling Strategy

The scheduling strategy employed in this research is composed of two critical decisions: how-to-schedule and when-to-schedule. They are explained below:

### 3.3.1. How-to-schedule

How-to-schedule decision determines the way in which the schedules are revised or updated. As discussed in Sabuncuoglu and Goren (2003), there are mainly three issues: scheduling scheme, amount of data used, and type of the response. The scheduling scheme can be off-line, on-line, or a combination of the two (i.e., hybrid). *Off-line scheduling* refers to scheduling all of available jobs for the entire scheduling period before the execution of the schedule. On the other, hand *on-line scheduling* is to take scheduling decisions one at a time during the execution of the schedule (e.g. scheduling via priority dispatching rules). Between these two extremes, hybrid or quasi-online scheduling lies. In quasi-online scheduling, a subset of the jobs is scheduled off-line and the rest of the schedule is developed as time goes on. The second issue is related to the amount of data used during the schedule generation process. This can be *full* or *partial*, where all the forecasted data is used in the former case whereas only a proportion of the available data is used in the partial case. The third issue is the *type of the response*. This is related to the question of "what should be done if the current schedule begins to perform worse". One possibility can be rescheduling, where a new schedule obtained from scratch. Another alternative can be

to take no corrective action (i.e., letting the system recover itself from the negative effects of disruptions). In addition, match-up scheduling or right/left shifting the remaining jobs can also be used for a type of response.

Our implementation is based on the on-line scheduling scheme. Specifically, DRs are selected by the learning tree and the scheduling decisions are made one at a time using these selected rules (see Table 3.1). In terms of the amount of data, we apply the "full" scheme, since all available information about the real manufacturing system is utilized to select a DR for a scheduling period. As the type of the response, we use "reschedule" option, as a new DR is selected at any time when the existing DR is found to be poor.

Table 3.1: Parameters of how-to-schedule

| | |
|---|---|
| **Scheduling Scheme** | On-line |
| **Amount of Data** | Full |
| **Type of Response** | Reschedule |

### 3.3.2. When-to-schedule

"When-to-schedule" determines the responsiveness of the system to various kinds of disruptions. As discussed in Sabuncuoglu and Goren (2003), there are different alternatives to decide on the timing of scheduling decisions. The first way is to schedule the system periodically, which is called as *periodic scheduling*. In periodic scheduling, the time intervals can be constant or variable. In the former case, schedule revisions are made at the beginning of fixed time intervals. In the latter case, revisions are made after a certain amount of schedule is realized. Another alternative, which is called *continuous scheduling*, updates the schedule after a number of random

events occur such as machine breakdowns, or a new job arrival, etc. In *adaptive scheduling*, a scheduling decision is made after a predetermined amount of deviation from the original schedule is observed. For example, a scheduling decision is triggered when the difference in average tardiness between the initial and the realized schedules exceeds a threshold value, say 10 minutes. There are also hybrid approaches, which are combinations of the above strategies, and in this research such a hybrid approach is employed for "when-to-schedule" decisions. In our hybrid approach, two different triggering events, called as *New Rule Selection Symptoms*, are defined for the time of selecting a new DR. These new rule selection symptoms and their definitions are given in Table 3.2.

Table 3.2: New Rule Selection Symptoms

| **Abbreviation** | **Name** | **Description** |
|---|---|---|
| **BSP** | Beginning of each Scheduling Period | Triggers the selection of a new DR at the beginning of each new scheduling period. |
| **MP** | Monitoring Points | Triggers the selection of a new DR at the monitoring points whenever necessary. |

The length of a scheduling period ($L_{SP}$) is a decision variable and a new DR is selected at the beginning of each period to carry out the dispatching process until the end of that scheduling period. As seen in Figure 3.2-a, the beginning of each scheduling period is a triggering event for selecting a new DR. However, a selected DR is not always used until the end of a scheduling period because of the existence other symptoms, MP, that may occur in the scheduling process. In such cases, a new DR is selected before the end of a scheduling period.

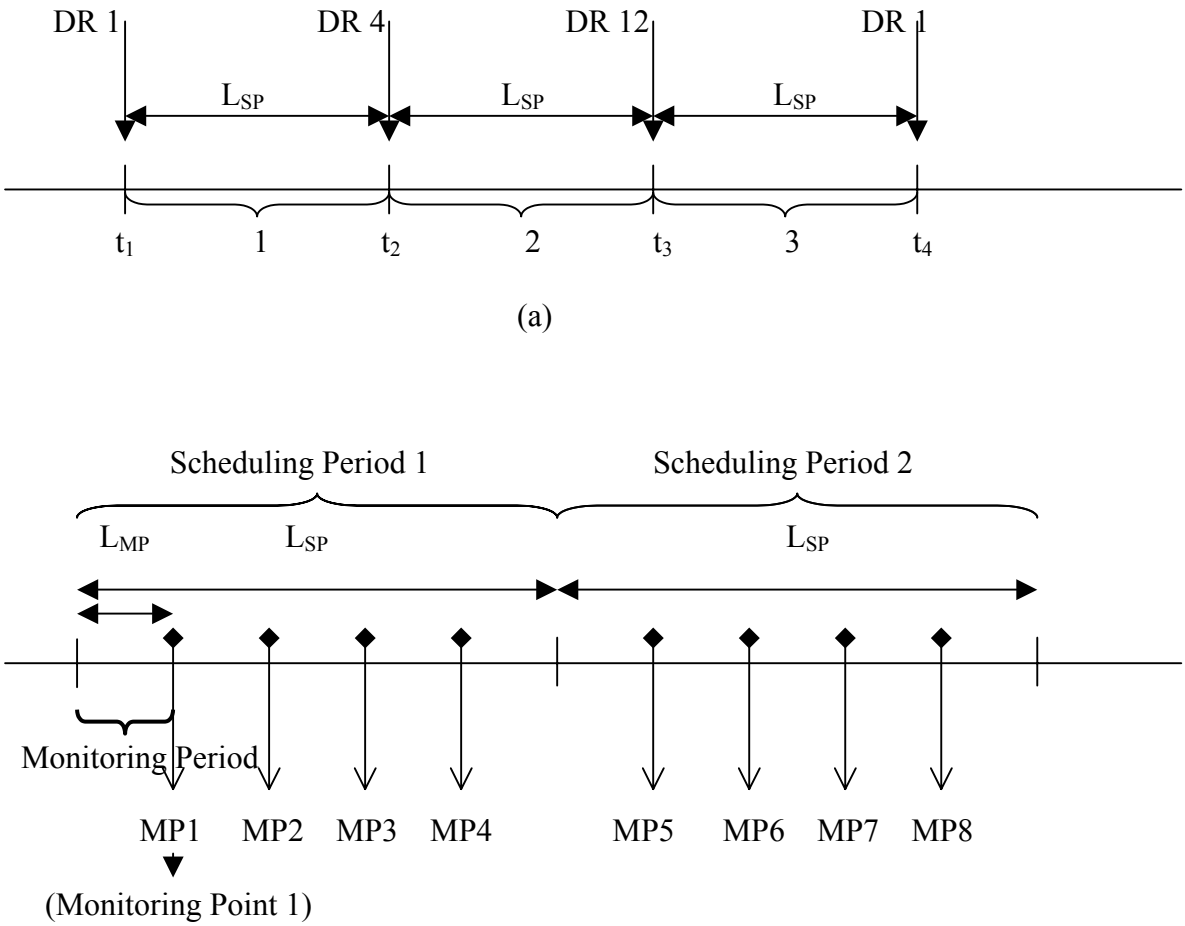



Figure 3.2: Representation of Rule Selection Symptoms. (a) New Rule Selection Symptom (BSP). (b) Monitoring Period and Monitoring Point

As seen in Figure 3.2-a, the performance of the current DR is monitored regularly at monitoring points and if it is found to be poor (i.e., the performance is worse than a certain percentage of the desired level), a new DR is requested from the learning tree. The length of a monitoring period ($L_{MP}$) is usually a decision variable (or policy variable) and a complete scheduling period contains a fixed number of monitoring points. $L_{MP}=L_{SP}/(k+1)$, where k is the number of monitoring points in a complete scheduling period (Figure 3.2-b).

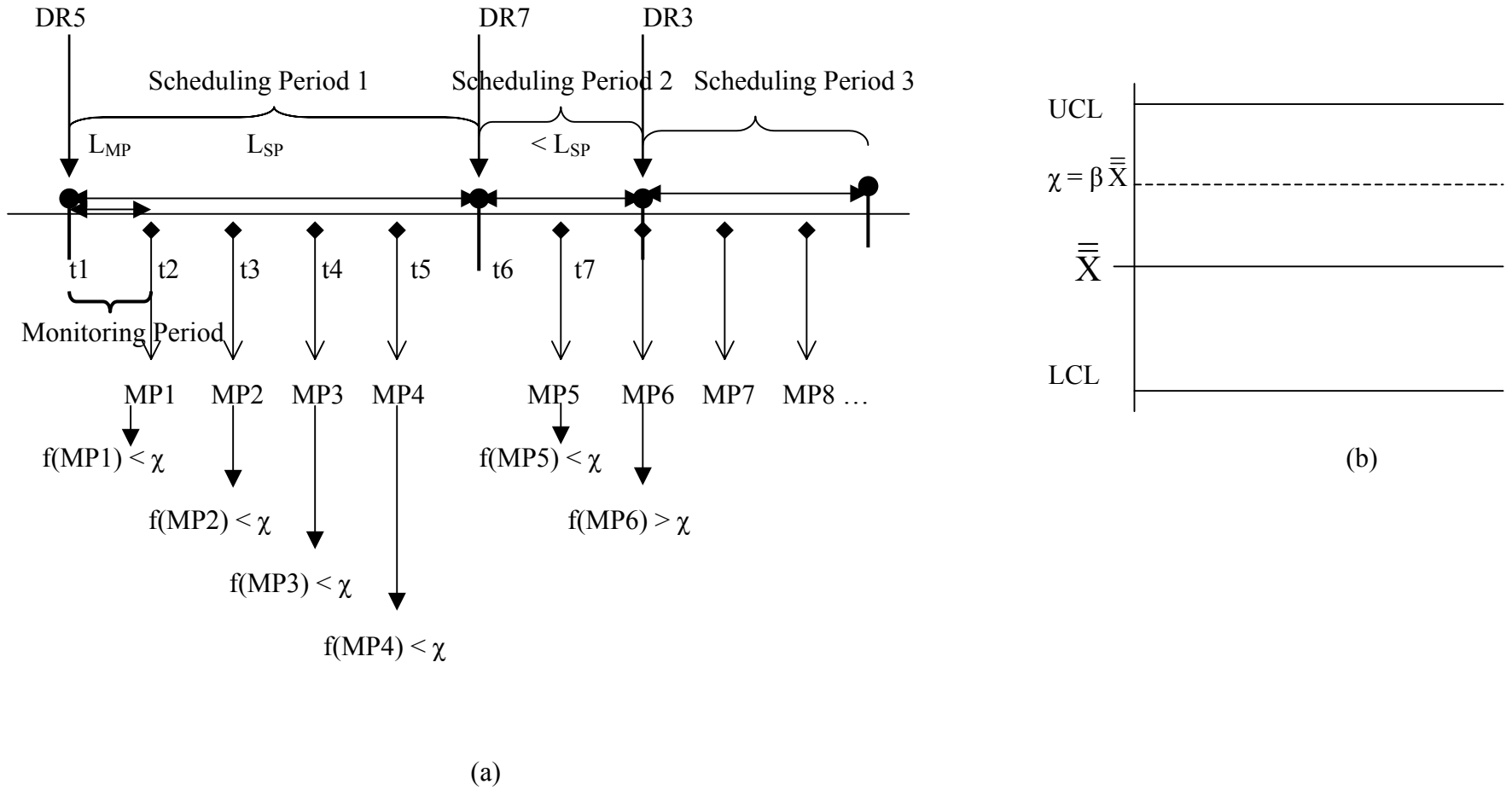



Figure 3.3: Rule Selection Symptoms. (a) New Rule Selection Symptom (MP). (b) Representation of χ in control.

As an example, consider the case in Figure 3.3, which displays the MP symptom and the actions to be taken. At every monitoring point in a scheduling interval, the current value of the performance measure is compared with a threshold value. If it is worse than the threshold, a new DR is requested from the learning tree. Otherwise, the system continues with the current DR. In Figure 3.3, in none of the monitoring points of the scheduling period 1 there is a need for a change and hence DR5 is used throughout the scheduling period 1 (i.e., type complete). At the beginning of scheduling period 2 (at t6), DR7 is selected as a new rule by the learning tree. At the monitoring point 6, its performance f(MP6) is found to be worse than the threshold value $\chi$, and a new DR is requested from the learning tree. Based on the learning tree recommendation, DR3 is assigned as the new DR for the scheduling period 3. Note that the scheduling period 2 is now of type incomplete, since its length is less than $L_{SP}$.

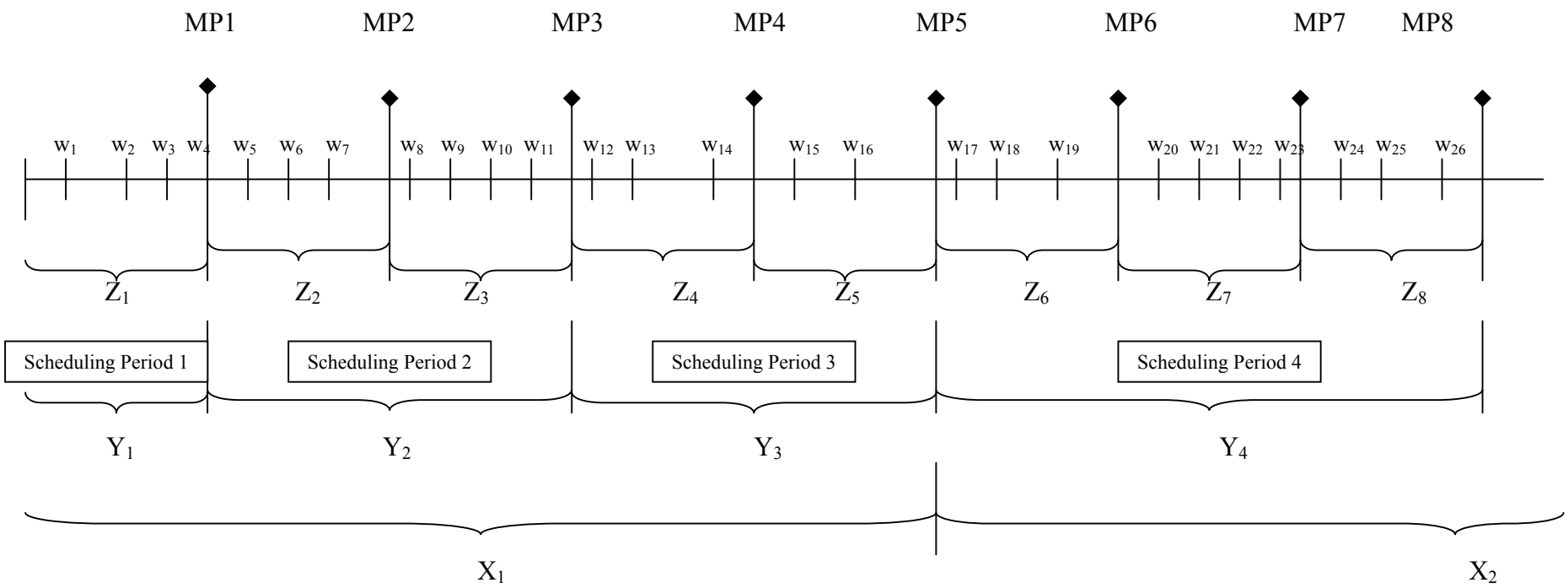
The function f(*) gives simply the average tardiness value of the completed jobs from the beginning of the current scheduling period. For example, f(MP2) is the average tardiness of all the jobs completed between the times t3 and t1, and f(MP5) is the average tardiness of all the jobs completed between the times t7 and t6 (see Figure 3.3). As seen in Figure 3.3-b, the threshold value $\chi$ is a multiple of the expected average tardiness ($\chi = \beta \overline{\overline{X}}$, where the parameter $\beta$ is $0<\beta$ and $\overline{\overline{X}}$ is the long-run expected average tardiness).

## 3.4. Data Structures

There are different data types used in the proposed scheduling system. These are explained below.

### 3.4.1. Performance Data (Realized System Performance)

*Performance Data* is the data type that represents the performance of a DR in a specific scheduling period in terms of tardiness, average tardiness and the average of average tardiness. These data are used for different purposes but in the following formats. We define three different formats for performance data as: *monitoring period performance* (Z), *scheduling period performance* (Y), and *aggregated performance* (X). In Figure 3.4, each of these data structures are displayed in detail. Each $w_i$ value is an individual tardiness value of a completed job. Monitoring period performance (Z values) is the average tardiness of all the completed jobs between the last monitoring point and the current monitoring point. For example, $Z_1=(w_1 + w_2 + w_3 + w_4) / 4$ for MP1, and $Z_2= (w_5 + w_6 + w_7) / 3$ for MP2. In other words, the $Z_i$ values are the average tardiness realized in a monitoring period. Scheduling period performance (Y values) is the average tardiness of all the completed jobs in a scheduling period. For example, $Y_1= Z_1$, and $Y_4= \left(\sum_{i=17}^{26} w_i\right)/10$. Note that scheduling period 1 is of type incomplete and contains only one monitoring period whereas scheduling period 4 is of type complete and consists of three monitoring periods (in this illustrative example a complete scheduling period is assumed to contain three monitoring periods). In other words, the $Y_i$ values are the average tardiness realized in a scheduling period. Aggregated performances, $X_i$ values, are samples of $Y_i$ values. In Figure 3.4, each $X_i$ value is defined as the average of three $Y_i$ values (the number of $Y_i$ values to be grouped is a parameter) and therefore $X_1= (Y_1+Y_2+Y_3)/3$, $X_2=(Y_4+Y_5+Y_6)/3$, $X_3= (Y_7+Y_8+Y_9)/3$. In other words, aggregated performance is the average tardiness realized in a number of consecutive scheduling periods.




$Y_1 = Z_1$ ; $Y_2 = \left(\sum_{i=5}^{11} w_i\right)/7$ ; $Y_3 = \left(\sum_{i=12}^{16} w_i\right)/5$ ; $Y_4 = \left(\sum_{i=17}^{26} w_i\right)/10$

$X_1 = (Y_1+Y_2+Y_3)/3$ ; $X_2 = (Y_4+Y_5+Y_6)/3$ ; $X_3 = (Y_7+Y_8+Y_9)/3$ .

Figure 3.4: Types of Performance Data

Among these data structures, the $Z_i$ values are used for monitoring the performance of the current DR at monitoring points. $Y_i$ values are used as the performance value of the DR that is used in a specific scheduling period. It is also a part of the realized scheduling period data, which is used by the simulation module. Finally, the $X_i$ values are used for the performance evaluation of the existing learning tree and these are the data that are plotted on the $\bar{X}$ chart. $Y_i$ values are aggregated to form $X_i$ values because of the normality assumption requirement of the control chart. Aggregating four to six data is sufficient for meeting this requirement.

### 3.4.2. Instance Data

Figure 3.5 shows the representation of instance data. Each row in this figure corresponds to an individual data, which has a number of *attributes*, the *class value* that indicates the best DR that works under these specific attribute-value combinations and the *performance value* (scheduling period performance) of that DR.

| Attribute-1 | Attribute-2 | … | Attribute-n | Performance Value | Class Value |
|---|---|---|---|---|---|
| A-1 Value | A-2 Value | … | A-n Value | f(DR3) | DR3 |
| A-1 Value | A-2 Value | … | A-n Value | f(DR5) | DR5 |
| A-1 Value | A-2 Value | … | A-n Value | f(DR7) | DR7 |
| … | … | … | … | … | … |
| A-1 Value | A-2 Value | … | A-n Value | f(DR1) | DR1 |

Figure 3.5: Instance Data Representation

These data are created from the outputs of the simulation module and are used for two important reasons in the system. First, it is used in the construction of the learning tree, where these data are supplied to the learning algorithm to make inferences about the characteristics of the manufacturing system based on the pre-specified set of attributes. Second, it is used to construct the process control charts. The column that stores the performance values is supplied to the process controller module whenever the process control charts are to be updated. In the second usage of

the instance data, only the performance value column is used. This column contains the best performance values found by the simulation module under different DRs based on the specified attribute values. These data points are represented as $X_i^*$ indicating the best performances (average tardiness) found for specific system conditions. Since, we monitor the performance of the learning tree relative to the best performance; we employ $X_i^*$ values when constructing the process control charts (for detailed information about f(DRj) see section 3.5.2. Simulation Module).

### 3.4.3. Realized Scheduling Period Data

In Figure 3.6, the *realized scheduling period data* structure is depicted. At the end of any scheduling period, the on-line controller module sends all the relevant realized manufacturing system data to the database. These data points include the values of the system attributes at the beginning of the scheduling period (scheduling period k in our case), the realized random events during that period as well as the average tardiness value obtained under the current DR in use in that scheduling period. In the current implementation, since we model actual manufacturing conditions in a simulation model, we store the seed values of the random number generations for each stochastic variable in this column. Thus, the entire history is easily generated using these seeds when necessary. Hence, these data points are the result of the tracking of the system by the on-line controller. The importance of this data type comes from the following fact: when a DR is selected by the learning tree and used in a scheduling period, we do not know whether it is actually the best DR for that scheduling period. The only way to know it is to simulate the other DRs in the CDR set under exactly the same system conditions. These data points provide an important feedback for the system to

improve the quality of the learning tree whenever necessary (these issues will be discussed later in the text).

| Scheduling Period | Attribute-1 | Attribute-2 | … | Attribute-n | Realized Scheduling Period Data |
|---|---|---|---|---|---|
| 1 | A-1 Value | A-2 Value | … | A-n Value | Realized Scheduling Period-1 Data |
| 2 | A-1 Value | A-2 Value | … | A-n Value | Realized Scheduling Period-2 Data |
| … | … | … | … | … | … |
| k | A-1 Value | A-2 Value | … | A-n Value | Realized Scheduling Period-k Data |

Figure 3.6: Realized Scheduling Period Data

## 3.5. Proposed System – A Detailed Explanation

General structure of the proposed system has been introduced in Section 3.2. We now explain each module in detail.

### 3.5.1. Database

The database of the proposed system is composed of two layers, called as D1 and D2 (Figure 3.7). D1 stores the "realized scheduling period data" discussed in Section 3.4.3. These data are supplied from the on-line controller to the simulation module. D2 stores the instance data discussed in Section 3.4.2. These data are supplied from the simulation module, and are used by the learning module to generate the learning tree.

As stated earlier, D1 stores the input data for the simulation module and D2 stores the output data of the simulation module. Hence, whenever a row of data from D1 is used in the simulation module, it is deleted from D1 and an associated row of the output data is added to D2. For example, in Figure 3.7, row 2 of the table in D1 is deleted from the table when it is used by the simulation module and the last row in the table of D2 is created (as indicated by dashed lines).

### 3.5.2. Simulation Module

The simulation module is activated upon the request by the process controller module to measure the performance of all DRs for the past scheduling periods (using the realized scheduling period data) and to generate new training sets for the learning module. In this way, the quality of the DRs used for the past periods is also assessed.

In Figure 3.8, scheduling period-k (one of the past scheduling periods) is simulated for all *m* DRs. Previous historical data stored in the D1 are used to generate input to simulation experiments. All *m* DRs are simulated one by one and their corresponding average tardiness values (f(DRj)) are measured. Then, the DR that results in the minimum average tardiness value (DRj = DR[argmin{f(Dri), i = 1,2,3,…,*m*}]) is identified as the best DR for scheduling period-k. Note that this rule may not be the same rule used previously for period-k. Running the simulation module for past periods and collecting the performance data help us to create training sets for the learning module. Hence, the best rule identified in the simulation experiments and the corresponding manufacturing conditions are stored in D2 of the database in the form of instance data (see Figure 3.8).

### 3.5.3. Learning Module

The learning module is mainly composed of two parts: “learning module-1” and “learning module-2”. Their functionalities are given below:

**Learning Module-1:** This module contains the learning tree that is constructed by the learning algorithm in learning module-2. Its responsibility is to select a new DR from the existing learning tree based on the current values of the system state attributes. The on-line controller module provides the current values of these attributes to learning module-1 and requests a new DR. In response, module-1 recommends the best DR to the on-line controller (Figure 3.9).

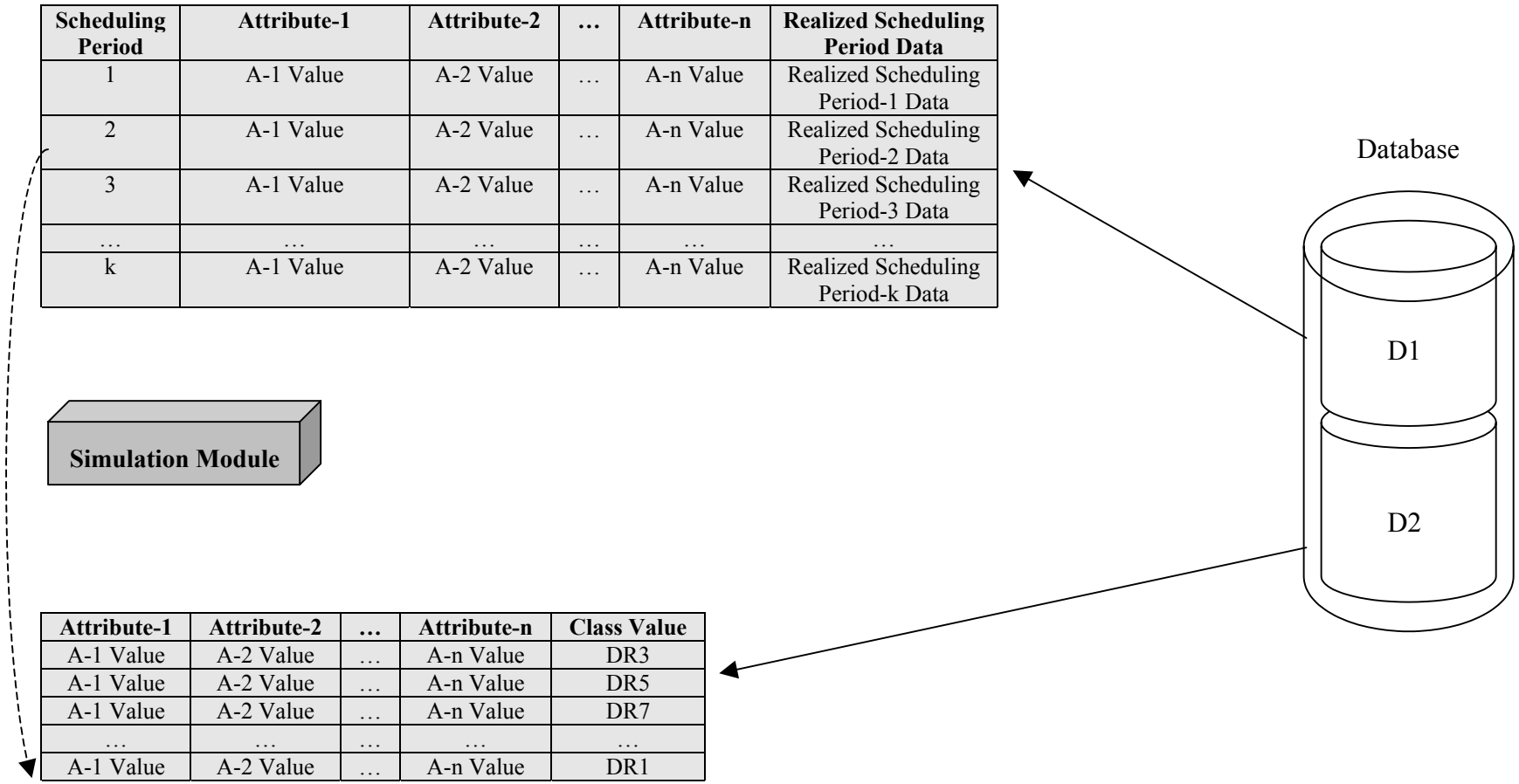



Figure 3.7: Database of the Proposed System

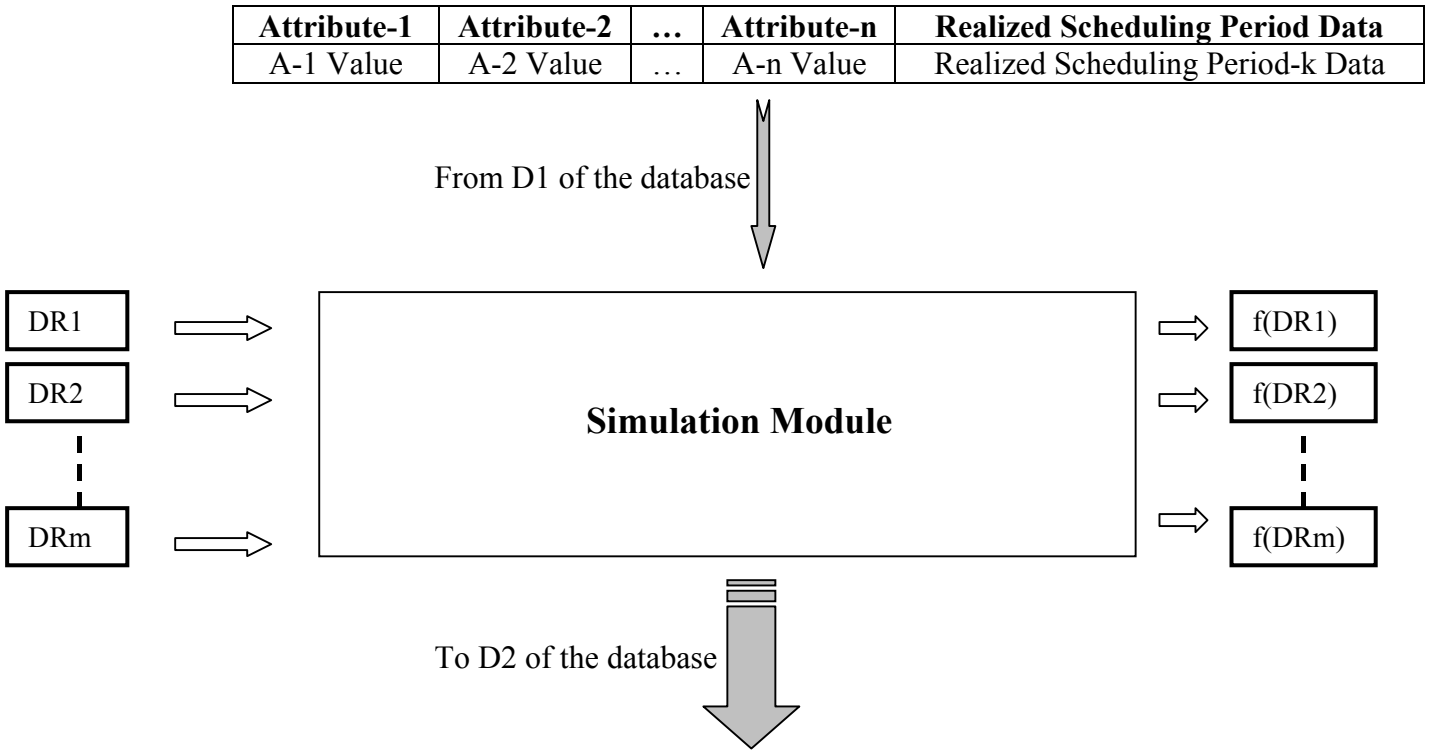



Figure 3.8: Simulation Module

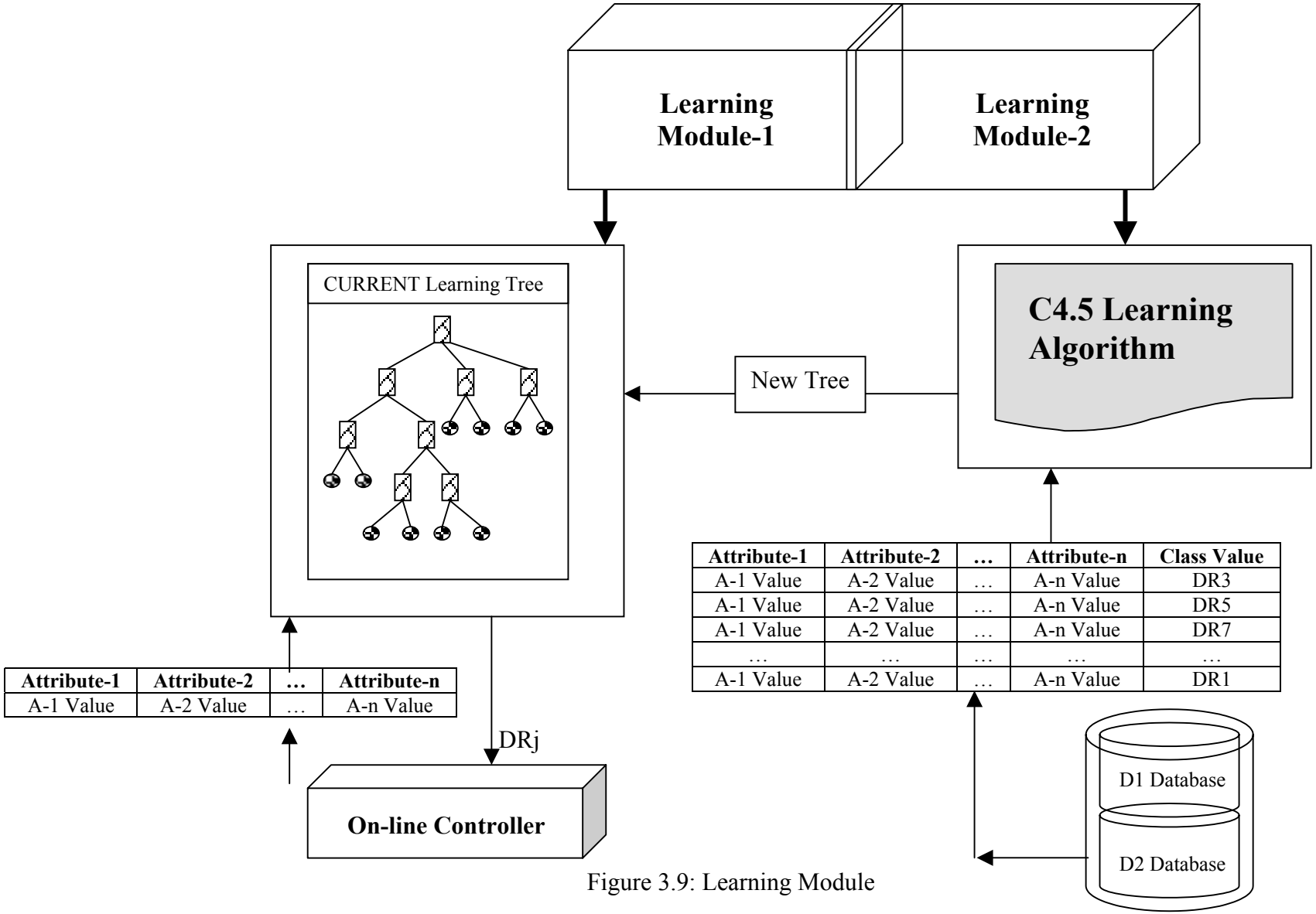



Figure 3.9: Learning Module

**Learning Module-2:** This module contains the learning algorithm that is used to generate the learning tree in learning module-1. As seen in Figure 3.9, the algorithm is invoked by the process controller module and the necessary data (instance data) is retrieved from the D2 database. C4.5 algorithms (Quinlan, 1993) are used to create the learning tree (see Figure 3.9).

### 3.5.4. On-line Controller

As discussed in Section 3.2, there are mainly two responsibilities of the on-line controller. These are as follows:

**i) Handling the realization of a scheduling decision**

Realization of a scheduling period is accomplished by the implementation of a scheduling decision (i.e., implementation of a dispatching rule) in either a real manufacturing system or a simulated environment. In this study, we use the second approach and run the internal simulation engine (see Figure 3.10). To get a realization of a scheduling period, on-line controller requests a DR from the learning module and implements it. The results of implementation in the form of realized scheduling period data is sent to D1 of the database.

**ii) Monitoring the real system for new rule selection symptoms**

Detecting new rule selection symptoms (see Table 3.2 on page 20) and taking the appropriate actions in response to the existence of these symptoms is another functionality of the on-line controller module. As discussed in Section 3.3.2 (Figures 3.2 and 3.3 in particular), there are two new rule selection symptoms (BSP and MP). Whenever the on-line controller module detects any one of these two symptoms during the realization of a scheduling period, it pauses the execution process and requests a new DR from the learning module. Upon the new DR supplied by the

learning module-1, on-line controller resumes the execution process with this new DR (see Figure 3.10).

### 3.5.5. Process Controller

Process controller coordinates the operations of the other modules. It takes its necessary inputs from the on-line controller and activates the other three modules appropriately. It has three sub-modules: process control chart constructor, process control chart, and logical controllers (see Figure 3.11). These are explained in detail as follows:

**i) Process Control Chart Constructor**

The purpose of this sub-module is to update the process control charts ($\overline{X}$ and R charts), which are responsible from the control of the learning tree. The construction of the process control charts requires data ($X_i^*$) from D2 (the construction methods of these two charts are given in Appendix A).

The $\overline{X}$ chart is used to detect the shifts in the mean performance of the decisions (selected DRs) given by the learning tree. Averages of the average tardiness values are plotted in this chart. R chart is used to detect the shifts in the variance of the performance of the decisions of the learning tree (see DeVor *et al.*, 1992). In other words, standard deviations of the average tardiness values of the realized scheduling periods are plotted in this chart.

**ii) Process Control Chart Sub-module**

This module contains the process control charts, which are created by the process control chart constructor module (Figure 3.11). The purpose of this module is to handle the monitoring operation of the learning tree by using these two charts ($\overline{X}$ and R charts).

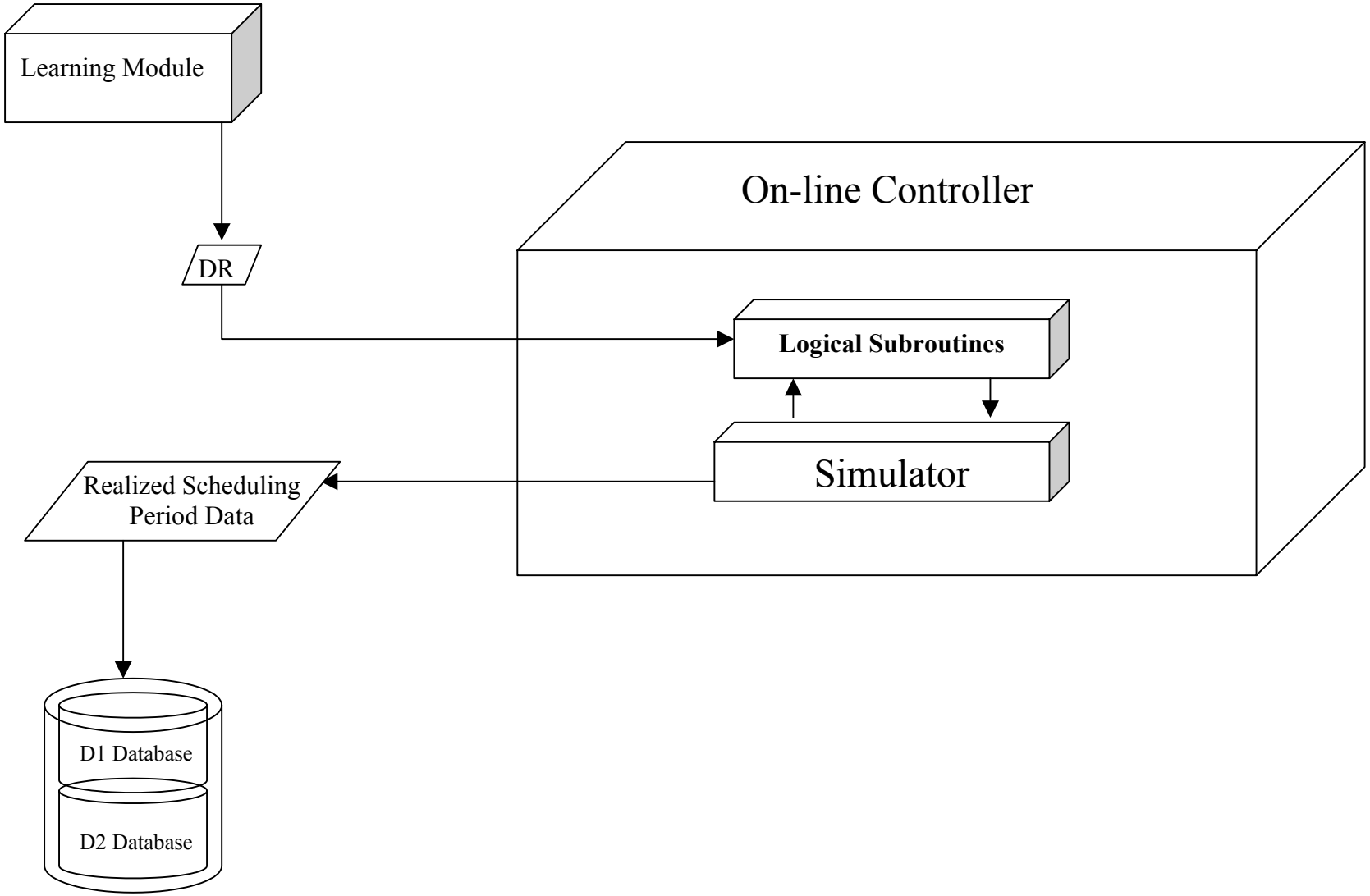



Figure 3.10: On-line Controller

One of the distinguishing features of the proposed scheduling system from the previous studies (Suwa *et. al.*, 2003 and Shaw *et. al.*, 1992) is the mechanism that continuously updates the learning tree. This continuous update is important since the manufacturing system often undergoes various types of changes in time. In this context, the process control charts act as a regulator of the learning tree. Moreover, the process control charts may also need to be updated due to changes in manufacturing conditions. Hence, as the proposed system evolves over time, two important decisions need to be made:

**Decision-1:** Is it necessary to update the existing learning tree at current time *t*?

**Decision-2:** Is it necessary to update the existing process control charts at current time *t*?

These two questions are to be answered every time when a new data point is plotted in the process control charts ($\bar{X}$ and R charts) and the decisions are made by the rules defined in the logical controllers of the process controller module. These rules are defined in the next section. In this section, however, we focus only on the data plotted on the process control charts. Recall that the data plotted on the $\bar{X}$ and R charts are obtained from the on-line controller (the $\bar{X}_i$ data) but the data used to update the charts are supplied from the D2 database.

In Figure 3.12, we illustrate the data points plotted on the $\bar{X}$ chart. The horizontal axis represents the time and the vertical axis is the average tardiness (i.e., performance measure). When the system continues, the $Y_i$ values (average tardiness per scheduling period) are collected by the on-line controller at the end of each scheduling period. These observations are then grouped in size 5 to create $X_i$'s (average of average tardiness).

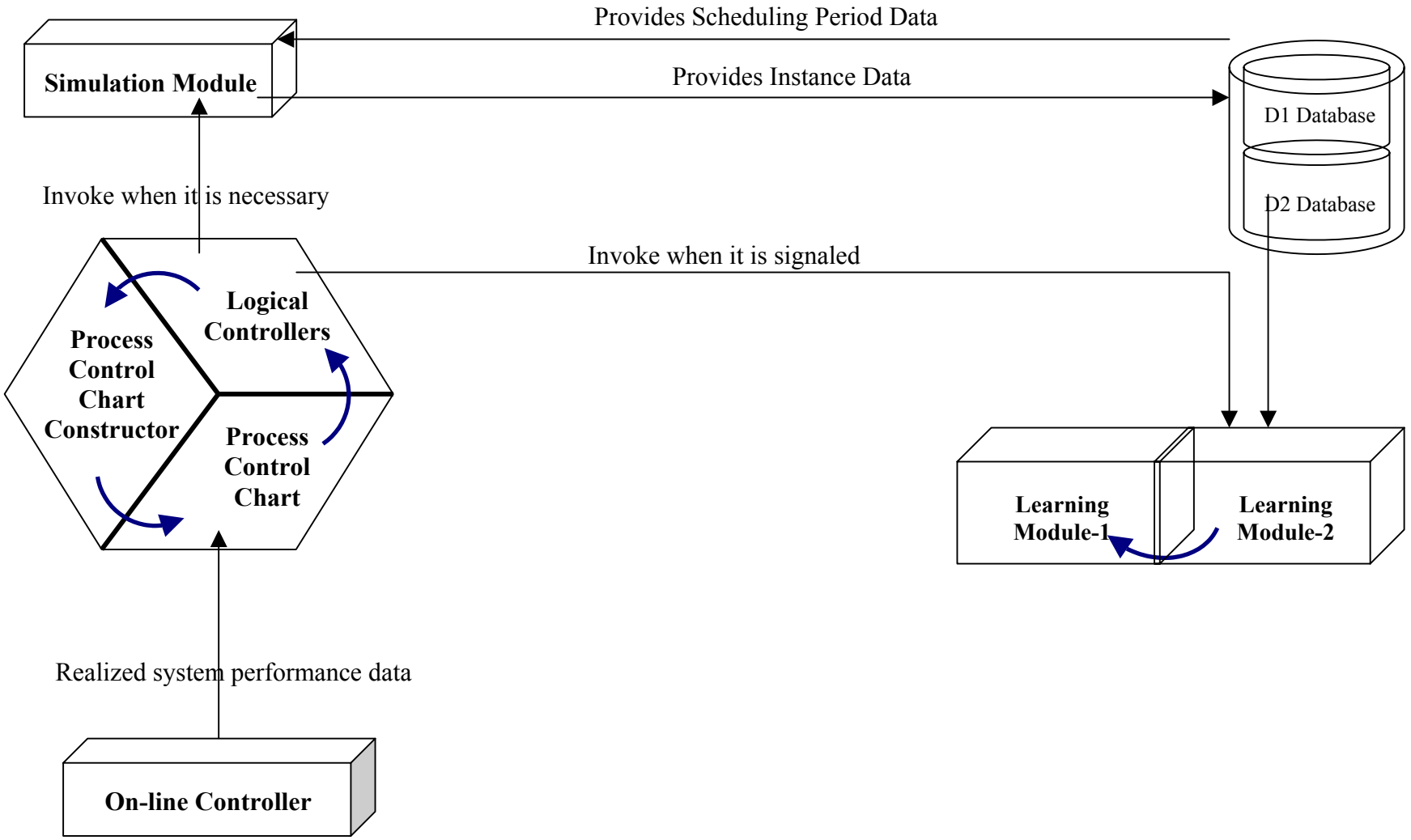



Figure 3.11: Process Controller Module and Its Relationships with Other Modules

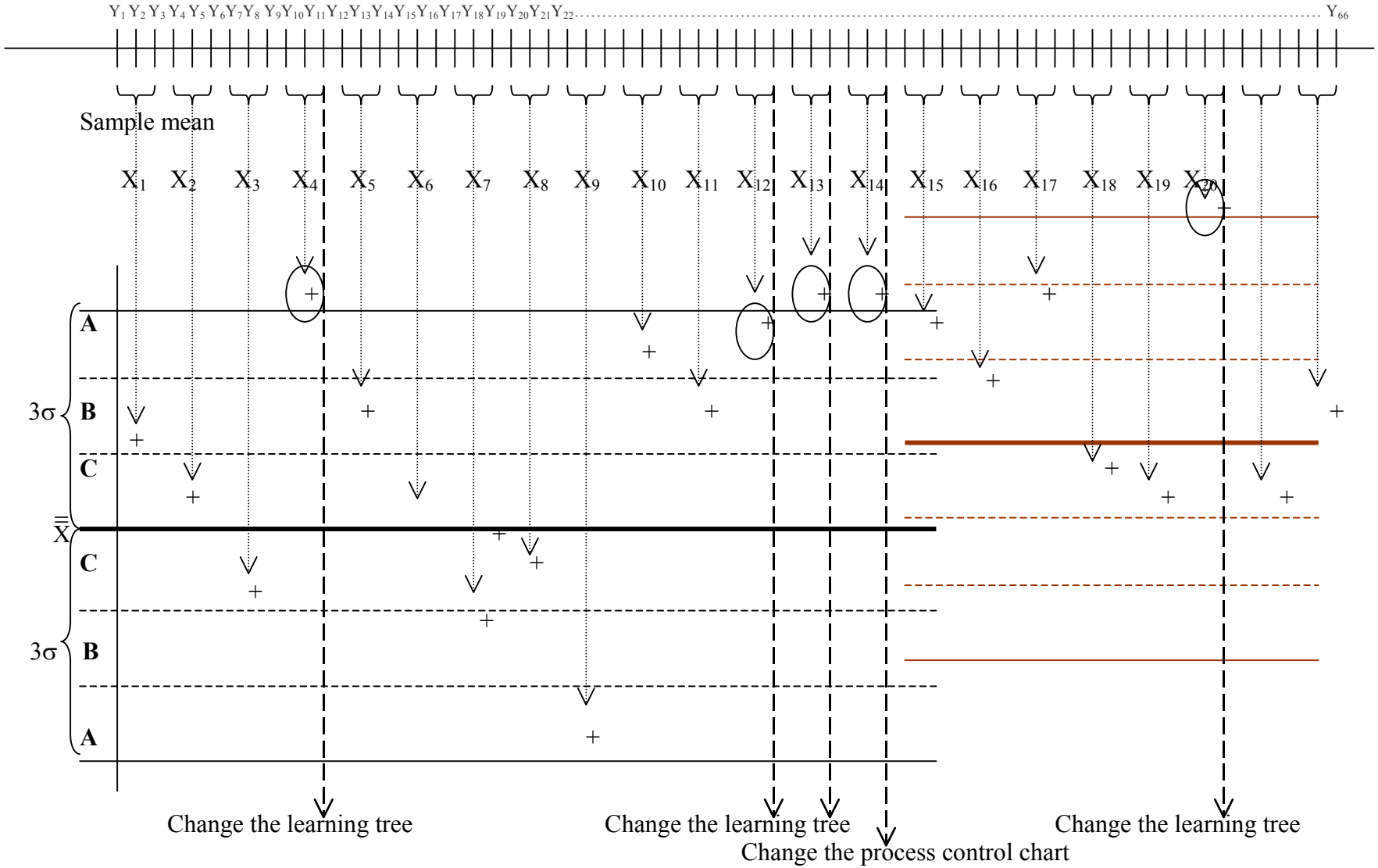



Figure 3.12: Plotted Data in the $\bar{\bar{X}}$ Chart

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

This is the same as the R chart in which the following $R_i$ values are plotted:

$$R_i = Y_{largest} - Y_{smallest} \quad \text{where } Y_{largest}, Y_{smallest} \in \left(Y_{i-2}, Y_{i-1}, Y_i, Y_{i+1}, Y_{i+2}\right)$$

There are two reasons for grouping five $Y_i$ values to generate $X_i$ and $R_i$ values for the $\bar{X}$ and R charts, respectively. First of all, the $\bar{X}$ and R charts require the normality assumption. As discussed in DeVor, Chang, Sutherland (1992, pp. 197-198), we satisfy the normality assumption for $\bar{X}_i$'s by grouping observations (a group of size 4-6 is usually recommended in the literature). Another reason for plotting every five observations is to give the current learning tree a chance for survival. In other words, the performance of the learning tree is judged in a reasonable time period without leading to nervousness.

**iii) Logical Controllers**

This part of the system contains predefined logical rules used for updating the learning tree and control charts. These decisions normally affect the entire system by triggering the other modules (i.e., simulation module, learning engine etc.). These rules are as follows:

**<u>Rule set 1:</u> Update Only Learning Tree Rules**

The first set of rules is used to update the existing learning tree. As discussed earlier, the current learning tree may loose its validity in time as its performance is monitored by the process control charts ($\bar{X}$ and R charts). Three rules, called as "update only learning tree rules", are given in Table 3.3. These rules are applied to the control charts ($\bar{X}$ and R charts) and when any of these three signals is detected, the existing learning tree is updated. These rules are provided in most of the statistical quality books to interpret the $\bar{X}$ and R charts (see for example DeVor *et. al.,* 1992).

In Figure 3.12, at $X_i$=$X_4$ an "extreme points" signal is detected and the current learning tree is updated at this point. Also, "zone-A signal" is detected at $X_i$ = $X_{12}$ and the learning tree is updated at this point again.

**<u>Rule set-2:</u> Update "Both the Learning Tree and the Process Control Charts" Rules**

The second set of rules defined in this module is used to give the update decision of the process control charts as well as the learning tree. As discussed previously, the process control charts may lose their validity in time since the manufacturing system conditions may change.

Table 3.3: Update ONLY Learning Tree Rules

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

In Table 3.4, two rules are defined to update both the learning tree and the process control charts. Whenever one of these two rules in Table 3.4 applies, a new learning tree is created and the process control charts ($\bar{X}$ and R charts) are updated. For example, in Figure 3.12, $X_i$=$X_{14}$ is captured by the second rule in Table 3.4, and therefore the learning tree as well as the control charts are updated at this point. The new control chart shifts upwards in terms of its centerline and control limits (see Figure 3.12).

The first rule in Table 3.4 is adapted from the literature (see DeVor *et. al.,* 1992) and the second one is developed in this research. Both of the two rules indicate a shift in the mean and/or variance of the process (manufacturing system).

Table 3.4: Update Both the Learning Tree and the Process Control Charts Rules

| Signal | Definition | Apply to | Action |
|---|---|---|---|
| 8 successive points | 8 or more successive points strictly above or below the centerline | $\bar{X}$ and R charts | Update both the learning tree and the process control charts |
| 2 successive signal from Rule Set-1 | Two successive occurring of "Update Only Learning Tree Signals" | $\bar{X}$ and R charts | Update both the learning tree and the process control charts |

## 3.6. System Attributes for Job Shop Scheduling System

The learning module of the system generates a learning tree that relies on the manufacturing system characteristics. Decisions on selecting dispatching rules are given by the existing learning tree on-line. In such a system, the learning algorithm requires a number of attributes that can provide valuable information about the current manufacturing system conditions. These attributes, therefore, play a key role in the performance of the proposed system, since they impact the quality of the tree in the construction phase as well as in the decision phase (i.e., selection of the right DRs from the learning tree for a scheduling period). Hence, appropriate attributes should be defined and used in such a way that they can represent a variety of important manufacturing system characteristics. In this section, the attributes defined for a job shop manufacturing environment are discussed. Detailed definitions of the proposed attributes are provided in Appendix B. Hence, our discussion in the rest of this section

will focus on the guidelines that we follow when defining those attributes rather than getting into the details of individual attributes.

First of all, we are faced with two critical and dependent questions (i.e., second question depends the first one). The first question is about defining high quality attributes, which are capable of capturing the important characteristics (information) of the manufacturing system. On the other hand, the second question is about deciding on a subset of these predefined attributes, which are to be embodied into our system. This subset of attributes should be selected in such a way that they should work in harmony and each individual attribute should capture some portion of the important information about the manufacturing system. In the following paragraphs, we address the first question and present our approach to that question. We leave the discussion and the results of the second one to the next chapter.

At the very first step of defining the candidate attributes, we realize that it is important to define attributes so that their values can be calculated easily. This is because of the fact that our proposed system is an on-line scheduling system, and hence the time required to select a new dispatching rule should be negligible. Moreover, when setting the values of the attributes at any time $t$, all we can use is the available information at that time such as the number of jobs in the system, processing times and due dates of the jobs, the realized performance of the system in the last scheduling period etc. Based on these observations, we define a number of attributes such as *total remaining processing time*, *maximum queue length at time t*, *average remaining time until due dates* and so on. In addition to the above observations, we also take into account the characteristics and dynamics of the candidate dispatching rules and try to figure out under what conditions a specific rule performs well and in what other conditions performs poorly. In light of this idea, we define a number of

attributes for each dispatching rule that might be helpful to differentiate that rule from the others. For instance, Attribute-12 (NumLongPT), which is the number of jobs with higher processing times than the average processing time of all jobs, is defined to distinguish SPT rule from the others. The idea behind this is as follows: if there are so many jobs with high processing time requirements, then the probability that the new arriving jobs with less processing time requirement than these jobs will be higher. This implies that these jobs, which have long processing times, will most probably be scheduled too late under SPT, resulting with a high average tardiness value. Second half of the attributes given in Appendix B (i.e., Attribute-12 and the rest) are defined in a similar fashion as we just discussed.

## 3.7. Dynamics of the Learning Algorithm

In Section 3.5.3, when we discuss the learning module, we say that the module is composed of two parts. One of these parts, which we call module-1, contains the learning tree and selects a new DR from this learning tree based on the current values of the system state attributes. This learning tree, on the other hand, is created by the second part, which we call as module-2. In this section, we present the internal dynamics of module-2. Also, to illustrate the concepts, we will give a simple example and show the construction of a learning tree step by step.

As we already mentioned, module-2 employs the C4.5 algorithms developed by Quinlan (1993) to create the learning tree. The fundamental feature of the algorithm is that it uses divide-and-conquer approach. That is, it divides the data set on the attributes' values at each branching and deals with the subsets of data. The main steps of the algorithm are as follows:

**Step 0.** (Initialization) Let the training data set be *C*.

**Step1.** If all the instances in *C* belong to the same class, then create a node with that class value and halt. Otherwise, go to Step 2.

**Step 2.** Select an attribute, *A* with values $\{v_1, v_2, v_3, ..., v_n\}$ and create a decision node.

**Step 3.** Partition the training instances in *C* into subsets *C1, C2, ..., Cn* according to the values of *A*.

**Step 4.** Apply the algorithm recursively to each of the sets *Ci*.

The algorithm stops when all the instances are perfectly classified or when there is no remaining attribute for further branching. Since a previously used attribute, for branching at a particular node, is not used for further branching for the successor nodes, termination of the algorithm is guaranteed.

The most challenging part of the algorithm is how to decide (or pick) the attribute to partition the instance data at some node. For example, let the data set in Table 3.5 is given. Note that, the definitions of the attributes used in this artificial data set are given in Appendix B. When the algorithm starts, it should pick an attribute among the four attributes for the first branching. But how does C4.5 decide which attribute is the best for branching at a given node? A statistical property, called *information gain* is used by C4.5. Gain measures how well a given attribute separates training examples into targeted classes. The one with the highest information (information being the most useful for classification) is selected. In order to define the gain function, an idea from information theory, which is called as *entropy (*or *information)*, is used. Entropy of a set is the average amount of information needed to identify the class of an instance in that set. The entropy is calculated as the following:

Let *S* be a set of instances and let *C* be the set of possible class values. In our example, *C*={SPT,ODD}. Then,

*Entropy(S)* = *Info(S)* = $\sum_{c \in C} -\frac{|S_c|}{|S|}\log_2(\frac{|S_c|}{|S|})$ , where $S_c$ is the set of instances that belongs to class *c*, $c \in C$ and $|*|$ is the cardinality of the set *.

Table 3.5: Artificial training data set

| **Attributes** | | | | **Class** |
|---|---|---|---|---|
| Discrete type | Continuous type | Continuous type | Discrete type | |
| **RUS** | **NumCust** | **PCompPT** | **NumLongPT** | **DR** |
| 0 | 75 | 70 | 1 | SPT |
| 0 | 80 | 90 | 1 | ODD |
| 0 | 85 | 85 | 2 | ODD |
| 0 | 72 | 95 | 2 | ODD |
| 0 | 69 | 70 | 2 | SPT |
| 1 | 72 | 90 | 1 | SPT |
| 1 | 83 | 78 | 2 | SPT |
| 1 | 64 | 65 | 1 | SPT |
| 1 | 81 | 75 | 2 | SPT |
| 2 | 71 | 80 | 1 | ODD |
| 2 | 65 | 70 | 1 | ODD |
| 2 | 75 | 80 | 2 | SPT |
| 2 | 68 | 80 | 2 | SPT |
| 2 | 70 | 96 | 2 | SPT |

For example, if *S* is the data set given in Table 3.5, then the *Entropy(S)* is:

*Entropy(S)* = $-\frac{5}{14}\log_2(\frac{5}{14}) - \frac{9}{14}\log_2(\frac{9}{14}) = 0.940$

Then, *Gain(S,A)* is the information gain of set *S* if it is partitioned on attribute *A*. The following formulation assumes that attribute *A* has discrete values. That is, there is a set *V* of possible values that A can take. We also explain the case of continuous valued attributes just after the following discussions.

Let *V*={*v1, v2, ..., vn*} is the set of all possible values of attribute *A*.

$Gain(S,A) = Entropy(S) - \sum_{v \in V} \frac{|S_v|}{|S|} Entropy(S_v)$, where $S_v$ is the subset of $S$ for which attribute $A$ has a value $v$.

For example, let S is the data set given in Table 3.5 and we want to find the gain for the attribute RUS. Note that, the attribute RUS is a discrete type attribute, which can take only three values: 0, 1 and 2. Then, the gain of set $S$ if it is partitioned on the attribute RUS will be:

$$Gain(S,RUS) = 0.940 - \left[ \frac{5}{14}\left\{ -\frac{2}{5}\log_2(\frac{2}{5}) - \frac{3}{5}\log_2(\frac{3}{5}) \right\} + \frac{4}{14}\left\{ -\frac{4}{4}\log_2(\frac{4}{4}) - \frac{0}{4}\log_2(\frac{0}{4}) \right\} + \frac{5}{14}\left\{ -\frac{3}{5}\log_2(\frac{3}{5}) - \frac{2}{5}\log_2(\frac{2}{5}) \right\} \right] = 0.940 - 0.694 = 0.246$$

It might seem that tests on continuous attributes would be more complicated, since they contain arbitrary thresholds. However, this is not the case. The following algorithm solves the problem for continuous attributes and is used in C4.5 and most of the other learning algorithms. The algorithm is the following: the instances in the training data set are first sorted on the values of the continuous attribute, say $A$. Let's denote the sorted values as $\{v_1, v_2, v_3, ..., v_n\}$. Any threshold value between $v_i$ and $v_{i+1}$ will have the same effect of dividing the cases into those whose value of the attribute $A$ lies in $\{v_1, v_2, \ldots, v_i\}$ and those whose value is in $\{v_{i+1}, v_{i+2}, \ldots, v_n\}$. Thus, there are n-1 possible splits on $A$. For each split, the gain function is calculated and the maximum of these n-1 gain values is taken as the gain on that attribute with its associated threshold value.

Now, let's return to our example and construct the learning tree by using the algorithm we discussed. We will choose the first attribute to partition the initial data set. For this, we calculate gain values for each of the attributes by using the above formulas. Gain values of each attribute are as follows:

*Gain*(*S,RUS*) = 0.246

*Gain*(*S,NumCust*) = 0.113

*Gain*(*S,PCompPT*) = 0.102

*Gain*(*S,NumLongPT*) = 0.048

Since we get the maximum gain value from the attribute RUS, we divide our initial data set on RUS and we get the partial tree in Figure 3.13. For the first branch, RUS = 0, we have a subset of instances, say $S_1$, which is shown in Table 3.6. We calculate gain for the other three attributes on this data set. The gain values are found to be:

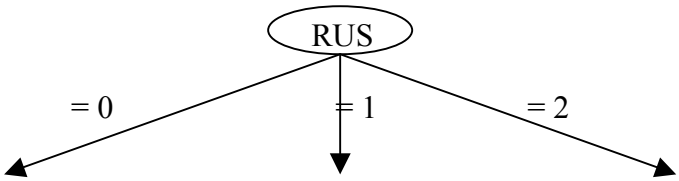



Figure 3.13: Construction of the learning tree: first step

*Gain*($S_1$,*NumCust*) = 0.419

*Gain*($S_1$,*PCompPT*) = 0.970

*Gain*($S_1$,*NumLongPT*) = 0.019

Therefore, the second division attribute is PCompPT with a threshold value of 70 and the next partial tree is shown in Figure 3.14. For the second branch, RUS = 1, the subset obtained is perfectly classified and therefore both gain values are found to be zero. Thus, we create a leaf node with associated class value (Figure3.15). If we continue to proceed with the algorithm until it stops, we will get the final learning tree as shown in Figure 3.16. This tree perfectly classifies all the instances and therefore the algorithm stops without attempting to divide the sets on the remaining unused attribute NumCust.

Table 3.6: Subset of initial data set for branch RUS = 0

| RUS | NumCust | PCompPT | NumLongPT | Dispatching Rule |
|---|---|---|---|---|
| 0 | 75 | 70 | 1 | SPT |
| 0 | 80 | 90 | 1 | ODD |
| 0 | 85 | 85 | 2 | ODD |
| 0 | 72 | 95 | 2 | ODD |
| 0 | 69 | 70 | 2 | SPT |

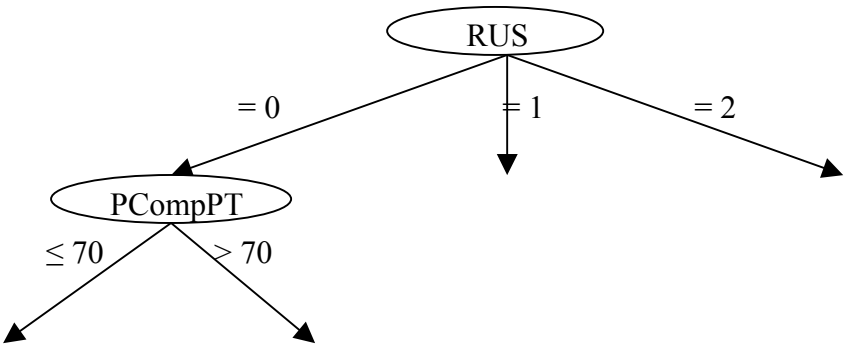



Figure 3.14: Construction of the learning tree: second step

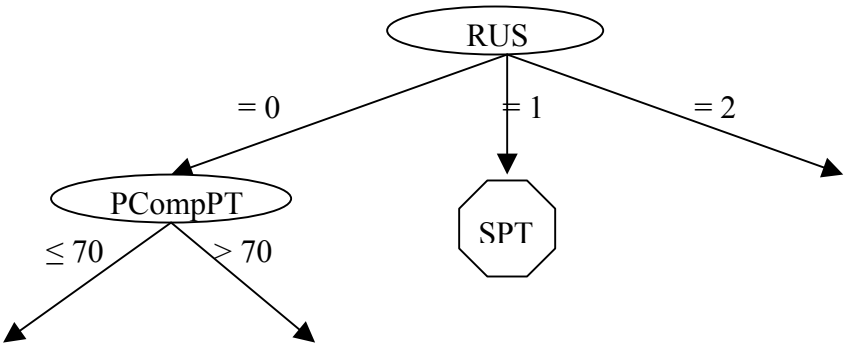



Figure 3.15: Construction of the learning tree: third step

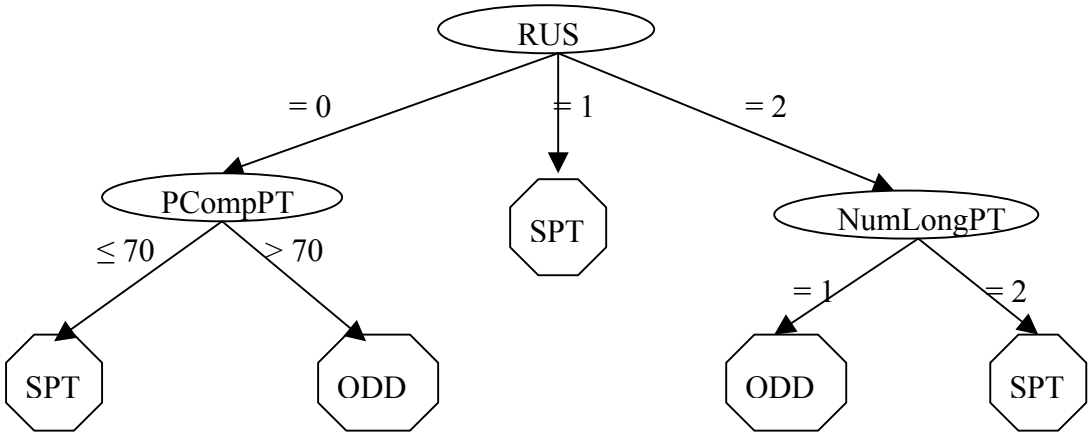



Figure 3.16: Final learning tree

The procedure for constructing the final learning tree is not limited by the above discussion. After the construction of the tree is completed, it is pruned by a pruning algorithm, which is also a part of the C4.5 algorithms. Simply, pruning algorithm eliminates the nodes of the tree that are not found to be significantly important. The details of the pruning procedure are not presented here but it can be found in Quinlan (1993).

## 3.8. Summary

In this chapter, we presented the parts of the learning-based scheduling system and explained the dynamics of the learning algorithm. This scheduling technique uses the dispatching rules where the rules are selected by the learning tree on-line. Also, to monitor changes in system parameters, process control charts are employed. Whenever the control chart signals a change in the system, the machine learning algorithm uses the new available data from the system in order to re-learn about the characteristics of the manufacturing environment in order to give better decisions in the future.

In the next section, we will present the experimental designs to answer various questions. Some of these experiments will aim to fine-tune up the system parameters as well as to provide valuable insights into the job shop scheduling problem. We will also test the performance of the proposed system under different experimental conditions.

# CHAPTER 4

# Experimental Design and Computational Results

In this chapter, we discuss the experimental conditions and present simulation results to measure the performance of the proposed system. In the first section, we give the underlying ideas behind the motivation required to explain the experimental results clearly. In Section 4.2, we set the utilization and due date tightness levels. In Section 4.3, we conduct experiments for the selection of scheduling period length and use the results in Section 4.4 to set the monitoring period length. In Section 4.5, we experiment on our predefined system attributes (see Appendix B) to select an appropriate subset among them, which is to be used in the proposed system. The results of experiments on the learning-based system are organized in two consecutive sections. In Section 4.6, we test the system with a static learning tree, where the learning takes place only at the beginning of the execution and the tree is not updated again in time. The proposed system as a whole is tested in Section 4.7. We end this chapter with a brief summary.

## 4.1. Motivation

In this section, we define our problem, list the assumptions and give important definitions. In general, the following assumptions are valid throughout the thesis unless otherwise stated:

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

In the literature, when a dispatching rule is used for an entire planning horizon, it is called single-pass scheduling. On the other hand, the best dispatching rule (among a candidate rule set) can be determined (via simulation) and used for each relatively short scheduling interval in a planning horizon. This second approach is called multi-pass scheduling.

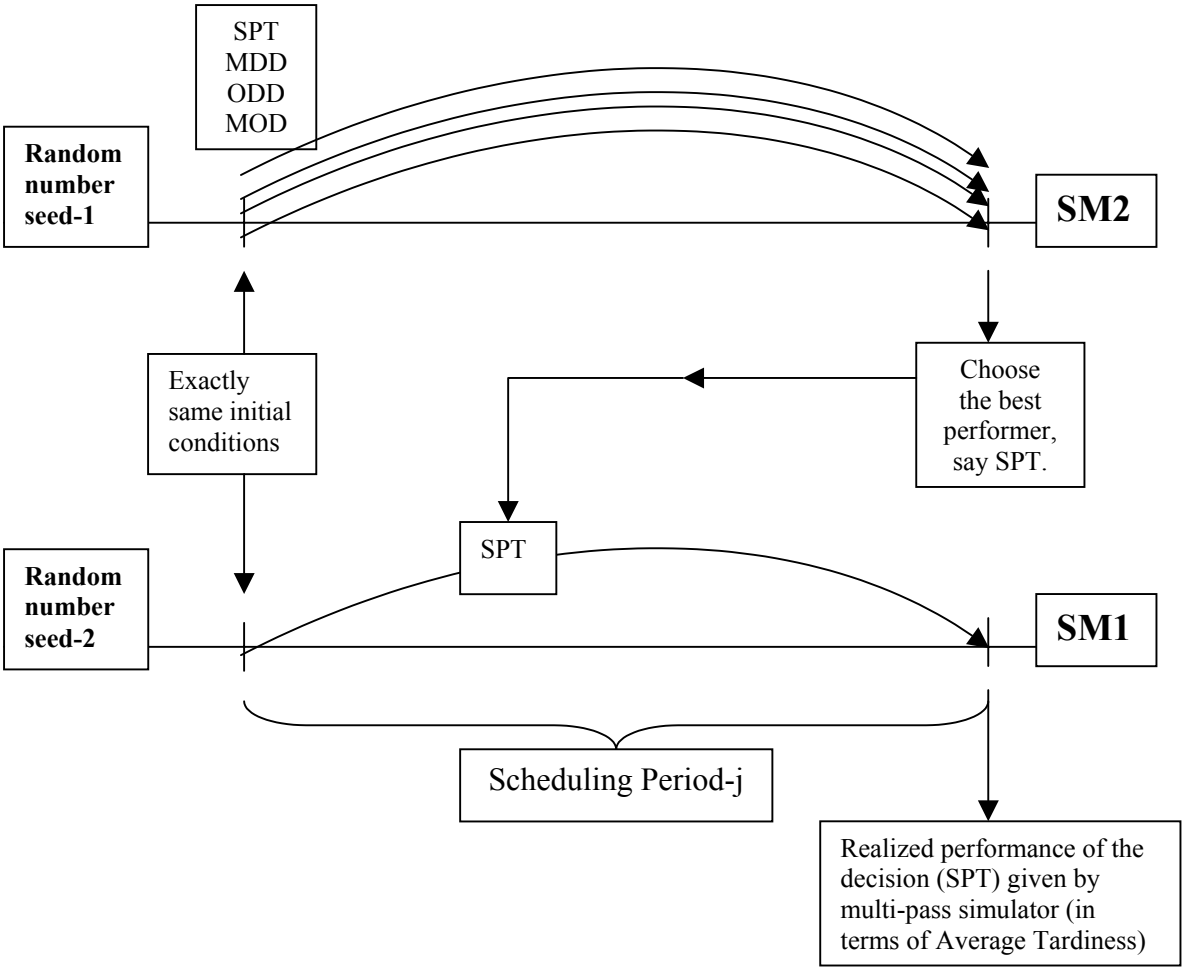
When explaining the experimental results, we use three performance measures, called as *Multi-pass Performance (MultiPass)*, *Best Performance (BestPerf)* and the *Learning Performance (LearnPerf)*. They are all measured in terms of average tardiness and defined in the following paragraphs.

Assume that we have two simulation models of the same manufacturing system, called as SM1 and SM2. SM1 will represent the real life and SM2 will represent the simulation environment, which is the imitation of SM1. Note that, SM1 is the simulation model used in the on-line controller and SM2 is the multi-pass scheduling simulator, which is used to compare the performance of our proposed system with the performance of multi-pass scheduling. Since, random events occurred

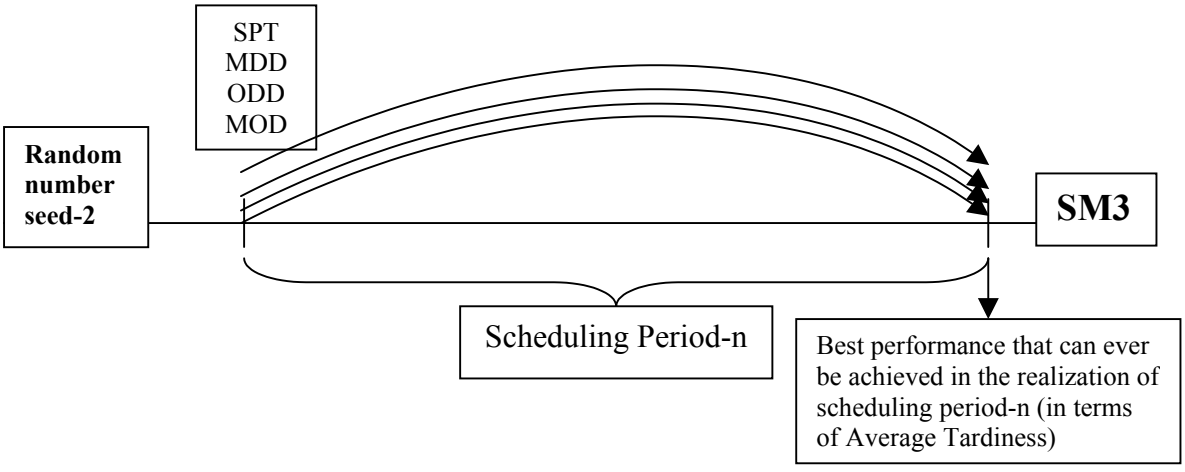
in real life differs from the simulated environment, these two models operate under different random number seeds. Say, SM1 uses random number seed-1 and SM2 uses seed-2. Also, assume a third simulation model, SM3, which also uses the seed-1. We can think of SM3 as the playback of the realized events occurred in SM1 in a scheduling period n. Therefore, SM3 can be run for scheduling period-j only if the realization of period-j in SM1 (real life) is completed. These three simulation models help us to measure the three performances we need.

Figure 4.1-a shows how we measure the *MultiPass* for scheduling period-j. At the beginning of period-j, the system state of SM2 is set equal to the system state of SM1. Then SM2 is run for each candidate dispatching rule (i.e., SPT, MDD, ODD, MOD) and the one resulting with the minimum average tardiness value, say SPT, is selected to be used in scheduling period-j. SPT is passed to the SM1 to realize its actual performance. At the end of scheduling period-j, the realized average tardiness value is our *MultiPass* value for scheduling period-j. In this sense, *MultiPass* is the average tardiness value achieved by the decisions of a multi-pass scheduling simulator.

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

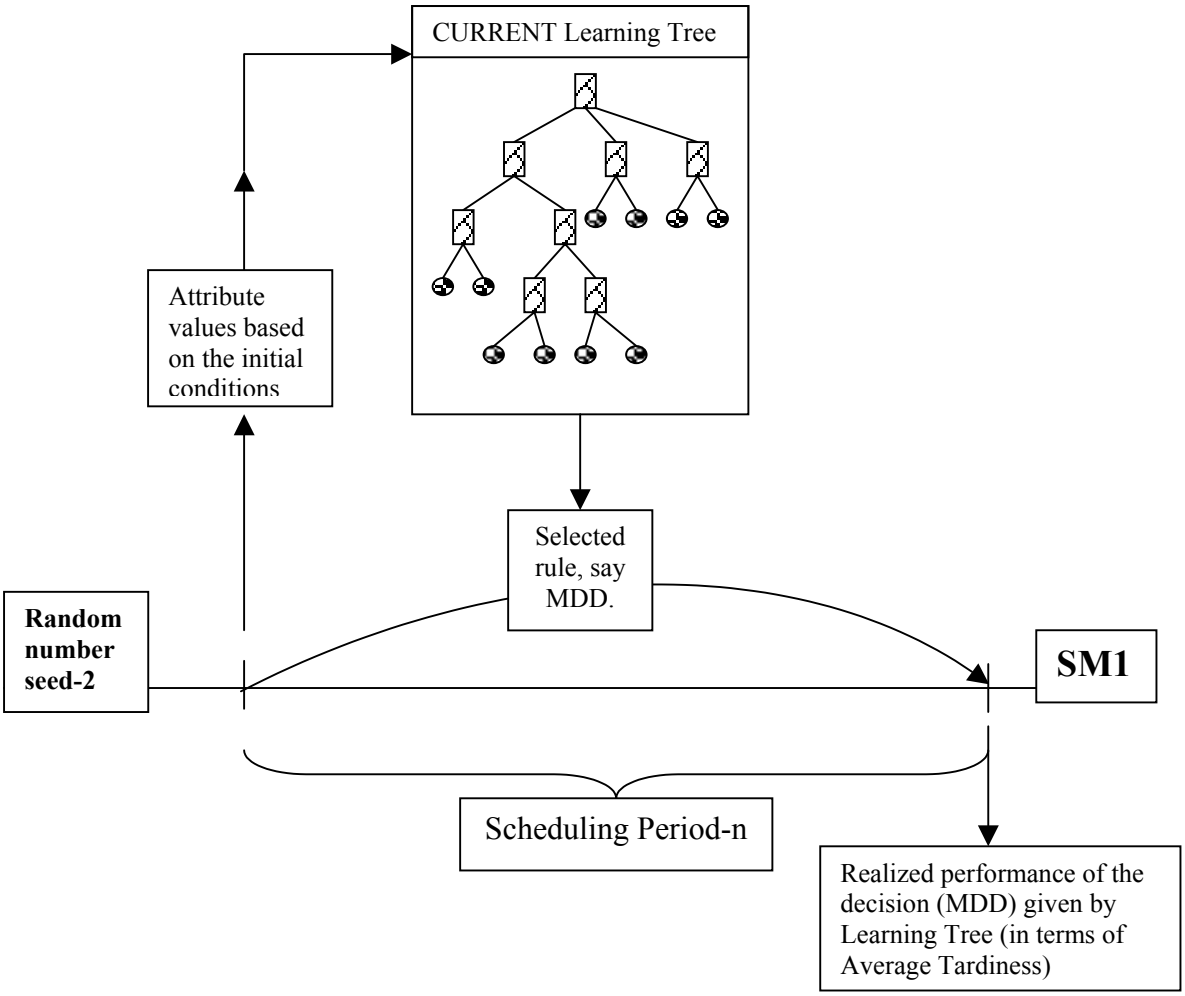



(a)




(b)

Figure 4.1: Performance measures. (a) Determination of Multi-pass performance (*MultiPass*). (b) Determination of Best Performance (*BestPerf*). (c) Determination of the Learning Performance (*LearnPerf*).




(c)

Figure 4.1: (Cont'd)

value of *BestPerf* to the minimum of those tardiness values (see Figure 4.1-b). We measure this performance value to see how far our proposed system's performance (*LearnPerf*) and multi-pass scheduling simulator performance (*MultiPass*) are away from the ideal.

Finally, the learning performance in period-j, *LearnPerf*, is the realized average tardiness value of the rule selected by the learning tree (see Figure 4.1-c). That is, we request a dispatching rule from the learning tree at the beginning of scheduling period-j based on the current values of the system attributes. This rule is used during period-j and the average tardiness value is computed. Since this is the real

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

From these definitions, it is clear that both *LearnPerf* and *MultiPass* should be worse than the *BestPerf*. That is because of the fact that *BestPerf* is the average tardiness value that can be achieved if and only if we know the realization of the random events before selecting the dispatching rules at each scheduling period throughout the planning horizon. Hence, *BestPerf* gives a lower bound for the other two performance functions.

## 4.2. Setting Due Date Tightness Levels and Utilization Levels

In the simulation experiments, two levels of utilization (i.e., low and high) and two levels of due date tightness (i.e., loose and tight) are considered. The two levels of utilization are 80% and 90%. Due dates are set by using the TWK due date assignment rule. The high and low levels are set in such a way that percent of tardy (PT) jobs is approximately as 10% and 40% under the FCFS rule for the loose and tight due date cases, respectively. Table 4.1 summarizes the results of simulation experiments to set the flow allowances. As can be seen in Table 4.1, due date of a job is equal its release time plus 5.5 times its total processing time for 80% utilization and tight due dates case. Note that, the required allowance is almost double when utilization is 90%.

Table 4.1: Simulation results for flow allowances

| **Utilization** | | **Tight** | **Loose** |
|---|---|---|---|
| **80%** | Flow Allowance (k) | **5.5** | **13** |
| | Mean Tardiness (MT) | 4.47 | 0.86 |
| | Percent Tardy (PT) | 41% | 9.6% |
| **90%** | Flow Allowance (k) | **11** | **26** |
| | Mean Tardiness (MT) | 9.03 | 1.76 |
| | Percent Tardy (PT) | 42% | 10% |

## 4.3. Setting Appropriate Scheduling Period Length (SPL)

In this section, our purpose is to properly set the scheduling period length for the proposed system. We consider the following additional assumptions in the experiments:

1. Every scheduling period is of type complete (i.e., no monitoring).
2. Once a rule is selected for a scheduling period, it cannot be changed until the end of that period.
3. The appropriate scheduling period length is determined by looking at the minimum *BestPerf* value.

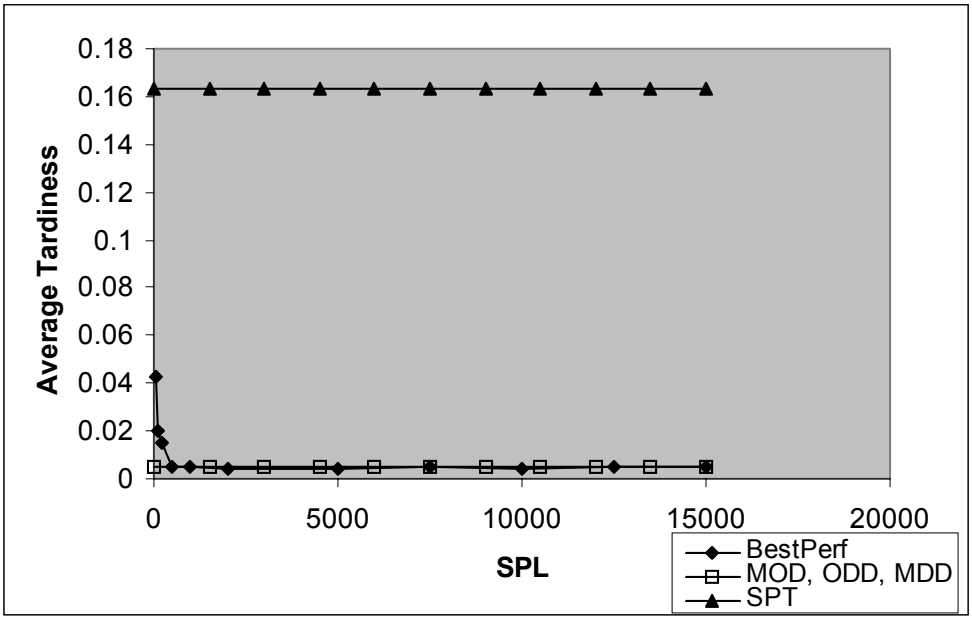
As can be seen in Table 4.2, 11 different levels of scheduling period length are tested in the experiments for four due date and utilization level combinations. The simulation results are taken in steady state with 20 replications each with 200000 minutes of a planning horizon. To find *BestPerf*, scheduling rules are compared under the same experimental conditions using the common random number (CRN) scheme.

Table 4.2: Experimental design of scheduling period length

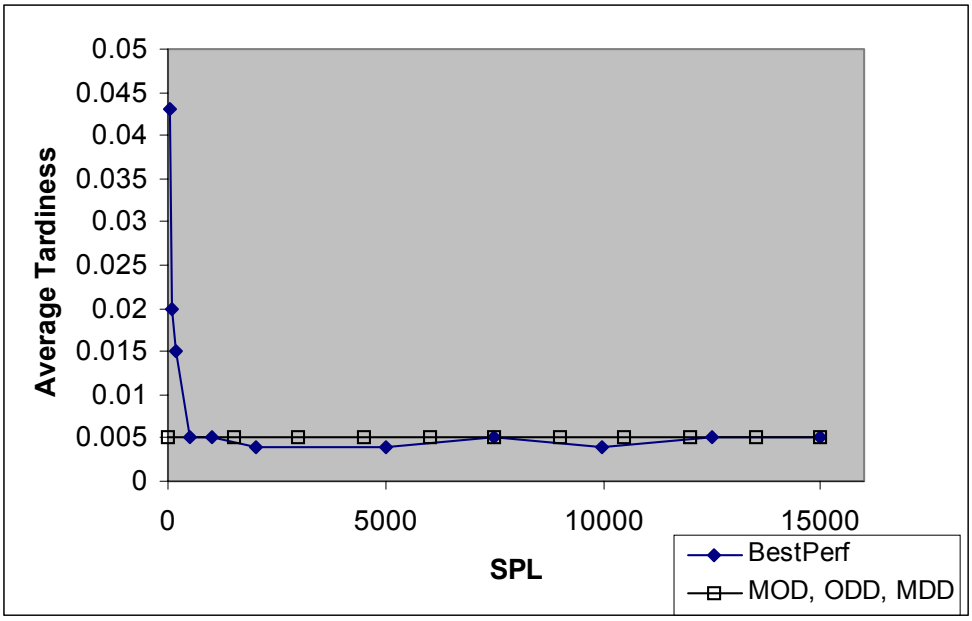
| **Factors** | **Levels** |
|---|---|
| Scheduling period length | 50, 100, 200, 500, 1000, 2000, 5000, 7500, 10000, 12500 and 15000 |

The results for loose due date case are given in Figures 4.2 and 4.3. In these figures, single-pass performances of each dispatching rule are also displayed in addition to *BestPerf*. In both figures, single-pass performance of SPT is found significantly worse than the other rules. The single-pass performances of MOD, ODD and MDD are almost equal. Also, *BestPerf* displays an exponential decay behavior as a function of SPL. It is interesting to note that for short scheduling period length selections, *BestPerf* is found to be significantly worse than the single-pass performances of the three dispatching rules (MOD, ODD and MDD). This is due to the fact that as SPL decreases, even though the selected rules seem to be the best for these short scheduling periods, the system switches to different rules so frequently that the performance of the system in the long run deteriorates. When we increase SPL, *BestPerf* begins to improve and converges to the single-pass performance of the rules MOD, ODD and MDD. Since the performances of the individual dispatching rules (MDD, ODD and MOD) are very close to each other in the long run for loose due-dates (as also stated by Baker, 1984), switching between these rules doesn't provide any benefit. Therefore, *BestPerf* converges to a limit (single-pass performance of the rules) showing a behavior of exponential decay function.

For the tight due dates, the experimental results show different behavior to some extend (see Figures 4.4 and 4.5). In both figures, single-pass performances of each dispatching rule are also displayed in addition to *BestPerf*. It is shown in the figures that the single-pass performances of the four dispatching rules are significantly different than each other and MOD performs at least twice better than the other rules. In addition, *BestPerf* displays an exponential decay behavior as we increase SPL, but having a minimum value at some point. For example in Figure 4.4, *BestPerf* reaches its minimum value at point A (i.e., SPL equals to 1000 minutes) and




(a)




(b)

Figure 4.2: 80% utilization, loose due-dates with CDR set {MOD, SPT, MDD, ODD}. (a) Complete display of the results. (b) Zoom-in version.

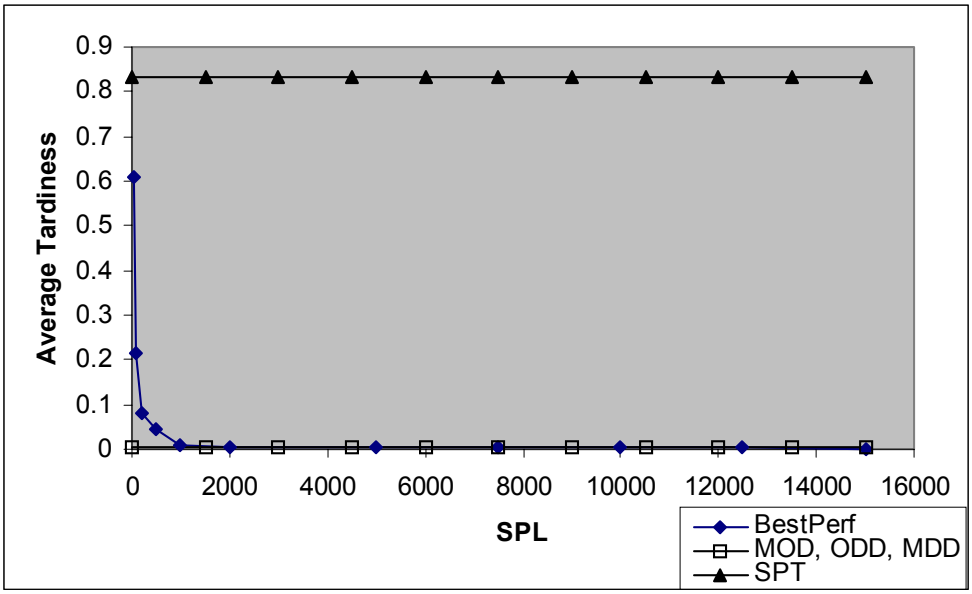
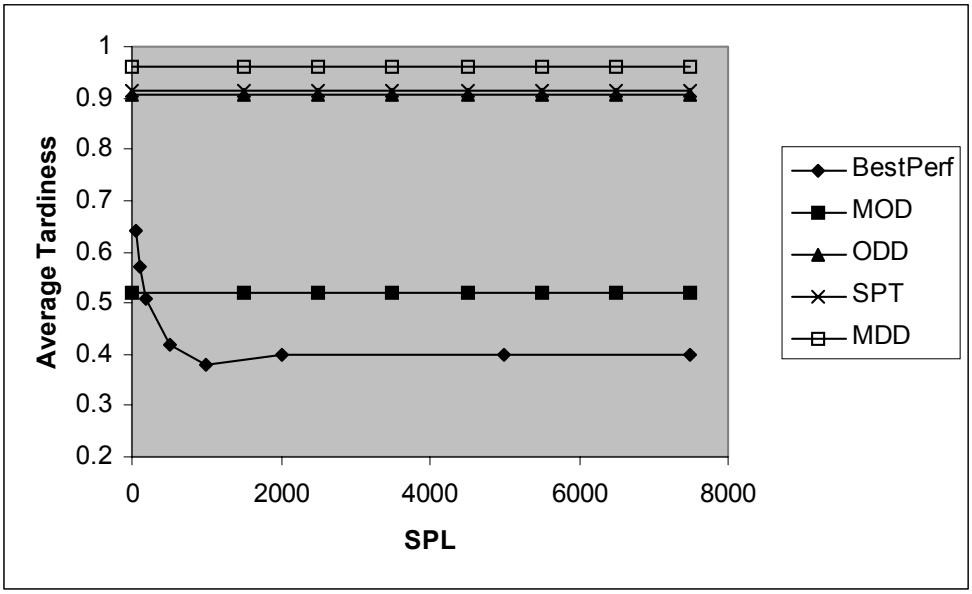


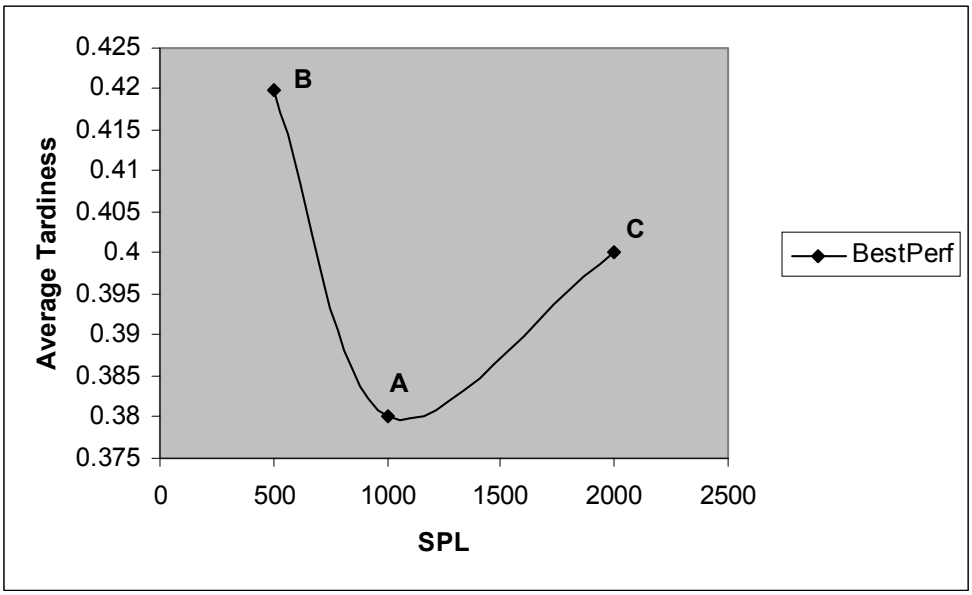

Figure 4.3: 90% utilization, loose due-dates with CDR set {MOD, SPT, MDD, ODD}.

then it begins to deteriorate and converges to a limiting value when we further increase the scheduling period length. We explain this interesting behavior as follows: choosing a shorter scheduling period length results in misdetection of the best dispatching rule for the sake of better long-run performance of the system (i.e., system switches between rules frequently). When we increase the scheduling period length, system begins to select the best rule combination and *BestPerf* reaches its minimum. But, when we continue to increase the scheduling period length further, performance deteriorates and converges to a higher value than the minimum. This higher value is close to the long-run performance of the most dominant dispatching rule, because system begins to choose that rule most of the time. Thus, this significant increase in system performance is attributable to the loss of the improvements that can be achieved by switching to different rules during those long scheduling periods.

We also check whether the minimum points achieved in the tight due date case are statistically significant or not. Figure 4.6-b shows the magnified portion of Figure 4.3-a around the minimum point. We say that the point A is statistically smaller than

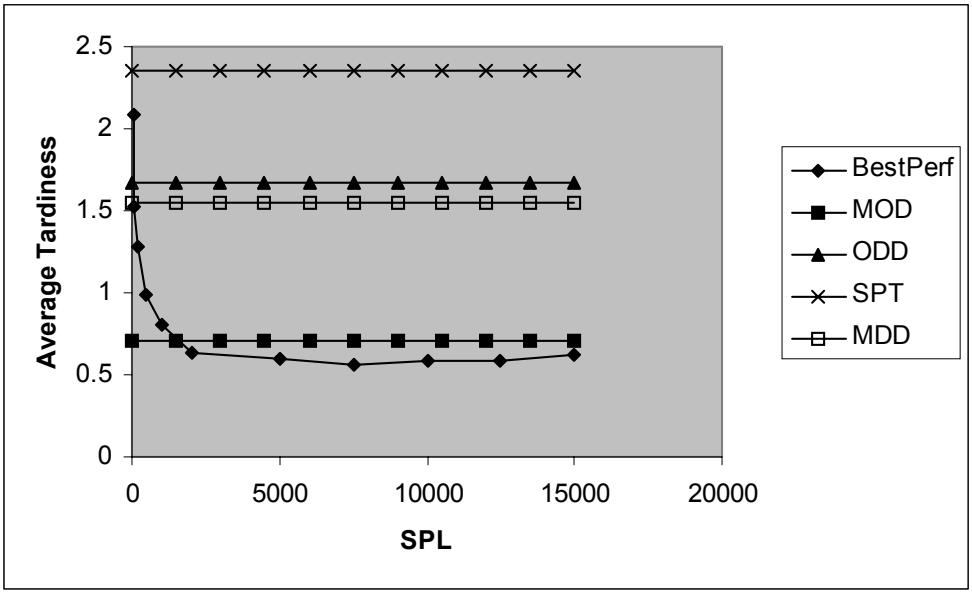



(a)

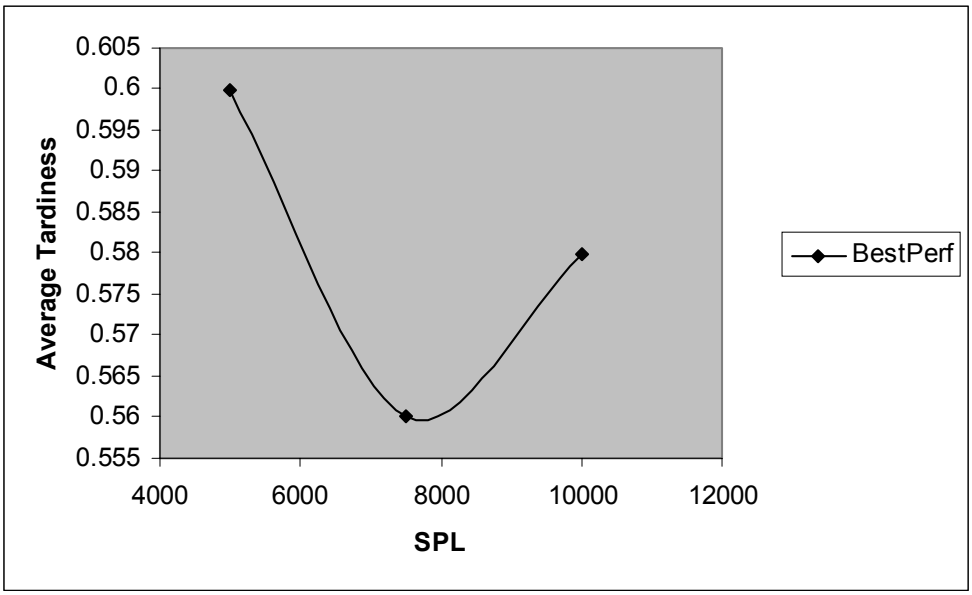



(b)

Figure 4.4: 80% utilization, tight due-dates with CDR set {MOD, SPT, MDD, ODD}. (a) Complete display of the results. (b) Zoom-in version around point A.




(a)




(b)

Figure 4.5: 90% utilization, tight due-dates with CDR set {MOD, SPT, MDD, ODD}. (a) Complete display of the results. (b) Zoom-in version around SPL 7500.

the two neighboring points (i.e., points B and C). This is testified by the paired-t test applied to difference of points AB and AC. For example, when the Confidence Interval is constructed on the difference between A and C, it is

$$\bar{Y}_1 - \bar{Y}_2 \pm t_{1-\alpha/2,2n-2}\sqrt{\frac{S_1^2 - S_2^2}{n}} = 0.40 - 0.38 \pm 2,021\sqrt{\frac{0.03^2 + 0.03^2}{20}} = (0.001, 0.039)$$

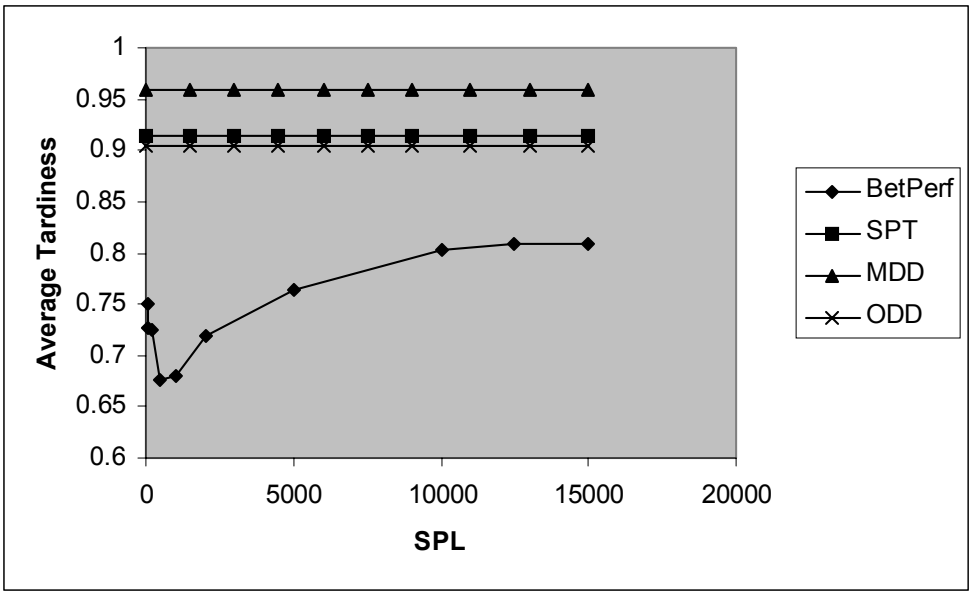
Since it does not include 0, we say that it is statistically smaller on 0.95 confidence level.

From Figures 4.4 and 4.5, we realize that the performances of the three dispatching rules (i.e., SPT, MDD and ODD) are close to each other when compared to MOD. Therefore, we repeat the same analyses for the set of dispatching rules SPT, MDD and ODD for tight due date case. Our objective is to see the behavior of *BestPerf* when the candidate dispatching rules have close performance to each other. Figures 4.6 and 4.7 show the corresponding simulation results. The behavior of *BestPerf* is still same as the previous results. Moreover, the minimum value achieved is more apparent in this case. Also, for 80% utilization case, *BestPerf* is better than any single-pass dispatching rule for every choice of SPL (Figure 4.6-a). These results clearly show that even if the long-run performances of the candidate dispatching rules are relatively close to each other, significant improvements can be achieved by selecting appropriate scheduling period length and the dispatching rule combination for the entire planning horizon.

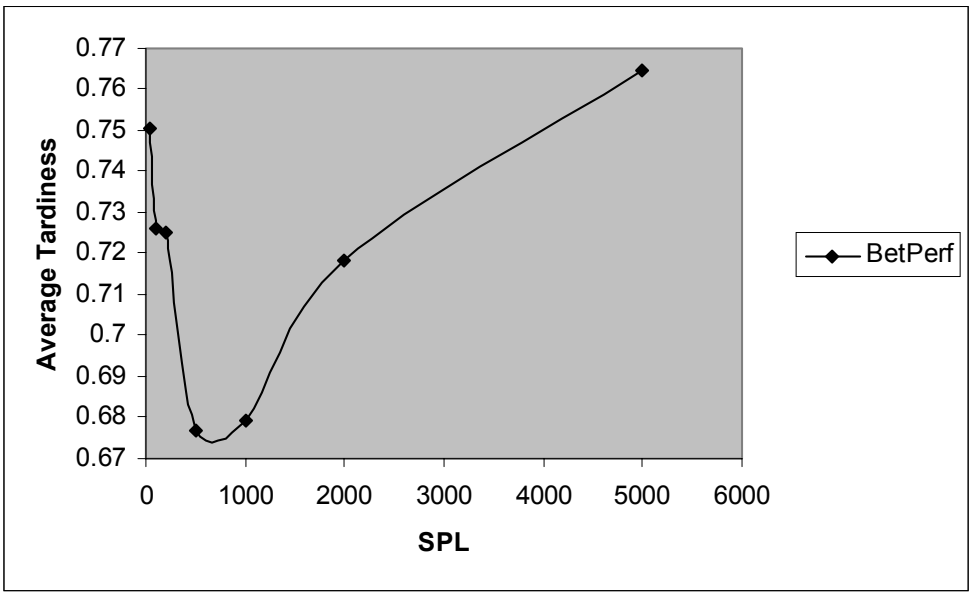
The detailed simulation results can be found in Appendix C. Based on the experimental results of this section we decide to use the following parameters in the rest of the experiments (Table 4.3).

Table 4.3: Parameter values considered for further experimentation

| Utilization Levels | Due date Tightness | Scheduling Period Length (SPL) | Candidate Dispatching Rule Set |
|---|---|---|---|
| 80 % | tight | 1000 | {SPT, MDD, ODD} |
| 90 % | | 7500 | {SPT, MDD, ODD, MOD} |

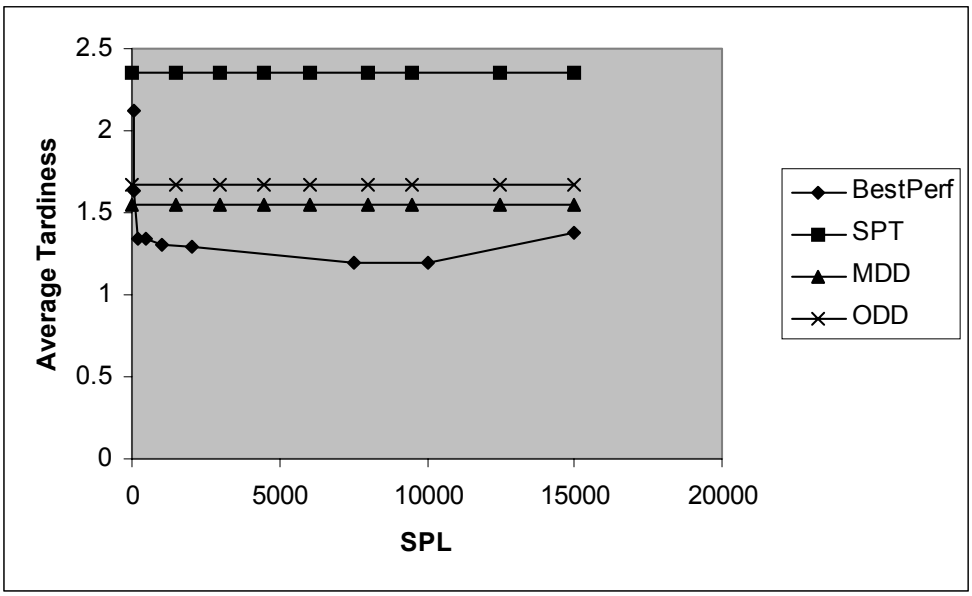



(a)

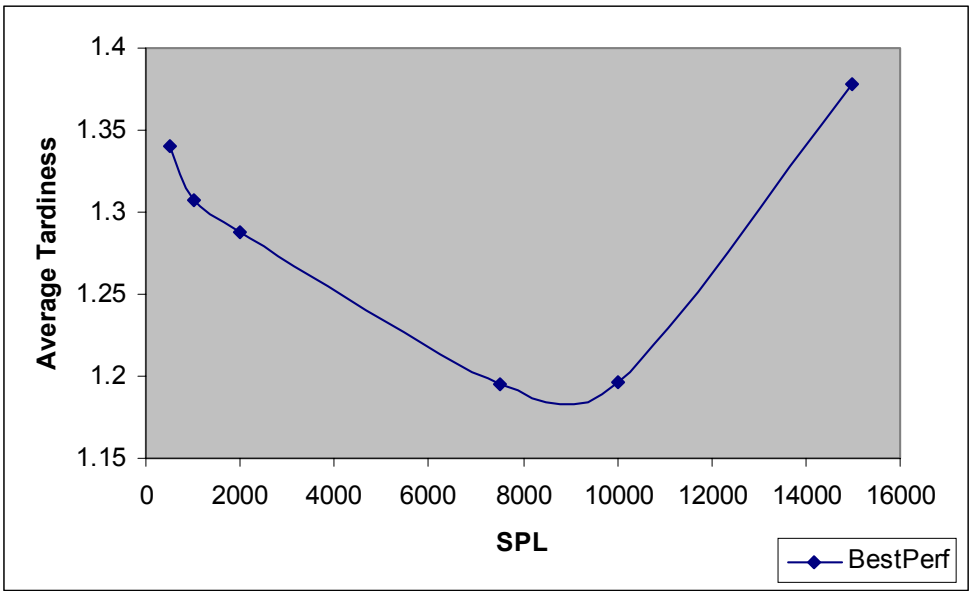



(b)

Figure 4.6: 80% utilization, tight due-dates with CDR set {SPT, MDD, ODD}. (a) Complete display of the experiments. (b) Zoom-in version around the minimum.




(a)




(b)

Figure 4.7: 90% utilization, tight due-dates CDR set {SPT, MDD, ODD}. (a) Complete display of the results. (b) Zoom-in version around the minimum.

## 4.4. The Effect of Monitoring Point (MP) and β-parameter Selection

As mentioned in Section 3.3.2, one of the criteria for when-to-schedule decision is monitoring point (MP) symptom. In the previous section, we experimented for selecting the appropriate scheduling period length when there is no monitoring on the system performance. In this section we address the questions of how far these monitoring points should be apart from each other and what should be the value of β-parameter (see 3.3.2 for definition of β, $\chi$ and $\overline{\overline{X}}$). The following additional assumptions are also used in this section:

1. At the beginning of each scheduling period a dispatching rule is selected among the candidate rules. This rule may be used until the end of that scheduling period or may be changed at some monitoring point within the scheduling period.
2. At any monitoring point, the decision to change the existing rule or continue with it (until the next monitoring point or end of the scheduling period) is given by the procedure defined in Section 3.3.2.
3. The appropriate monitoring period length is determined by looking at the minimum *BestPerf* value.

Experimental design is given in Table 4.4. This is a nested experimental design, in which the factor monitoring period length (MPL) is nested inside the factor scheduling period length (SPL). The reason for this kind of a design requirement is that selecting a monitoring period length of 2500 makes nonsense when the scheduling period length is 1000. Hence, the factor levels of MPL depend on the level of SPL and therefore nested in factor SPL. For SPL value of 1000, three levels of monitoring period lengths are considered, whereas four levels of MPLs are tested for SPL of 7500. Note that, MPL of 1000 for SPL being 1000 corresponds to the case of

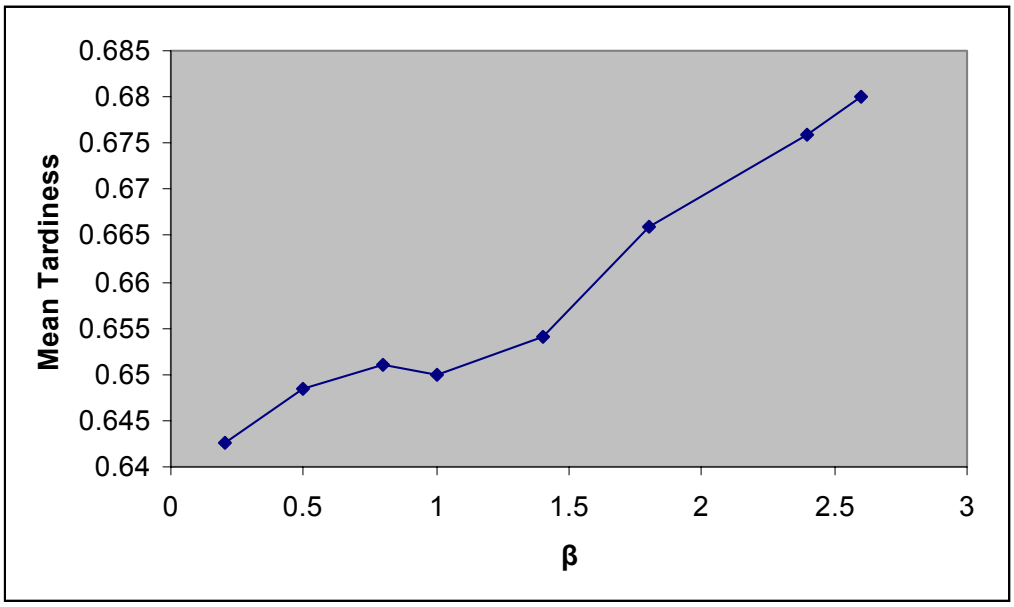
no monitoring at all, since they are equal. This is also the case for MPL of 7500 for SPL being 7500. 12 levels of β parameter are used in the experiments. The value of $\overline{\overline{X}}$ is taken to be 0.6795 and 1.195 for 80% and 90% utilizations, respectively. These values are the *BestPerf* values for 80% and 90% utilizations with respective SPLs (i.e., SPL=1000 for 80%, SPL=7500 for 90% utilization), which we found in the previous section.

Table 4.4: Experimental design of monitoring period length and β-parameter

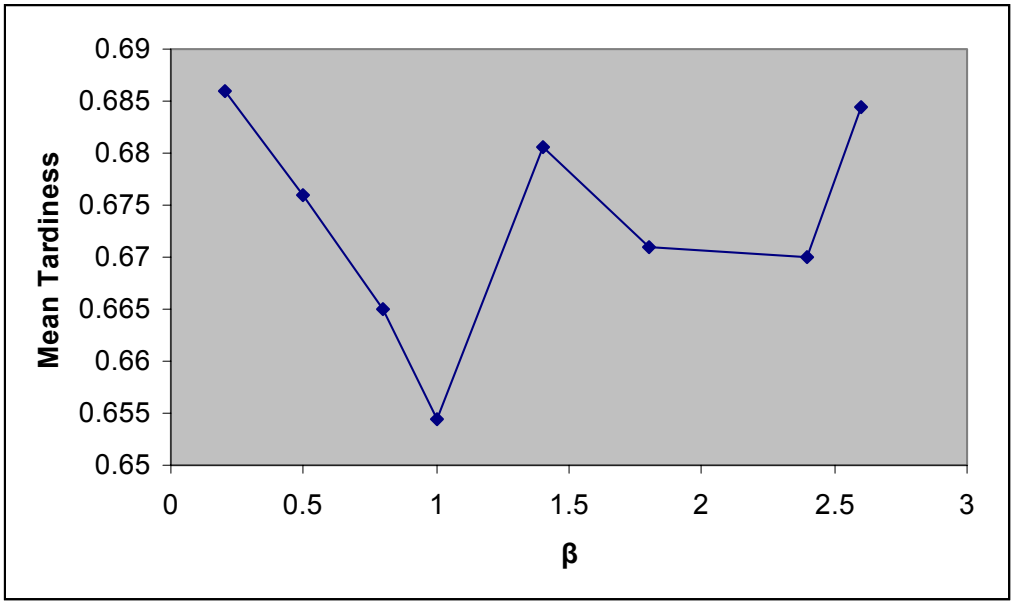
| **Factors** | **Levels** | | | | | | |
|---|---|---|---|---|---|---|---|
| **Due date tightness** | Tight | | | | | | |
| **Dispatching rule set** | {SPT, MDD, ODD} | | | | | | |
| β | 0.2, 0.5, 0.8, 1, 1.4, 1.8, 2.2, 2.4, 2.6 | | | | | | |
| **Utilization** | 80% | | | 90% | | | |
| **SPL** | 1000 | | | 7500 | | | |
| **MPL** | 250 | 500 | 1000 | 500 | 2500 | 3750 | 7500 |

Again, the simulation runs are taken in steady state with 20 replications and each replication with 200000 minutes of a planning horizon for each factor combination. Also, common random numbers are used when deciding which dispatching rule to use in a scheduling period. All of the simulation results are tabulated in Appendix D.

The results for 80% utilization case are shown in Figure 4.8. In Figure 4.8-a, for monitoring period length of 250, best mean tardiness value is achieved with a β value of 0.2, and for MPL of 500 a β value of 1 yields the best (Figure 4.8-b). These points are statistically smaller than the others on 0.95 confidence, which is also tested by paired-t test. We also compare the performances of best MPL-β pairs with each other, which is also shown in Figure 4.9. These points are also statistically different




(a)




(b)

Figure 4.8: *BestPerf* for various MPL-β combinations when system utilization is 80%. (a) MPL = 250. (b) MPL = 500.

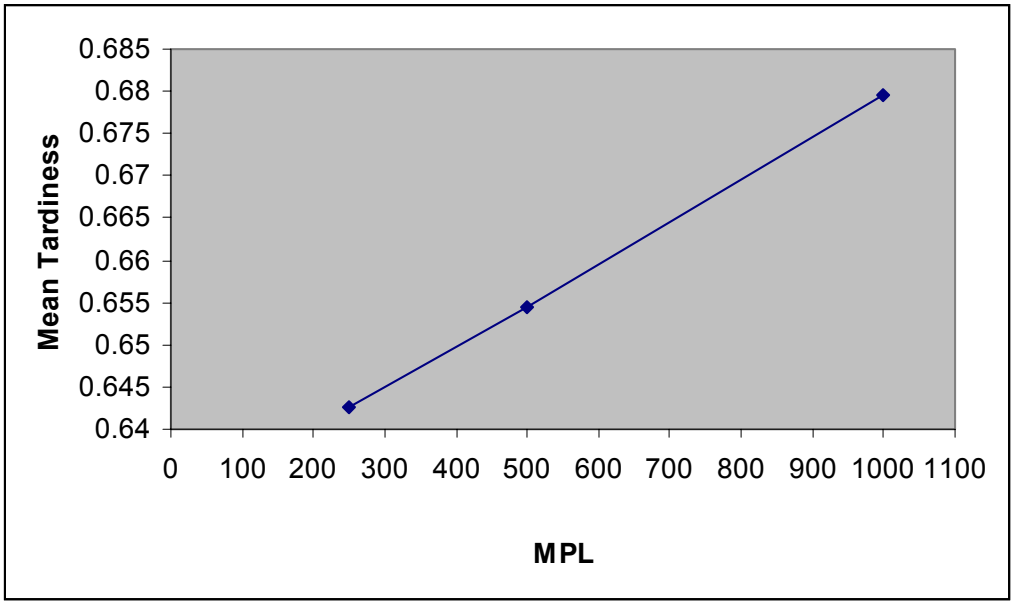
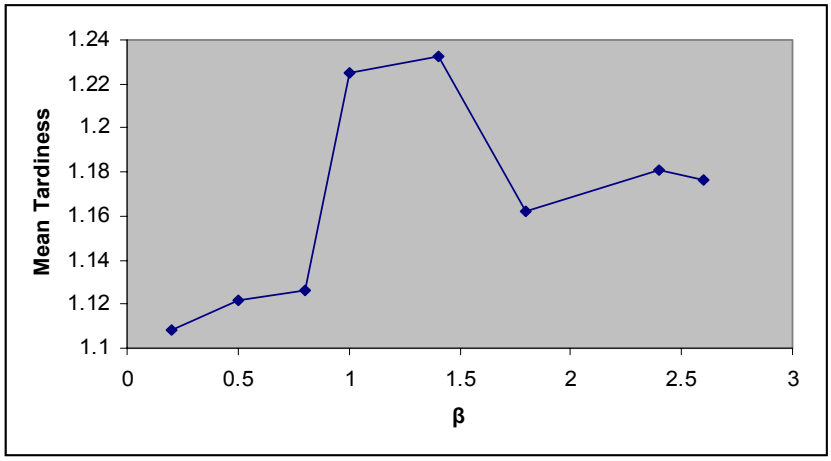



Figure 4.9: Comparison of best MPL-β pairs for 80% utilization

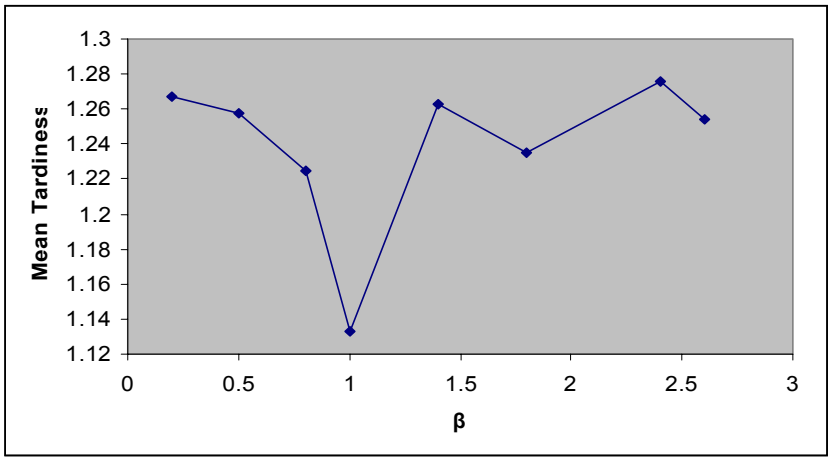
than each other on 0.95 confidence level. Hence, our results indicate that monitoring the system performance improves our performance measure (i.e., average tardiness). In addition to that, imposing frequent monitoring points on the system with a small β value further improves the system performance.

For 90% utilization case, the results of the experiments are shown in Figure 4.10. In Figure 4.10-a, for monitoring period length of 500, best mean tardiness value is achieved with a β value of 0.2, and for MPLs of 2500 and 3750, β value of 1 yields the best (Figures 4.10-b and 4.10-c). These points are statistically smaller than the others on 0.95 confidence level. We compare the performances of best MPL-β pairs with each other, which is also shown in Figure 4.11. The results are both similar in 80% and 90% cases, where system performance deteriorates when we increase MPL (Figures 4.9 and 4.11). Also, these points in Figure 4.11 are statistically different than each other on the 0.90 confidence level.

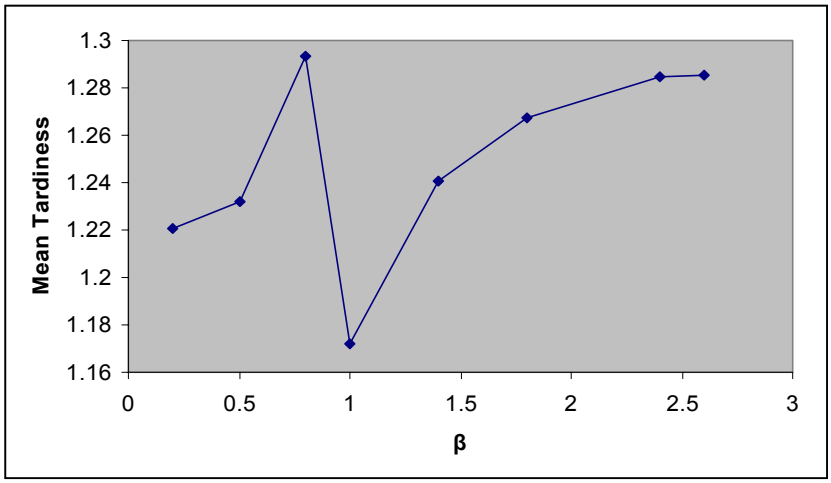
In summary, for both 80% and 90% utilizations, monitoring the system performance in discrete points in time improves our objective function (i.e., average




(a)




(b)




(c)

Figure 4.10: *BestPerf* for various MPL-β combinations when system utilization is 90%. (a) MPL = 500. (b) MPL = 2500. (c) MPL = 3750.

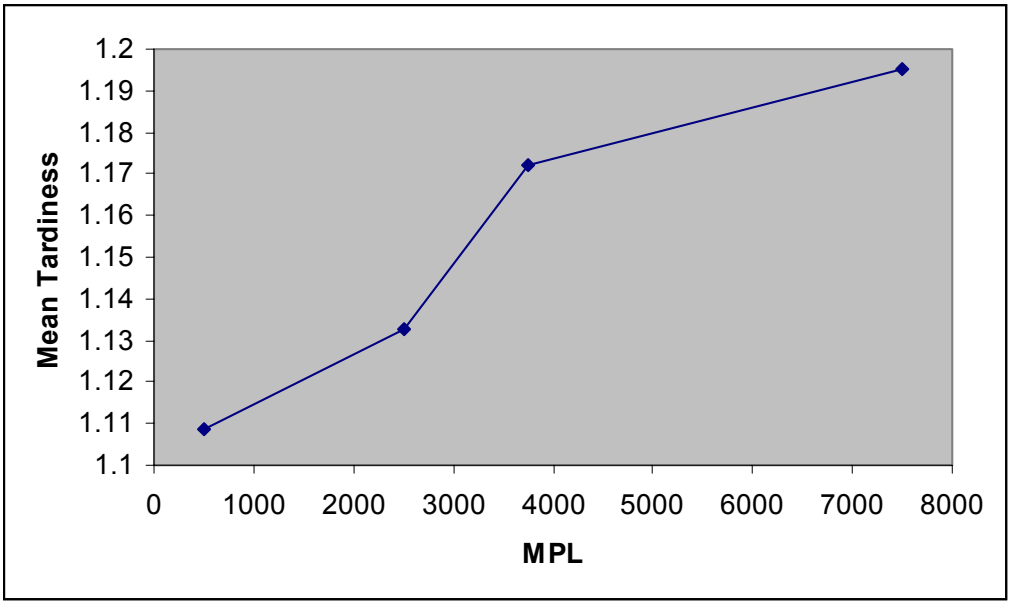



Figure 4.11: Comparison of best MPL-β pairs for 90% utilization.

tardiness). Also, experiments show that it is vitally important to select not only the right monitoring period length, but also the right β parameter for that MPL. For example, for 90% utilization case, a MPL of 2500 with a β value other than 1 result in worse performance measures than no monitoring case (i.e., mean tardiness values greater than 1.195). Moreover, for both 80% and 90% utilizations, system performance improves with small monitoring period lengths. In addition to that, for small monitoring period lengths, small β values work better. For example, for 80% utilization, it is best to choose β=0.2 when MPL=250 whereas β=1 when MPL=500.

## 4.5. The Selection of System Attributes

As we defined in Section 3.1, the system attributes carry information about the state of the manufacturing system. These attributes take its place as the fundamental structure in the representation of instance data (see Figure 3.5 for instance). Learning module of our proposed system utilizes the information in these instance data to learn about the characteristics of the manufacturing system. After the learning algorithm

processes these data, we end up with a learning tree by which we will select dispatching rules to be used in future scheduling periods. At the beginning of each forthcoming scheduling period, the system attributes' values are set by looking at the state of the manufacturing system. Then, a dispatching rule is selected from the learning tree by branching on the tree based on the values of these attributes. The selected rule is used until the end of the scheduling period and a new dispatching rule is selected for the next scheduling period by following the same procedure.

It is evident from the above discussion that it is extremely vital to use the right set of attributes not only to construct a representative learning tree but also to branch correctly on the constructed tree to end up with the right dispatching rule to use. Since we don't have a given set of attributes to use, we have to decide the set of attributes to use among our candidate rules, which are given in Appendix B. In this section, we decide the set of attributes that will be used in our system to generate high quality learning trees. Before attacking the question, we first try to get insights about the problem. For this, we create a number of attribute sets and test the performance of the learning trees, which are created upon these attribute sets. From these initial tries, we observe the following:

1. Increasing the number of attributes in the set does not necessarily improve the quality of the learning tree.
2. The effect of each attribute in the set on the performance of the generated learning tree depends on the other attributes in the set.
3. Performance of the generated tree deteriorates when the attribute set contain Attribute-2 and Attribute-3 of Appendix B.

Based on these observations we disregard the Attributes 2 and 3 in Appendix B from further consideration. However, we still have 24 attributes to consider for

selection and considering every combination of these attributes requires testing of $2^{24}$ attribute sets, which is impractical. Therefore, we ignore the 2nd observation above and assume that the effect of each attribute on the quality of the learning tree is independent of the other attributes in that set. Then we use the following heuristic algorithm to select the attributes among the candidates:

Step 0. (Initialization) Define $C$ as the set of all candidate attributes. $D_1$ is the set of training data and $D_2$ is the set of test data. Construct the tree on $D_1$ set with attributes in $C$ and test the performance of the tree on $D_2$ set. Let this performance denoted by $P(C)$.

Step 1. Discard attribute-i from C and construct the tree on $D_1$ set with attributes in $C\backslash\{i\}$ and test the performance of the tree ($P(C\backslash\{i\})$) on $D_2$ set. Repeat this step for all $i \in C$.

Step 2. Let new $C$ be $C\backslash\{i\}$ for $i$, where $i = \arg\max\{P(C \setminus \{j\}) \mid j \in C\}$. Continue with the next step if $C=\emptyset$, otherwise return Step 2.

Step 3. Select the attribute set that result in the maximum performance among all tested.

In this algorithm, the performance function $P(*)$ is defined as the percentage of correctly classified instances in the test data set, $D_2$. By using this algorithm we experiment for the right attribute set that should be used by the learning tree for two cases: when we have sufficiently large data to learn on and when the data is scarce. For sufficiently large data and scarce data cases, we use 2000 and 200 training data, respectively. The size of the test data set is taken to be 2000 in both cases. Data sets, $D_1$ and $D_2$, are generated under the experimental conditions given in Table 4.5.

Table 4.5: Experimental conditions for generated data sets

| Utilization Level | Due date tightness | SPL | MPL | β | Dispatching Rules |
|---|---|---|---|---|---|
| 80 % | Tight | 1000 | 250 | 0.2 | SPT, MDD, ODD |

For both scarce and sufficiently large data set cases, the algorithm found that the attribute set that result in the best learning tree contains all the attributes in the initial set *C*. In other words, we should use all the attributes defined in Appendix B other than the second and third attributes. The performance values of the learning trees in each case with all attributes used are provided in Table 4.6. Interestingly, the performance of the tree is found to be better in the scarce data case than the large data case but the difference is not significantly high.

Table 4.6: Summary of the experimental results on the attribute set selection

| Cases | Performance of the Tree |
|---|---|
| Sufficiently large data | 80.7 % |
| Scarce data | 81.55 % |

Based on the experimental results of this section, we embodied all the attributes defined in Appendix B (not including the second and third attributes) into our system. In the rest of the experiments where learning take place, the learning tree is constructed upon this attribute set.

## 4.6. Job Shop Scheduling with a Static Learning Tree

In the previous sections of this chapter, we experimented with the important system parameters. Now in this section, we use our learning based scheduling system in a job shop environment. First we measure the performance of the proposed system when the tree is constructed only once. That is, the learning tree is not updated over time. For that reason, we call this application as scheduling with a *static* learning tree.

For the experiments we again use a nested experimental design (Table 4.7). In this design, MPL factor is nested inside the SPL factor, as it is in the previous section, and the β factor is nested inside the MPL factor. The reason for nesting the β factor inside the MPL factor is due to the fact that we take the best level of β factor for each MPL . Two different dispatching rule sets are also considered in the experiments, where one of the sets contains MOD and the other set does not.

Table 4.7: Experimental design of scheduling with a static learning tree

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

In the simulation experiments, we take 20 replications for each experimental condition and each replication is composed of two phases: the Warm-up Phase and the Testing Phase. In the warm-up phase we generate the necessary instance data for our learning algorithm to construct the learning tree. This phase is composed of 2000 scheduling periods (for the 80% utilization settings), which provides us a training data set that contains 2000 instance data (i.e., each scheduling period provides one instance data). At the end of this warm-up phase, a learning tree is constructed by using this training data set and the second phase starts. In the second phase, the dispatching rules for each scheduling period are selected from the learning tree. This phase also contains 2000 scheduling periods for each replication in the 80% utilization case and the statistics are collected in this phase (i.e., *BestPerf*, *LearnPerf* and *MultiPass*). As before, the common random numbers (CRN) scheme is used in the experiments. In the 90% utilization case, each replication is composed of only 1000 scheduling

periods for both phases due to the high memory requirement in the high utilization case.

We are interested in the following questions:

1. Does the learning based system give better average tardiness values than the simulation-based multi-pass algorithms (i.e., can we get *LearnPerf* < *MultiPass*?)
2. What are the percentage differences between *BestPerf-LearnPerf* and *BestPerf-MultiPass*?
3. What are the percentage use of dispatching rules for *BestPerf*, *LearnPerf* and *MultiPass*?

The results of the experiments are given in Appendix E and Tables 4.8 through 4.10 summarizes these results. As it is expected *BestPerf* gives the lower bounds for both *MultiPass* and *LearnPerf*. In both of the experimental conditions, our learning-based scheduling system performs better than the simulation-based scheduling (see Table 4.8 and 4.9). However, *LearnPerf* approaches to *MultiPass* as we increase the monitoring period length. At the extreme, when there is no monitoring at all, the performances of learning-based and simulation-based scheduling approaches become almost equal. This result is consistent with our findings in Section 4.3, in which smaller values for MPL results better average tardiness values for *BestPerf*. Therefore, it is vital to set the appropriate SPL, MPL and β values to get the maximum efficiency from the learning-based system.

For the small values of MPL, 250 for 80% and 500 for 90% utilizations, the percentage of the gap between the *LearnPerf* and the *BestPerf* are considerably smaller (at least the half) than the gap between *MultiPass* and *BestPerf*. Also, the percentage of the gap between the *LearnPerf* and the *BestPerf* found to be better in

the high utilization cases (i.e., 90% utilization) when compared with the low utilization case. This shows that our proposed system works much better when the utilization increases. Also, based on the paired-t test results we show that these performance values are statistically different than each other on a 0.95 confidence interval.

Also, when we compare the results given in Table 4.8 with the results of Table 4.9, we observe that when we add a very competitive dispatching rule, such as MOD, to the dispatching rule set, all the performance metrics (*BestPerf, LearnPerf* and *MultiPass*) improve significantly (almost 50% better results). Moreover, *LearnPerf* and *MultiPass* get closer to *BestPerf* (small $\Delta_1$ and $\Delta_2$ values in Table 4.9 than in Table 4.8) when the MOD is added to the rule set. Alternatively, we expect that

Table 4.8: Summary of performance values for the rule set {SPT, MDD, ODD}

| Utilization | MPL | BestPerf | LearnPerf | MultiPass | Single-pass | $\Delta_1$ | $\Delta_2$ |
|---|---|---|---|---|---|---|---|
| **80%** | **250** | 0.648 | 0.799 | 0.894 | 0.905 | 23.3% | 37.96% |
| | **500** | 0.655 | 0.876 | 0.896 | 0.905 | 33.74% | 36.79% |
| | **1000** | 0.679 | 0.891 | 0.895 | 0.905 | 31.22% | 31.81% |
| **90%** | **500** | 1.1 | 1.383 | 1.494 | 1.545 | 25.72% | 35.81% |
| | **2500** | 1.139 | 1.487 | 1.532 | 1.545 | 30.55% | 34.5% |
| | **7500** | 1.196 | 1.51 | 1.514 | 1.545 | 26.25% | 26.58% |

Table 4.9: Summary of performance values for the rule set {SPT,MDD,ODD,MOD}

| Utilization | MPL | BestPerf | LearnPerf | MultiPass | Single-pass | $\Delta_1$ | $\Delta_2$ |
|---|---|---|---|---|---|---|---|
| **80%** | **250** | 0.359 | 0.415 | 0.492 | 0.52 | 15.59% | 37.04% |
| | **500** | 0.374 | 0.428 | 0.44 | 0.52 | 14.43% | 17.64% |
| | **1000** | 0.383 | 0.435 | 0.44 | 0.52 | 13.57% | 14.88% |
| **90%** | **500** | 0.52 | 0.568 | 0.684 | 0.704 | 9.23% | 31.53% |
| | **2500** | 0.539 | 0.587 | 0.632 | 0.704 | 8.9% | 17.25% |
| | **7500** | 0.559 | 0.591 | 0.595 | 0.704 | 5.72% | 6.44% |

$$\Delta_1 = 100 \times \frac{(LearnPerf - BestPerf)}{BestPerf},\ \Delta_2 = 100 \times \frac{(MultiPass\text{-}BestPerf)}{BestPerf}$$

*LearnPerf* and *MultiPass* to move away from *BestPerf* when a low-quality DR is added to the candidate dispatching rule set.

We also keep track of the dispatching rule usage percentages in the experiments for both *MultiPass*, *LearnPerf* and *BestPerf*. Table 4.10 summarizes these percentages and the detailed values are given in Appendix E. Note that the values don't add up to 100 because of the rounding. As it is clear from Table 4.10, for the low values of MPLs, learning-based scheduling system uses dispatching rules as close to their best dispatching rule combinations. On the other hand, simulation-based scheduling gives much different values than the best combinations for small MPLs. For high values of MPL, percentage usage of the dispatching rules for each system converge to each other as expected.

Table 4.10: Average dispatching rule usage percentages

| | | 80% Utilization | | | 90% Utilization | | |
|---|---|---|---|---|---|---|---|
| **Dispatching Rule Percentages** | **Rules** | **MPL= 250** | **MPL= 500** | **MPL= 1000** | **MPL= 500** | **MPL= 2500** | **MPL= 7500** |
| **Multi-pass** | **SPT** | 13% | 14% | 14% | 2% | 2% | 2% |
| | **MDD** | 34% | 34% | 34% | 58% | 58% | 58% |
| | **ODD** | 52% | 51% | 51% | 38% | 39% | 39% |
| | | | | | | | |
| **Learning** | **SPT** | 10% | 14% | 14% | 2% | 4% | 2% |
| | **MDD** | 54% | 38% | 32% | 84% | 63% | 58% |
| | **ODD** | 34% | 46% | 53% | 12% | 32% | 38% |
| | | | | | | | |
| **Best** | **SPT** | 9% | 13% | 14% | 2% | 3% | 2% |
| | **MDD** | 54% | 39% | 33% | 82% | 63% | 58% |
| | **ODD** | 36% | 47% | 52% | 14% | 33% | 38% |

In summary, our learning-based scheduling system performs better than the simulation-based scheduling approach in all the experiments. The results also show the importance of setting for the SPL, MPL and β parameters. When they are fine tuned up, the proposed system can provide significant improvements on the system

performance. Also, we expect further improvements when proposed learning tree is updated in time and we will analyze this in the next section.

## 4.7. Job Shop Scheduling with Dynamic Learning Structure

In previous sections, we focused on important issues such as the selection of scheduling period and monitoring period, attributes and so on. We also tested our learning-based scheduling system with a static learning tree. In this section, we test our learning-based scheduling system with a dynamic learning tree (i.e., all of its modules discussed in Chapter 3 are activated). In other words, we now continuously monitor the quality of the learning tree by the control charts and update it whenever necessary. Thus, we call this experiment as the scheduling with a *dynamic learning structure*.

In the simulation experiments, we consider a manufacturing system in which its internal parameters change in time (i.e., arrival rate, due date tightness levels). The details of the experimental design are given in Table 4.11. Note that, we take 5 planning horizons, where each horizon contains 1000 scheduling periods. At the beginning of each horizon, we change some of the parameters of the manufacturing system. For example, in Table 4.11, the factor "parameter sequence for arrival rate" represents the value of the arrival rate of the jobs during each horizon. Specifically, in horizons 1, 2, 3, 4 and 5, jobs arrive exponentially with parameters 0.8, 0.9, 0.7, 0.9 and 0.8, respectively. For the construction of the learning tree, we consider two different strategies, which are represented by the factor "Training Data Set" in Table 4.11. When this factor is at its level Full, the learning tree is constructed based on all the accumulated data points since the beginning of the experiment. On the other hand,

if its level is set to Partial, the most recent 200 data points (1/5 of a horizon length) are used each time when the learning tree is updated.

Table 4.11: Experimental Design for scheduling with dynamic learning structure

| **Factors** | **Levels** |
|---|---|
| DR set | {MOD, MDD, ODD, SPT}, {MDD, ODD, SPT} |
| Sequence for arrival rate parameter | {0.8, 0.9, 0.7, 0.9, 0.8} |
| Horizon lengths (number of SPs) | 1000 |
| Training Data Set | Full, Partial (1/5 of horizon length) |
| SM2 type | Reactive, non-reactive, partially reactive |
| Due date tightness | Adjusted, not adjusted |
| (SPL, MPL, β) | {(1000, 250, 0.2), (7500, 500, 0.2)} |

The next factor, SM2 type, represents the characteristics of the simulation module 2 (see Figure 4.1). Recall that the SM2 is responsible for assessment of the decisions if they are given via the simulation runs rather than the learning tree (i.e., multi-pass). We consider three levels for the SM2 type: *reactive*, *non-reactive* and *partially reactive*. When SM2 type is reactive, SM2 model is updated immediately when there is any parameter change in the actual manufacturing environment. In other words, if the arrival rate of the jobs changes in real world, this information is made available for simulation model 2, which is used for simulation-based scheduling, immediately. Intuitively, this is impossible in the real world implementation, because when any parameter of the manufacturing system changes it can be made available to the simulation model of the system after a period of time. This delay is inevitable since detecting the shift in the parameters requires data collection and statistical analysis. For this reason, we also consider the partially reactive level for SM2 type. When the type is partially reactive, SM2 is updated for the arrival rate changes, but with some time delay and an accuracy level. Specifically, arrival rate in SM2 is updated with a delay of 200 scheduling periods (1/5 of a horizon length) after the

actual change in the real world takes place and set to the values in the sequence {0.8, 0.875, 0.725, 0.875, 0.8} for each horizon 1 through 5, respectively. The time delay for the update in SM2 represents the passage of time for collecting sufficient data, which is necessary to statistically determine the new arrival rate. As another extreme, we consider SM2 type as non-reactive. In this case the model, SM2, is not updated for any changes in the manufacturing environment. For example, when arrival rate changes from 0.8 to 0.9 in real world, simulation-based scheduling (SM2) continues to operate under the initial arrival rate, which is 0.8.

Another factor considered in the experiments is *due date tightness* and it has two levels, *adjusted* and *not adjusted*. For the adjusted case, we set the allowance factor *k* for setting the due dates such that percent tardy is always 40% under the FCFS rule. In Section 4.2, we found k equal to 5.5 and 11 for the arrival rates 0.8 and 0.9 to achieve 40% of percent tardy jobs, respectively. We also look for the arrival rate of 0.7 and k being 3.75 results in 40% tardy. Therefore, for the adjusted case, flow allowance factor k is set to 3.75, 5.5 and 11 for arrival rates of 0.7, 0.8 and 0.9, respectively, for all simulation models SM1, SM2 and SM3 (these models are discussed in Section 4.1). For the not adjusted case, flow allowance factor is always at the level 5.5 for all arrival rates. Therefore, the first case corresponds to a policy such that the manufacturing firm adjusts its due date setting policy when the arrival rate of the jobs changes and in the second case no action is taken for setting the due dates of the jobs when the utilization of the shop floor changes.

The last factor that we consider in the experiments is the choice of scheduling and monitoring period lengths along with the β value. For the levels of this factor, we simply consider the best combinations that we previously determined for 80% and

90% utilization levels. Therefore, the two levels, (1000, 250, 0.2) and (7500, 500, 0.2), are considered for this factor.

With these factors and their associated levels, we end up with 48 experimental conditions. At the beginning of each experiment, there is a warm up period with a length of 200 scheduling periods to provide necessary initial data to the system to construct the first learning tree and the control charts. System statistics are initialized after the warm up period and each experimental condition is run for 5 consecutive horizons (5000 scheduling periods) as it is given in Table 4.11. The results of the experiments, *MultiPass*, *LearnPerf* and *BestPerf*, are summarized in Table 4.12 and 4.13. Note that, *BestPerf* provides the lower bound values for both *MultiPass* and the *LearnPerf*.

From these results, our first observation is that our learning-based scheduling system outperforms the simulation-based scheduling approach (*MultiPass*) in 38 experimental conditions out of 48. In these cases, *LearnPerf* is closer to *BestPerf* more than *MultiPass* in a range of 2.34% to 40.87%. In 2 cases, both *MultiPass* and *LearnPerf* is found as equal. In the remaining 8 cases simulation-based scheduling (*MultiPass*) perform slightly better than *LearnPerf* (i.e., between 1.68% and 7.83% better). However, in these cases SM2 type is reactive, which is difficult to achieve such conditions in the real world.

When we compare *LearnPerf* for full and partial training data set cases, we see that using all available data always results better performances (see Tables 4.12 and 4.13). At first glance, this seems to be counter intuitive because when parameters of the manufacturing system change, learning with the most recent data is expected to yield better performance. However, the results show that our learning algorithm gets benefit from the past data as well as the recent data. Note that, *BestPerf* and *MultiPass*

Table 4.12: Summary of the experimental results for DR set {MDD, ODD, SPT}

| | | **Training data set:** | **Full** | | **Partial** | |
|---|---|---|---|---|---|---|
| | | **(SPL, MPL, β):** | **(1000, 250, 0.2)** | **(7500, 500, 0.2)** | **(1000, 250, 0.2)** | **(7500, 500, 0.2)** |
| **Due date tightness:** | **SM2 type:** | | | | | |
| **Adjusted** | **Reactive** | MultiPass | 1.18 | 1.25 | 1.18 | 1.25 |
| | | LearnPerf | 1.1 | 1.11 | 1.13 | 1.2 |
| | | BestPerf | 0.98 | 1.02 | 0.98 | 1.02 |
| | **Non-reactive** | MultiPass | 1.37 | 1.52 | 1.37 | 1.52 |
| | | LearnPerf | 1.1 | 1.11 | 1.13 | 1.2 |
| | | BestPerf | 0.98 | 1.02 | 0.98 | 1.02 |
| | **Partially Reactive** | MultiPass | 1.25 | 1.32 | 1.25 | 1.32 |
| | | LearnPerf | 1.1 | 1.11 | 1.13 | 1.20 |
| | | BestPerf | 0.98 | 1.02 | 0.98 | 1.02 |
| **Not Adjusted** | **Reactive** | MultiPass | 2.38 | 1.79 | 2.38 | 1.79 |
| | | LearnPerf | 2.3 | 1.75 | 2.42 | 1.9 |
| | | BestPerf | 2.15 | 1.71 | 2.15 | 1.71 |
| | **Non-reactive** | MultiPass | 2.59 | 2.26 | 2.59 | 2.26 |
| | | LearnPerf | 2.3 | 1.75 | 2.42 | 1.9 |
| | | BestPerf | 2.15 | 1.71 | 2.15 | 1.71 |
| | **Partially Reactive** | MultiPass | 2.49 | 2.02 | 2.49 | 2.02 |
| | | LearnPerf | 2.3 | 1.75 | 2.42 | 1.9 |
| | | BestPerf | 2.15 | 1.71 | 2.15 | 1.71 |

Table 4.13: Summary of the experimental results for DR set {MDD, ODD, SPT, MOD}

| | | **Training data set:** | **Full** | | **Partial** | |
|---|---|---|---|---|---|---|
| | | **(SPL, MPL, β):** | **(1000, 250, 0.2)** | **(7500, 500, 0.2)** | **(1000, 250, 0.2)** | **(7500, 500, 0.2)** |
| **Due date tightness:** | **SM2 type:** | | | | | |
| **Adjusted** | **Reactive** | MultiPass | 0.81 | 0.65 | 0.81 | 0.65 |
| | | LearnPerf | 0.81 | 0.65 | 0.82 | 0.68 |
| | | BestPerf | 0.71 | 0.6 | 0.71 | 0.6 |
| | **Nonreactive** | MultiPass | 0.96 | 0.73 | 0.96 | 0.73 |
| | | LearnPerf | 0.81 | 0.65 | 0.82 | 0.68 |
| | | BestPerf | 0.71 | 0.6 | 0.71 | 0.6 |
| | **Partially Reactive** | MultiPass | 0.87 | 0.69 | 0.87 | 0.69 |
| | | LearnPerf | 0.81 | 0.65 | 0.82 | 0.68 |
| | | BestPerf | 0.71 | 0.6 | 0.71 | 0.6 |
| **Not Adjusted** | **Reactive** | MultiPass | 1.52 | 1.2 | 1.52 | 1.2 |
| | | LearnPerf | 1.2 | 1.15 | 1.61 | 1.27 |
| | | BestPerf | 1.15 | 1.1 | 1.15 | 1.1 |
| | **Nonreactive** | MultiPass | 1.67 | 1.35 | 1.67 | 1.35 |
| | | LearnPerf | 1.2 | 1.15 | 1.61 | 1.27 |
| | | BestPerf | 1.15 | 1.1 | 1.15 | 1.1 |
| | **Partially Reactive** | MultiPass | 1.6 | 1.26 | 1.6 | 1.26 |
| | | LearnPerf | 1.2 | 1.15 | 1.61 | 1.27 |
| | | BestPerf | 1.15 | 1.1 | 1.15 | 1.1 |

are not affected from this parameter, since the training data set characteristic only influences the learning tree and have not any influence on these performances.

The third observation is related with the selection of SPL, MPL and β values. For the rule set of {MOD, MDD, ODD, SPT}, the combination (7500, 500, 0.2) gives always better results for *LearnPerf* than the combination (1000, 250, 0.2) (Table 4.13) regardless of partial or full data sets being used. The reason why the combination (7500, 500, 0.2) yields better results when MOD is in the rule set is that the performance of MOD dominates the performance of other rules when it is used for a long period of time. Thus, the combination (7500, 500, 0.2) yields better results than the combination (1000, 250, 0.2). For the rule set {MDD, ODD, SPT}, the best choice of (SPL, MPL, β) combination depends on the parameter "due date tightness". When the due date tightness factor is at its level *not adjusted*, the choices of (7500, 500, 0.2) results in again the improved performance than (1000, 250, 0.2) regardless of the partial or full data sets being used. But, when it is at the *adjusted* level, (7500, 500, 0.2) and (1000, 250, 0.2) results in better for the full and partial training data sets, respectively, (Table 4.12). These results stress us the importance of the appropriate selection of SPL, MPL and β values once more.

As stated before, partially reactive SM2 is a more realistic case for the simulation-based scheduling approach. In this case, the simulation model used for the scheduling decisions of multi-pass approach is updated with some time delay and inaccuracy that may exist in detecting the parameter changes in the actual manufacturing environment. Therefore, the comparison of the learning-based (*LearnPerf*) and the simulation-based (*MultiPass*) systems for this factor level is of special importance. When the SM2 type is partially reactive, *LearnPerf* is better than *MultiPass* in 14 cases out of 16. That is, *LearnPerf* is closer to *BestPerf* more than *MultiPass* in a range of 3.26% to 34.79% in these 14 cases. In the

Table 4.14: Summary of the experimental results for DR set {MDD, ODD, SPT} (Percentage of deviation from the best)

| | | **Training data set:** | **Full** | | **Partial** | |
|---|---|---|---|---|---|---|
| | | **(SPL, MPL, β):** | **(1000, 250, 0.2)** | **(7500, 500, 0.2)** | **(1000, 250, 0.2)** | **(7500, 500, 0.2)** |
| **Due date tightness:** | **SM2 type:** | | | | | |
| **Adjusted** | **Reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 20.40% | 22.55% | 20.40% | 22.55% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 12.24% | 8.82% | 15.30% | 17.64% |
| | **Non-reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 39.79% | 49.01% | 39.79% | 49.01% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 12.24% | 8.82% | 15.30% | 17.64% |
| | **Partially Reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 27.55% | 29.41% | 27.55% | 29.41% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 12.24% | 8.82% | 15.30% | 17.64% |
| **Not Adjusted** | **Reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 10.69% | 4.67% | 10.69% | 4.67% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 6.97% | 2.33% | 12.55% | 11.11% |
| | **Non-reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 20.46% | 32.16% | 20.46% | 32.16% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 6.97% | 2.33% | 12.55% | 11.11% |
| | **Partially Reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 15.81% | 18.12% | 15.81% | 18.12% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 6.97% | 2.33% | 12.55% | 11.11% |

Table 4.15: Summary of the experimental results for DR set {MOD, MDD, ODD, SPT} (Percentage of deviation from the best)

| | | **Training data set:** | **Full** | | **Partial** | |
|---|---|---|---|---|---|---|
| | | **(SPL, MPL, β):** | **(1000, 250, 0.2)** | **(7500, 500, 0.2)** | **(1000, 250, 0.2)** | **(7500, 500, 0.2)** |
| **Due date tightness:** | **SM2 type:** | | | | | |
| **Adjusted** | **Reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 14.08% | 8.33% | 14.08% | 8.33% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 14.08% | 8.33% | 15.49% | 13.33% |
| | **Non-reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 35.21% | 21.66% | 35.21% | 21.66% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 14.08% | 8.33% | 15.49% | 13.33% |
| | **Partially Reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 22.53% | 15% | 22.53% | 15% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 14.08% | 8.33% | 15.49% | 13.33% |
| **Not Adjusted** | **Reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 32.17% | 9.09% | 32.17% | 9.09% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 4.34% | 4.54% | 40% | 15.45% |
| | **Non-reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 45.21% | 22.72% | 45.21% | 22.72% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 4.34% | 4.54% | 40% | 15.45% |
| | **Partially Reactive** | $\frac{MultiPass - BestPerf}{BestPerf} \times 100$ | 39.13% | 14.54% | 39.13% | 14.54% |
| | | $\frac{LearnPerf - BestPerf}{BestPerf} \times 100$ | 4.34% | 4.54% | 40% | 15.45% |

remaining 2 cases, *MultiPass* is closer to *BestPerf* more than *LearnPerf* only 0.9%, which means they are almost equal.

In the experiments, we also keep track of the statistics about the number of updates of learning trees and control charts. Tables F.2 and F.3 in the appendix summarizes these statistics. Furthermore, we analyze the detailed output reports about the exact timing of these updates of the learning tree and the control charts. From these reports, we observe that when the parameters in the actual manufacturing environment changes, the reconstruction process of the learning tree and the control charts accelerates. This continues for a while until the new charts and the learning tree stabilizes. Therefore, the total number of updates for both the learning tree and the control charts increase.

To illustrate the operations of the control charts, update signals of the learning tree and the control charts, we plot a portion of the control charts in Figure 4.12 and 4.13. The plotted data is taken from the experiment with the following conditions: training data set full; SM2 type reactive; due date tightness adjusted; (SPL, MPL, β) is (7500 , 500, 0.2); DR set is {MDD, ODD, SPT}. The actual data points are given in Appendix F.

In these figures, the extreme points (i.e., the points outside the chart limits) are designated by the signals, which trigger the need for the learning tree update. In addition, at R-signal 4 the control charts are also updated because at that point *two successive learning tree update signal* is received (i.e., learning tree is updated just in the previous point and at that point). Furthermore, at the same time we receive X-signal 4 and R-signal 6, where both the learning tree and the control charts are updated. Note that, each plotted data point in the charts come from the aggregation of the data points (average tardiness values observed) of five scheduling periods. Thus, when X-signal 1 and R-signal 1 signal the first update of the learning tree at point 12, we have 12*5=60 new instance data points available for the

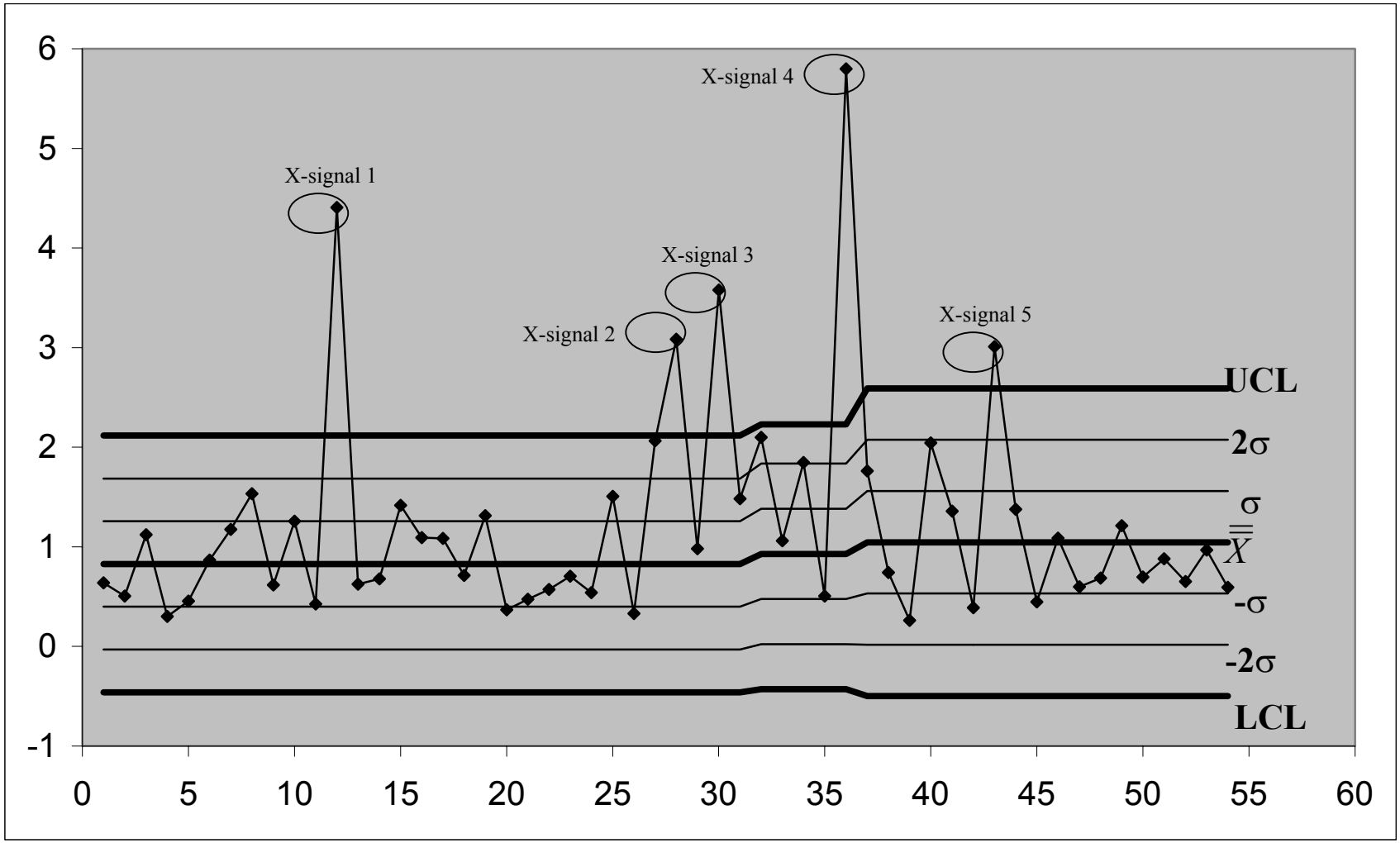



Figure 4.12: X chart

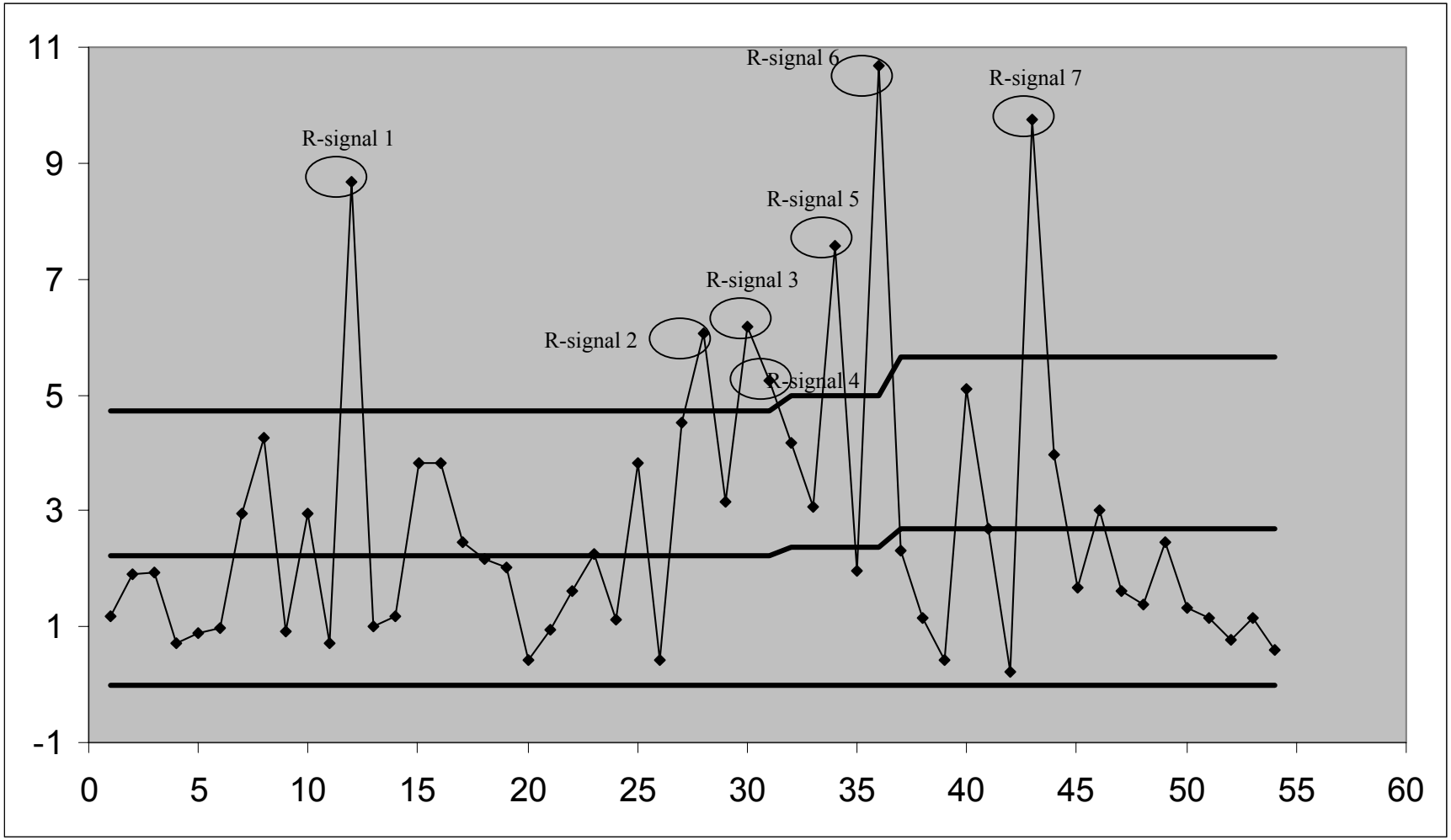



Figure 4.13: R chart

construction of the new learning tree. Hence, the first update of the learning tree takes place after 60 scheduling periods since the system is initiated.

From these charts, we can see how our system adapts itself to the changes in time. Signal types used in the proposed system are based on the statistical principles in which the probability of each signal is very low if there is no change in the actual manufacturing environment. Hence, detecting any of these signals is due to the changes occurred in the actual manufacturing environment and hence the appropriate actions (i.e., updating the learning tree and the control charts) are taken. For example, detecting two consecutive learning tree update signals have a very low probability when there isn't any change in the actual manufacturing system. Therefore, detecting such a signal triggers the reconstruction of the control charts, since the reason of such a signal can most probably be the lost of validity of the existing charts. Also, as it is understand from the example, our system considers updating the learning tree in the first place, which is a direct result of the definitions of the signal types (see Chapter 3). If updating the learning tree doesn't bring the process under control, then updating the control charts takes place. This is due to the fact that when updating the learning tree, we provide new training instances to the learning algorithm and expect to have a new tree that performs within the limits of the existing control charts. However, if updating the learning tree is not sufficient for being within the limits of the control, then it is concluded that the process parameters have shifted and the charts are need to be updated. Our system survives in time by the help of this capability and yields very promising results.

To sum up, the proposed learning-based scheduling system performs well in the experiments. It also outperforms the simulation-based scheduling significantly in most of the cases. Monitoring the performance of the learning tree by the control charts not only improve

the performance of the system but also let the system to survive in time without any external manipulations. However, the simulation-based scheduling requires external manipulations such as data collection, statistical analysis of data and modifying the existing simulation model when the SM2 type is reactive or partially reactive.

## 4.8. Summary

We begin this chapter with the experiments on important parameters of the scheduling problem such as the scheduling and monitoring period lengths. In these experiments, we show the importance of selecting the appropriate values for these parameters. After these experiments, we also restrict our dispatching rule set with the rules that have close single-pass performances and perform the rest of the experiments with this restricted rule set (i.e., {MDD, ODD, SPT}) as well as the original rule set (i.e., {MOD, MDD, ODD, SPT}). Before experimenting on our proposed system with these DR sets, we perform experiments on selecting the system attributes to embody into our learning-based scheduling system. For this, we follow a heuristic approach and determine the system attributes. After that point, we begin to experiment on our proposed system. As a first step, we consider the learning tree as static. In other words, once the learning tree is constructed, it is never updated again. In these experiments, we assume that the manufacturing system does not undergo any changes such as a change in the arrival rate or the service rate. The results of the experiments are compared with the best performance values that can be achieved as well as the simulation-based scheduling performances. Results show that the learning-based scheduling outperforms the simulation-based scheduling approach. Furthermore, the performance values achieved are not larger than the best values. Also, *LearnPerf* is always found to be better than the single-pass performances of individual dispatching rules.

In the last section of this chapter, we finally experiment on our proposed system as it is presented in Chapter 3. We employ the control charts and update the learning tree as well as the charts' themselves whenever it is signaled. In these experiments, we assume that the conditions of the manufacturing system change from one horizon to another. In the experiments we only consider a change in the arrival rate. Results show that the proposed system outperforms the simulation-based scheduling approach and provide significantly close values to the best performances that can be achieved.

# Chapter 5

# Conclusion and Future Research Directions

In this thesis, we presented a learning-based scheduling system for a classical job shop problem. C4.5 algorithms, which are developed by Quinlan (1993), are used for the learning process to construct the learning tree. Process control charts are also employed in the proposed system to continuously monitor the performance of the system so that it adapts itself to the changes in the manufacturing environment (without any external manipulations). In the next section, we discuss our contributions and in Section 5.2, we give some future research directions.

## 5.1. Contributions

In Chapter 4, we conducted extensive computational experiments to fine tune up the parameters (e.g., monitoring period, scheduling period length) and to understand their impacts on the system performance (i.e., average tardiness). Our results indicate that scheduling period length plays a critical role as it significantly affects the system performance. Specifically, the system performance is worst when short SPLs are used. This is due to the fact that for small SPL, even though the selected rules seem to be the best for these short scheduling periods, the system switches back and forth between different rules so frequently that the performance of the system is never stabilized and it deteriorates in the long run.

Moreover, for the loose due dates, the system performance does not differ much for different scheduling period lengths that are larger than a threshold value. That is, the performance of the system converges to the performance of the single-pass dispatching rules for large value of SPL when the due dates are loose. This is due to the fact that the performances of the individual dispatching rules (MDD, ODD and MOD) are very close to each other in the long run for loose due-dates (as also stated by Baker, 1984), switching between these rules doesn't provide any benefit. Therefore, *BestPerf* converges to a limit (single-pass performance of the rules) showing a behavior of exponential decay function.

In the other case of tight due dates, *BestPerf* displays an exponential decay behavior as we increase SPL, but it reaches a minimum value at some point. At the minimum, the system selects the best rule combination and *BestPerf* reaches to its minimum. But, when we continue to increase the SPL further, the system performance deteriorates and converges to a higher value than the minimum. This higher value is again close to the long-run performance of the most dominant dispatching rule, because system begins to choose this particular rule

most of the time. Note that this increase in tardiness after the minimum point is attributable to the loss of the improvements that can be achieved by switching to different rules during the long scheduling periods.

Our second set of experiments is conducted on the monitoring policy (i.e., MPL and β parameters). In general, monitoring improves the system performance. For short monitoring intervals, best performance is achieved with very small β values. This suggests that we should use a small threshold value (χ) for small monitoring intervals. This also implies that we impose more restriction on the system performance. When we increase the MPL, we observe that the best performance value is achieved with larger β values. This means that, when we allow the system to have higher performance values at the monitoring points, we get better performance values.

In summary, it is very important to select the appropriate values for SPL, MPL and β. For poorly selected parameters, performance of multi-pass methods can be worse than the single-pass performances of the individual dispatching rules.

After all these preliminary analyses of the system parameters, we measured the performance of the proposed system in two stages. First we used a static learning tree, (i.e., the learning tree is not updated in time). In this set of experiments, our proposed system first constructs the learning tree at the beginning of a planning horizon and it is used throughout the planning horizon for selecting the dispatching rules. The results indicated that the proposed system performs better than the simulation-based multi-pass scheduling and the single-pass scheduling. But for very large values of monitoring intervals, the performance of the proposed system deteriorates to the level of the performance of the multi-pass scheduling system. Hence, at this point we conclude that the monitoring process is really essential for our learning based algorithm. Moreover, when we add a competitive dispatching rule (i.e.,

MOD) to our rule set, the performance of the proposed system as well as the *MultiPass* further improves (gets closer to *BestPerf*). Hence, deciding the rules in the candidate dispatching rule set is also important for our learning based algorithm. Alternatively, we expect that the gap between *LearnPerf* and BestPerf as well as the gap between *MultiPass* and *BestPerf* increase when we add low-quality DR to the candidate dispatching rule set.

In the second stage, we conduct experiments with the proposed system with a dynamic learning structure. In this experimental setting, the control charts are used to monitor the performance of the learning tree and the tree is updated whenever necessary. In these simulation experiments, we consider the system in which the parameters of the manufacturing system (e.g., arrival rate) change in time (as in the case of real life). The results show that significant improvements are achieved by our proposed system when the manufacturing system parameters change from one planning horizon to other.

In such a realistic environment, even though the simulator of multi-pass scheduler is updated for these parameter changes, our proposed system still gives better results. Since our system adapts itself to the parameter changes automatically, we eliminate the external work required such as data collection and statistical analyses of the collected data. However, this external works are required for the multi-pass scheduling algorithm to update the simulation model. Furthermore, we showed that when the candidate dispatching rule set contains a competitive dispatching rule such as MOD, both our system and the multi-pass algorithm resulted in closer values to the best performance. Therefore, it is vital to select high quality dispatching rules and set the system parameters (i.e., SPL, MPL and $\beta$) appropriately to attain better performance. Finally, since our system selects dispatching rules from the learning tree automatically (i.e., on-line), we also eliminated the extensive simulation experiments that should be required for simulation-based multi-pass scheduling approach.

## 5.2. Future Research

In today's highly competitive business environment, product variety of a firm tends to increase due to the demand for highly customized goods, which in turn increases the complexity of operating a manufacturing system. In addition to these, the demand patterns of commodities may also change too rapidly. Especially for high tech industries, the product life cycles become very short and the customer demand can change drastically due to the introduction of new technologies in the market (i.e., introduction by the competitors). In this research, we developed a scheduling system that survives in time and handles the scheduling operations in such a changing manufacturing environment. We tested our proposed system when the arrival rate parameter changes in time. One possible future research topic can be to test the proposed system for other parameter changes, such as the shifts in processing time parameter. Moreover, it can be tested in a manufacturing system with machine breakdowns. Also, the behavior of the system performance as a function of SPL and MPL can be further analyzed by considering machine breakdowns.

Another research direction could be to combine the capabilities of multi-pass scheduling and the proposed learning-based system. For example, the decisions of the learning-based system can also be tested via simulation prior to use and some corrective actions can be applied (i.e., altering the recommendation of the learning tree).

Another possible research area may be to develop a more sophisticated learning system. For example, a second, high-level learning can be developed upon the proposed learning structure. In the current implementation, we update our learning tree each time it is signaled and the old tree is trashed. However, a high-level learning that also learns on the characteristics of the constructed learning trees may provide further insights to the problem.

In other words, such a system may detect the general patterns behind all the constructed learning trees and provide valuable information about the problem.

Finally, in some manufacturing systems there might be a high implementation cost for switching between the rules frequently. Another research direction can be to incorporate this cost factor into the objective function.

# Bibliography

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

DeVor, R. E., Chang, T-h., Sutherland, J. W., *Statistical Quality Design and Control*, Prentice Hall, New Jersey, 1992.

Duncan, A. J., *Quality Control and Industrial Statistics*, Irwin, Illinois, 1986.

Huyet, A.L., Paris, J.L., 2003, Synergy between evolutionary optimization and induction graphs learning for simulated manufacturing systems, *Working Paper*.

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

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

Jeong, K. –C. and Kim, Y.-D., 1998, A real-time scheduling mechanism for a flexile manufacturing system: using simulation and dispatching rules. *International Journal of Production Research,* **36**, 2609-2626.

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

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

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

Quinlan, J.R., *C4.5 Programs for Machine Learning*, Morgan Kaufmann, California, 1993.

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

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

Suwa, H., Fujii, Susumu, 2003, Rule acquisition for rolling horizon heuristics in single machine dynamic scheduling, _____.

Takahashi, K., Nakamura, N., 1999, Reacting JIT ordering systems to the unstable changes in demand. *International Journal of Production Research*, **37**, 2293-2313.

Takahashi, K., Nakamura, N., 2002, Decentralized reactive Kanban system. *Eurapean Journal of Operational Research*, **139**, 262-276.

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

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

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

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

# APPENDIX A

# Construction Methods of Control Charts

For $n$ samples where each sample has size $m$:

1. For each sample, an average is calculated:

$\bar{X}_k = \frac{\sum_{j=1}^{m} X_{kj}}{m}$, where $X_{kj}$ are from Performance Value column of Figure-6 and $X_{kj} = X_i^*$ where j represents any Realised Scheduling Period for which $X_i^*$ is obtained.

2. The spread or dispertion within the $k^{th}$ sample is measured by the range $R_k$ :

$R_k = X_{largest} - X_{smallest}$ where $X_{largest}, X_{smallest} \in k^{th}\ sample$

3. The Grand Average, $\bar{\bar{X}}$, is an estimate of the process mean and becomes the centerline of the $\bar{X}$ chart:

$$\bar{\bar{X}} = \frac{\sum_{k=1}^{n} \bar{X}_k}{n}$$

4. The average of the sample ranges $\bar{R}$ :

$$\bar{R} = \frac{\sum_{k=1}^{n} R_i}{n}$$

5. The true range of samples of size $m$ is related to the standard deviation of the population (process) by the formula:

$$\frac{E(R)}{\sigma_X} = d_2$$

where $d_2$ is a function of the sample size under an assumed normal distribution of $\bar{X}_k$'s. For values $d_2$ of for varying sample sizes $m$ (see for example DeVor et. al.,1992)

6. Control Limits for $\bar{X}$ chart:

$$\sigma_{\bar{X}} = \frac{\sigma_X}{\sqrt{m}}$$

$$\hat{\sigma}_{\bar{X}} = \frac{\bar{R}}{d_2\sqrt{m}}$$

$$UCL = \bar{\bar{X}} + 3\frac{\bar{R}}{d_2\sqrt{m}} \qquad LCL = \bar{\bar{X}} - 3\frac{\bar{R}}{d_2\sqrt{m}}$$

7. Control Limits for $R$ chart:

$UCL = D_4 \times \bar{R}$ $\qquad$ $LCL = D_3 \times \bar{R}$

where values of $D_3$ and $D_4$ can be found in most of the quality books (see for example DeVor et. al.,1992)

# APPENDIX B

# Notation and Definition of the System Attributes

## Notation

$M$ : set of machines, $M = \{1, 2, 3, 4\}$.

$m$ : number of machines in the manufacturing system, which is equal to cardinality of $M$, $|M|$.

$Q$ : set of queues in front of the machines, $Q = \{1, 2, 3, 4\}$.

$I_q$ : set of jobs in the queue $q$ at time $t$, $q \in Q$.

$O_m$ : set of operations of all jobs in the system that is to be processed on machine $m$, $m \in M$.

$I_t$ : set of jobs in the system at time $t$.

$n$ : cardinality of $I_t$, that is the number of jobs in the system at time $t$.

$J_i$ : set of all operations of job $i$, $i \in I$.

$\hat{J}_i$ : set of all remaining operations of job $i$, $i \in I$, $\hat{J} \bullet J$.

$r_i$ : release time of job $i$, $i \in I$.

$p_{ij}$ : processing time of operation $j$ of job $i$, $i \in I$, $j \in J$.

$p_{i,cur}$ : processing time of job i at the machine just after its current queue, $i \in I$, $j \in J$.

$i_{<j>}$ : machinery location of $j^{th}$ operation of job $i$, $i \in I$, $j \in J$.

$d_i$ : due date of job $i$, $i \in I$.

$d_{ij}$ : due date of operation $j$ of job $i$, $i \in I$, $j \in J$.

$o_i$ : number of operations of job $i$, $i \in I$.

$k$ : flow allowance factor.

$SPL$ : Scheduling period length

$MPL$ : Monitoring period length

## Definition of the System Attributes

**Attribute-1) Number of Customers in the System (NumCust)**

This attribute stores the number of customer in the system at time *t*. Hence, its value is equal to cardinality of $I_t$, which is *n*.

**Attribute-2) Percentage of Maximum Relative Machine Workload (PMaxRMW)**

This attribute is calculated as the following: for each machine, we find the total processing times of the operations that will be performed on that machine. Then we take the maximum of these values and divide it to total remaining processing time and multiply by 100.

$$PMaxRMW = \frac{Max_{q \in Q}\{\sum_{i \in I_q} p_{i,cur}\}}{\sum_{i \in I_t}\sum_{j \in \hat{J}} p_{ij}} \times 100$$

**Attribute-3) Percentage of Completed Processing Times (PCompPT)**

This attribute is calculated by dividing the total completed processing time to the total processing time and multiplying by 100.

$$PCompPT = \frac{\sum_{i \in I_t}\sum_{j \in J-\hat{J}} p_{ij}}{\sum_{i \in I_t}\sum_{j \in J} p_{ij}} \times 100$$

**Attribute-4) Relative Tightness Ratio (RTR)**

This attribute is simply the ratio of average flow allowance to the average remaining processing times of the jobs. RTR is calculated as follows:

$$RTR = \frac{\bar{d}_t}{\bar{p}},$$

$$where, \bar{d}_t = \max\{0, \frac{\sum_{i \in I_t}(d_i - t)}{n}\}, \bar{p} = \frac{\sum_{i \in I_t}\sum_{j \in \hat{J}} p_{ij}}{n}$$

**Attribute-5) Rule Updating Signal (RUS)**

In Section 3.3.2, new rule selection symptoms for updating the current DRs are discussed. Whenever one of these symptoms is detected, a new DR is requested from the learning module to continue the scheduling operations. Hence, it may be a good idea to provide this information to the learning module as an attribute. RUS is a discrete type attribute and takes its values as follows:

$$RUS = \begin{cases} 0 \text{ if the signal is BSP} \\ 1 \text{ if the signal is MP} \end{cases}$$

**Attribute-6) Total Remaining Processing Time (TotRemPT)**

This attribute, as it is clear from its name, stores the total remaining processing time of the jobs that are in the system at time *t*.

$$Tot\,\mathrm{Re}\,mPT = \sum_{i \in I_t}\sum_{j \in \hat{J}} p_{ij}$$

**Attribute-7) Average Remaining Processing Time (AvRemPT)**

This attribute is equal to the total remaining processing time divided by the number of jobs in the system and calculated as follows:

$$Av\,\mathrm{Re}\,mPT = \frac{Tot\,\mathrm{Re}\,mPT}{n} = \frac{\sum_{i \in I_t}\sum_{j \in \hat{J}} p_{ij}}{n}$$

**Attribute-8) Total Slack Time (AverageSlackT)**

This attribute stores the value of total slack times minus the total remaining processing time of all the jobs in the system divided by the number of jobs. Mathematical formulation is as follows:

$$AverageSlackT = \frac{\sum_{i \in I}(d_i - t) - \sum_{i \in I}\sum_{j \in J} p_{ij}}{n}$$

**Attribute-9) Average Period Queue Length (AvPerQL)**

This attribute stores the average queue length of all queues in the last scheduling period. That is, if the current time is *t* and $QT_q(t - SPL, t)$ is the total queue time of the jobs in queue *q* 0 *Q* between time t-SPL and t, then:

$$AvPerQL = \frac{\sum_{q \in Q} QT(t - SPL, t)}{|Q| \times SPL}$$

**Attribute-10) Maximum Queue Length at Time *t* (MaxQL-t)**

This attribute stores the information about the number of jobs waiting in the longest queue in the system. That is,

$MaxQL - t = \max_{q \in Q}\{|I_q|\}$, where $|I_q|$ is the cardinality of set $I_q$.

**Attribute-11) Average Remaining Time Until Due Dates (AvRemTDd)**

Store the average of remaining time of all jobs until their due dates. Mathematical representation is as follows:

$$Av\,\mathrm{Re}\,mTDd = \frac{\sum_{i\in I}(d_i - t)}{n}$$

**Attribute-12) Number of Jobs with Long Processing Time (NumLongPT)**

This attribute especially defined to separate SPT rule from the other rules. It stores the number of jobs that have a processing time greater than the average processing time of all jobs in the system. The mathematical representation is the following:

$NumLongPT = |L|$ such that $L = \{i | p_i \geq \overline{p}, i \in I_t\}$, where $\overline{p} = \frac{\sum_{i\in I}\sum_{j\in J} p_{ij}}{n}$ and $|L|$ is the cardinality of the set L.

**Attribute-13) Percentage of Jobs with Long Processing Times (PercentLongPT)**

It is calculated by dividing NumLongPT to the number of jobs in the system and multiplying by 100. That is,

$$PercentLongPT = \frac{NumLongPT}{n} \times 100$$

**Attribute-14) Difference between Maximum and Average Processing Times (Max_AvPT)**

It stores the difference between the maximum processing time and the average processing time of the jobs at time *t*. That is,

$$Max\_AvPT = \max_{i\in I}\{\sum_{j\in J} p_{ij}\} - \overline{p}, \text{ where } \overline{p} = \frac{\sum_{i\in I}\sum_{j\in J} p_{ij}}{n}$$

**Attribute-15) Percentage of Difference between Maximum and Average Processing Times (PercentMax_AvPT)**

It is the percentage of Max_AvPT and calculated by dividing Max_AvPT by average processing time and multiplied by 100. That is,

$$PercentMax\_AvPT = \frac{Max\_AvPT}{\bar{p}} \times 100$$

**Attribute-16) Maximum Due Date (MaxDue)**

This attribute stores the value of maximum due date of the jobs minus the current time *t*. In other words, it is the difference between the current time t and the maximum due date of the jobs in the system at time *t*.

$$MaxDue = \max\{0, \max_{i \in I}\{d_i\} - t\}$$

**Attribute-17) Number of Jobs with Long Due Dates (NumberLongDD)**

This attribute especially defined to separate EDD rule from the other rules. It stores the number of jobs that have due dates greater than the average due date of all jobs in the system. The mathematical representation is the following:

$$NumberLongDD = |S|, \text{ where } S = \{i | d_i \geq \bar{d}, i \in I\}, \bar{d} = \frac{\sum_{i \in I} d_i}{n} \text{ and } |S| \text{ is the cardinality of S.}$$

**Attribute-18) Percentage of Jobs with Long Due Dates (PercentLongDD)**

It stores the percentage of jobs that have due dates greater than the average due date of all jobs in the system. The mathematical representation is the following:

$$PerentLongDD = \frac{|S|}{n} \times 100, \text{ where } S = \{i | d_i \geq \bar{d}, i \in I\}, \bar{d} = \frac{\sum_{i \in I} d_i}{n} \text{ and } |S| \text{ is the cardinality of } S.$$

**Attribute-19) Maximum Coefficient of Variation of Processing Times of Machines (MaxCV_PT_Machines)**

To set the value of this attribute at any time *t*, we first form a set for each machine, which is composed of the operations that should be performed on that machine. Then, for each set, we calculate the coefficient of variation of processing times of the operations in the set. Then we set the attribute value to the maximum of these coefficient of variations. That is,

$MaxCV\_PT\_Machines = \max_{m \in M} \{cv_m\}$ where,

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

**Attribute-20) Average of Coefficient of Variations of the Job Processing Times (Mean_CVPT)**

To set the value of this attribute at time *t*, we calculate the coefficient of variation of operation processing times of each individual job in the system. Then we take the average of these $cv_i$ values and set it as the value of our attribute. The mathematical formulation is the following:

$$Mean\_CVPT = \frac{\sum_{i \in I} cv_i}{n} \text{ where,}$$

$$cv_i = \frac{\overline{p}_i}{\sigma_i}, \overline{p}_i = \frac{\sum_{j} p_{ij}}{|o_i|}, \sigma_i = \sqrt{\frac{\sum_{j} (p_{ij} - \overline{p}_i)^2}{|o_i| - 1}}$$

**Attribute-21) Average of Variances of the Job Processing Times (Mean_VarPT)**

It is quite similar to attribute-20, but instead of taking the average of coefficient of variations, we take the average of the variances. That is,

$$Mean\_VarPT = \frac{\sum_{i \in I} \sigma_i^2}{n} \text{ where,}$$

$$\sigma_i^2 = \frac{\sum_j (p_{ij} - \overline{p}_i)^2}{|o_i| - 1}, \overline{p}_i = \frac{\sum_j p_{ij}}{|o_i|}$$

**Attribute-22) Maximum of Coefficient of Variations of the Job Processing Times (Max_CVPT)**

As it is in the calculations of Attribute-20, we calculate the coefficient of variation of operation processing times of each individual job in the system and set Max_CVPT to the maximum of these values.

$$Max\_CVPT = \max_{i \in I} \{cv_i\} \text{ where,}$$

$$cv_i = \frac{\overline{p}_i}{\sigma_i}, \overline{p}_i = \frac{\sum_j p_{ij}}{|o_i|}, \sigma_i = \sqrt{\frac{\sum_j (p_{ij} - \overline{p}_i)^2}{|o_i| - 1}}$$

**Attribute-23) Difference between Maximum and Average of Coefficient of Variations of the Job Processing Times (Diff_Max/Mean_CVPT)**

This attribute is simply the difference of Attribute-22 and Attribute-20. That is,

$$Diff\_Max/Mean\_CVPT = (Max\_CVPT) - (Mean\_CVPT) = \max_{i \in I} \{cv_i\} - \frac{\sum_{i \in I} cv_i}{n}$$

**Attribute-24) Number of Jobs Already Tardy (Num_AlreadyTardy)**

If the total remaining processing time of any job plus the current time *t* exceeds the due date of that job, we are certain that this job will be tardy. The number of such jobs at time *t* is set as the value of that attribute. Hence,

$Num\_AlreadyTardy = |S|$ where,

$S = \{i | \sum_{j \in \hat{J}} p_{ij} > d_i - t, i \in I_t\}$ and $|S|$ is the cardinality of set *S*.

**Attribute-25) Average Remaining Processing Time of Already Tardy Jobs (AveragePT_AlreadyTardy)**

Referring to the discussion of the previous attribute, we calculate the average remaining processing time of the jobs that we certainly know as tardy and record this value to this attribute. Hence, mathematically,

$$AveragePT\_AlreadyTardy = \frac{\sum_{i \in S} \sum_{j \in \hat{J}} p_{ij}}{|S|}$$ where,

$S = \{i | \sum_{j \in \hat{J}} p_{ij} > d_i - t, i \in I_t\}$ and $|S|$ is the cardinality of set *S*.

**Attribute-26) Last Period's Average Tardiness Value (MeanTardiness)**

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

# APPENDIX C

# Results for Scheduling Period Length

**Table C.1: 80% utilization, tight due-dates replication mean tardiness values (Ri: replication i, SPL: scheduling period length, DR set: SPT, MDD, MOD, ODD)**

| SPL | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 0.649 | 0.708 | 0.62 | 0.581 | 0.667 | 0.641 | 0.579 | 0.645 | 0.569 | 0.709 | 0.712 |
| **100** | 0.589 | 0.599 | 0.536 | 0.493 | 0.575 | 0.622 | 0.577 | 0.608 | 0.466 | 0.548 | 0.563 |
| **200** | 0.435 | 0.575 | 0.532 | 0.461 | 0.542 | 0.486 | 0.583 | 0.525 | 0.551 | 0.47 | 0.455 |
| **500** | 0.438 | 0.399 | 0.44 | 0.447 | 0.418 | 0.5 | 0.379 | 0.414 | 0.409 | 0.378 | 0.355 |
| **1000** | 0.42 | 0.37 | 0.385 | 0.394 | 0.425 | 0.382 | 0.46 | 0.32 | 0.365 | 0.338 | 0.436 |
| **2000** | 0.384 | 0.36 | 0.391 | 0.434 | 0.441 | 0.42 | 0.385 | 0.372 | 0.414 | 0.396 | 0.356 |
| **5000** | 0.405 | 0.376 | 0.343 | 0.441 | 0.344 | 0.451 | 0.406 | 0.364 | 0.451 | 0.421 | 0.351 |
| **7500** | 0.378 | 0.436 | 0.374 | 0.424 | 0.38 | 0.362 | 0.386 | 0.426 | 0.466 | 0.377 | 0.499 |
| **10000** | 0.396 | 0.456 | 0.376 | 0.36 | 0.39 | 0.379 | 0.377 | 0.448 | 0.435 | 0.374 | 0.442 |
| **12500** | 0.388 | 0.361 | 0.404 | 0.408 | 0.436 | 0.332 | 0.399 | 0.383 | 0.439 | 0.406 | 0.456 |
| **15000** | 0.388 | 0.403 | 0.412 | 0.338 | 0.451 | 0.324 | 0.388 | 0.377 | 0.439 | 0.424 | 0.366 |

| SPL | R12 | R13 | R14 | R15 | R16 | R17 | R18 | R19 | R20 | MEAN | STD. DEV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 0.543 | 0.634 | 0.756 | 0.705 | 0.692 | 0.565 | 0.621 | 0.633 | 0.631 | 0.643 | 0.057954 |
| **100** | 0.555 | 0.666 | 0.505 | 0.604 | 0.548 | 0.582 | 0.58 | 0.628 | 0.545 | 0.56945 | 0.047788 |
| **200** | 0.465 | 0.524 | 0.496 | 0.522 | 0.466 | 0.543 | 0.57 | 0.485 | 0.433 | 0.50595 | 0.046796 |
| **500** | 0.407 | 0.431 | 0.42 | 0.466 | 0.391 | 0.45 | 0.399 | 0.411 | 0.39 | 0.4171 | 0.033659 |
| **1000** | 0.416 | 0.368 | 0.333 | 0.423 | 0.358 | 0.403 | 0.362 | 0.336 | 0.382 | 0.3838 | 0.038087 |
| **2000** | 0.422 | 0.431 | 0.413 | 0.393 | 0.422 | 0.341 | 0.412 | 0.447 | 0.378 | 0.4006 | 0.029726 |
| **5000** | 0.38 | 0.409 | 0.464 | 0.372 | 0.511 | 0.348 | 0.4 | 0.379 | 0.42 | 0.4018 | 0.045425 |
| **7500** | 0.397 | 0.393 | 0.361 | 0.4 | 0.381 | 0.371 | 0.393 | 0.425 | 0.398 | 0.40135 | 0.035435 |
| **10000** | 0.39 | 0.417 | 0.389 | 0.405 | 0.357 | 0.41 | 0.452 | 0.392 | 0.427 | 0.4036 | 0.030855 |
| **12500** | 0.41 | 0.379 | 0.393 | 0.371 | 0.459 | 0.451 | 0.435 | 0.374 | 0.357 | 0.40205 | 0.035413 |
| **15000** | 0.415 | 0.426 | 0.366 | 0.389 | 0.429 | 0.435 | 0.444 | 0.362 | 0.394 | 0.3985 | 0.035585 |

**Table C.2: 90% utilization, tight due-dates replication mean tardiness values (Ri: replication i, SPL: scheduling period length, DR set: SPT, MDD, MOD, ODD)**

| SPL | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 1.978 | 2.425 | 2.348 | 1.766 | 2.362 | 2.067 | 1.97 | 2.309 | 1.76 | 2.32 | 2.572 |
| **100** | 1.599 | 1.651 | 1.584 | 1.314 | 1.39 | 1.758 | 1.471 | 1.427 | 1.161 | 1.431 | 1.357 |
| **200** | 1.255 | 1.457 | 1.585 | 1.036 | 1.501 | 0.911 | 1.812 | 1.351 | 1.38 | 1.115 | 1.082 |
| **500** | 1.071 | 0.922 | 1.09 | 0.995 | 0.953 | 1.346 | 0.913 | 0.988 | 0.909 | 0.815 | 0.75 |
| **1000** | 0.919 | 0.754 | 0.812 | 0.947 | 0.815 | 0.633 | 0.824 | 1.105 | 0.912 | 0.605 | 1.225 |
| **2000** | 0.656 | 0.477 | 0.521 | 0.662 | 0.666 | 0.754 | 0.493 | 0.688 | 0.729 | 0.704 | 0.547 |
| **5000** | 0.65 | 0.587 | 0.591 | 0.69 | 0.46 | 0.519 | 0.58 | 0.567 | 0.654 | 0.49 | 0.55 |
| **7500** | 0.504 | 0.433 | 0.569 | 0.504 | 0.602 | 0.471 | 0.569 | 0.478 | 0.635 | 0.529 | 0.586 |
| **10000** | 0.592 | 0.776 | 0.42 | 0.5 | 0.445 | 0.499 | 0.584 | 0.697 | 0.794 | 0.76 | 0.716 |
| **12500** | 0.523 | 0.612 | 0.443 | 0.374 | 0.79 | 0.71 | 0.761 | 0.604 | 0.65 | 0.517 | 0.561 |
| **15000** | 0.561 | 0.621 | 0.696 | 0.534 | 0.922 | 0.772 | 0.506 | 0.406 | 0.495 | 0.632 | 0.441 |

| SPL | R12 | R13 | R14 | R15 | R16 | R17 | R18 | R19 | R20 | MEAN | STD. DEV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 1.518 | 2.33 | 2.017 | 2.197 | 1.999 | 1.678 | 1.81 | 1.879 | 2.23 | 2.07675 | 0.285243 |
| **100** | 1.456 | 2.11 | 1.288 | 1.73 | 1.729 | 1.511 | 1.4 | 1.883 | 1.334 | 1.5292 | 0.228364 |
| **200** | 1.123 | 1.489 | 1.331 | 1.186 | 0.974 | 1.113 | 1.668 | 1.103 | 1.069 | 1.27705 | 0.24725 |
| **500** | 0.854 | 1.151 | 0.943 | 1.317 | 0.748 | 1.009 | 1.418 | 0.839 | 0.801 | 0.9916 | 0.192514 |
| **1000** | 0.807 | 0.673 | 0.852 | 0.706 | 0.73 | 0.755 | 0.615 | 0.763 | 0.796 | 0.8124 | 0.155278 |
| **2000** | 0.727 | 0.509 | 0.592 | 0.618 | 0.634 | 0.553 | 0.697 | 0.779 | 0.525 | 0.62655 | 0.093758 |
| **5000** | 0.783 | 0.562 | 0.554 | 0.86 | 0.537 | 0.492 | 0.691 | 0.731 | 0.497 | 0.60225 | 0.105693 |
| **7500** | 0.679 | 0.629 | 0.553 | 0.502 | 0.79 | 0.618 | 0.659 | 0.526 | 0.543 | 0.56895 | 0.084129 |
| **10000** | 0.621 | 0.582 | 0.534 | 0.568 | 0.48 | 0.451 | 0.621 | 0.498 | 0.588 | 0.5863 | 0.113324 |
| **12500** | 0.749 | 0.745 | 0.651 | 0.407 | 0.631 | 0.54 | 0.472 | 0.577 | 0.499 | 0.5908 | 0.121374 |
| **15000** | 0.744 | 0.525 | 0.52 | 0.549 | 0.731 | 0.569 | 0.718 | 0.782 | 0.633 | 0.61785 | 0.13136 |

**Table C.3: 80% utilization, loose due-dates replication mean tardiness values (Ri: replication i, SPL: scheduling period length, DR set: SPT, MDD, MOD, ODD)**

| SPL | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 0.031 | 0.054 | 0.041 | 0.048 | 0.035 | 0.037 | 0.047 | 0.04 | 0.036 | 0.066 | 0.043 |
| **100** | 0.018 | 0.019 | 0.014 | 0.015 | 0.014 | 0.027 | 0.025 | 0.02 | 0.012 | 0.014 | 0.024 |
| **200** | 0.023 | 0.014 | 0.018 | 0.012 | 0.018 | 0.022 | 0.016 | 0.012 | 0.023 | 0.015 | 0.011 |
| **500** | 0.006 | 0.006 | 0.009 | 0.008 | 0.004 | 0.004 | 0.003 | 0.007 | 0.007 | 0.003 | 0.003 |
| **1000** | 0.005 | 0.005 | 0.003 | 0.002 | 0.015 | 0.003 | 0.003 | 0.008 | 0.004 | 0.002 | 0.003 |
| **2000** | 0.005 | 0.004 | 0.003 | 0.004 | 0.004 | 0.003 | 0.004 | 0.003 | 0.003 | 0.003 | 0.003 |
| **5000** | 0.003 | 0.004 | 0.003 | 0.005 | 0.003 | 0.004 | 0.012 | 0.003 | 0.004 | 0.004 | 0.004 |
| **7500** | 0.004 | 0.004 | 0.003 | 0.013 | 0.004 | 0.004 | 0.005 | 0.003 | 0.003 | 0.004 | 0.005 |
| **10000** | 0.004 | 0.004 | 0.004 | 0.005 | 0.004 | 0.004 | 0.004 | 0.005 | 0.004 | 0.003 | 0.005 |
| **12500** | 0.005 | 0.005 | 0.006 | 0.012 | 0.008 | 0.004 | 0.005 | 0.004 | 0.003 | 0.005 | 0.004 |
| **15000** | 0.004 | 0.004 | 0.004 | 0.004 | 0.005 | 0.004 | 0.004 | 0.004 | 0.008 | 0.004 | 0.004 |

| SPL | R12 | R13 | R14 | R15 | R16 | R17 | R18 | R19 | R20 | MEAN | STD. DEV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 0.033 | 0.038 | 0.065 | 0.037 | 0.038 | 0.052 | 0.032 | 0.049 | 0.049 | 0.04355 | 0.010092 |
| **100** | 0.029 | 0.027 | 0.022 | 0.024 | 0.022 | 0.019 | 0.026 | 0.019 | 0.02 | 0.0205 | 0.005021 |
| **200** | 0.013 | 0.02 | 0.014 | 0.014 | 0.012 | 0.019 | 0.017 | 0.016 | 0.01 | 0.01595 | 0.003967 |
| **500** | 0.004 | 0.005 | 0.005 | 0.005 | 0.003 | 0.005 | 0.004 | 0.006 | 0.005 | 0.0051 | 0.001714 |
| **1000** | 0.006 | 0.002 | 0.002 | 0.007 | 0.004 | 0.003 | 0.006 | 0.004 | 0.004 | 0.00455 | 0.003 |
| **2000** | 0.003 | 0.003 | 0.007 | 0.004 | 0.003 | 0.003 | 0.004 | 0.005 | 0.006 | 0.00385 | 0.001137 |
| **5000** | 0.004 | 0.004 | 0.003 | 0.004 | 0.006 | 0.005 | 0.004 | 0.004 | 0.006 | 0.00445 | 0.001986 |
| **7500** | 0.004 | 0.003 | 0.007 | 0.012 | 0.003 | 0.004 | 0.004 | 0.004 | 0.004 | 0.00485 | 0.002777 |
| **10000** | 0.003 | 0.008 | 0.005 | 0.004 | 0.004 | 0.003 | 0.007 | 0.004 | 0.004 | 0.0044 | 0.001231 |
| **12500** | 0.005 | 0.004 | 0.004 | 0.006 | 0.004 | 0.004 | 0.005 | 0.004 | 0.004 | 0.00505 | 0.001959 |
| **15000** | 0.004 | 0.004 | 0.004 | 0.008 | 0.004 | 0.006 | 0.004 | 0.004 | 0.008 | 0.00475 | 0.001482 |

**Table C.4: 90% utilization, loose due-dates replication mean tardiness values (Ri: replication i, SPL: scheduling period length, DR set: SPT, MDD, MOD, ODD)**

| SPL | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 0.604 | 0.628 | 0.758 | 0.564 | 0.676 | 0.728 | 0.664 | 0.609 | 0.419 | 0.888 | 0.782 |
| **100** | 0.139 | 0.217 | 0.344 | 0.14 | 0.148 | 0.126 | 0.302 | 0.18 | 0.158 | 0.217 | 0.131 |
| **200** | 0.056 | 0.146 | 0.07 | 0.124 | 0.061 | 0.084 | 0.07 | 0.035 | 0.058 | 0.064 | 0.047 |
| **500** | 0.06 | 0.03 | 0.04 | 0.06 | 0.037 | 0.033 | 0.153 | 0.03 | 0.053 | 0.035 | 0.058 |
| **1000** | 0.014 | 0.006 | 0.01 | 0.017 | 0.004 | 0.01 | 0.013 | 0.031 | 0.001 | 0.009 | 0.019 |
| **2000** | 0.001 | 0.013 | 0.0006 | 0.0005 | 0.001 | 0.001 | 0.0007 | 0.007 | 0.002 | 0.002 | 0.008 |
| **5000** | 0.0009 | 0.0008 | 0.001 | 0.0007 | 0.0008 | 0.001 | 0.001 | 0.001 | 0.0009 | 0.0007 | 0.0008 |
| **7500** | 0.0008 | 0.0009 | 0.0008 | 0.0006 | 0.017 | 0.0008 | 0.002 | 0.016 | 0.0008 | 0.008 | 0.001 |
| **10000** | 0.003 | 0.0006 | 0.014 | 0.0007 | 0.001 | 0.006 | 0.001 | 0.0009 | 0.0009 | 0.006 | 0.002 |
| **12500** | 0.001 | 0.02 | 0.0008 | 0.002 | 0.0007 | 0.0009 | 0.0009 | 0.0006 | 0.0008 | 0.065 | 0.001 |
| **15000** | 0.001 | 0.0008 | 0.001 | 0.0008 | 0.001 | 0.001 | 0.001 | 0.0009 | 0.001 | 0.0009 | 0.002 |

| SPL | R12 | R13 | R14 | R15 | R16 | R17 | R18 | R19 | R20 | MEAN | STD. DEV |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 0.353 | 0.692 | 0.689 | 0.688 | 0.534 | 0.392 | 0.47 | 0.449 | 0.629 | 0.6108 | 0.140149 |
| **100** | 0.226 | 0.527 | 0.188 | 0.178 | 0.18 | 0.271 | 0.214 | 0.296 | 0.138 | 0.216 | 0.096233 |
| **200** | 0.284 | 0.047 | 0.06 | 0.11 | 0.051 | 0.049 | 0.078 | 0.057 | 0.084 | 0.08175 | 0.055057 |
| **500** | 0.024 | 0.033 | 0.031 | 0.042 | 0.052 | 0.033 | 0.027 | 0.018 | 0.021 | 0.0435 | 0.028743 |
| **1000** | 0.015 | 0.007 | 0.002 | 0.013 | 0.004 | 0.007 | 0.012 | 0.01 | 0.014 | 0.0109 | 0.006828 |
| **2000** | 0.0007 | 0.0006 | 0.0007 | 0.007 | 0.001 | 0.006 | 0.0008 | 0.0006 | 0.002 | 0.00281 | 0.003468 |
| **5000** | 0.003 | 0.001 | 0.001 | 0.035 | 0.001 | 0.0009 | 0.001 | 0.0008 | 0.0008 | 0.002705 | 0.007617 |
| **7500** | 0.028 | 0.008 | 0.001 | 0.0008 | 0.001 | 0.0009 | 0.0009 | 0.0008 | 0.0009 | 0.00455 | 0.00746 |
| **10000** | 0.005 | 0.002 | 0.0007 | 0.0009 | 0.001 | 0.0009 | 0.008 | 0.005 | 0.0008 | 0.00302 | 0.00344 |
| **12500** | 0.018 | 0.003 | 0.001 | 0.0008 | 0.001 | 0.0009 | 0.002 | 0.0009 | 0.0009 | 0.00611 | 0.014921 |
| **15000** | 0.001 | 0.008 | 0.0008 | 0.0009 | 0.001 | 0.004 | 0.0008 | 0.001 | 0.003 | 0.001595 | 0.001718 |

**Table C.5: 80% utilization, tight due-dates replication mean tardiness values (Ri: replication i, SPL: scheduling period length, DR set: SPT, MDD, ODD)**

| SPL | REP1 | REP2 | REP3 | REP4 | REP5 | REP6 | REP7 | REP8 | REP9 | REP10 | REP11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 0.75 | 0.84 | 0.73 | 0.69 | 0.76 | 0.74 | 0.68 | 0.75 | 0.67 | 0.84 | 0.83 |
| **100** | 0.8 | 0.74 | 0.68 | 0.64 | 0.71 | 0.79 | 0.75 | 0.77 | 0.58 | 0.69 | 0.73 |
| **200** | 0.63 | 0.83 | 0.79 | 0.7 | 0.77 | 0.69 | 0.82 | 0.76 | 0.76 | 0.67 | 0.66 |
| **500** | 0.69 | 0.64 | 0.7 | 0.71 | 0.64 | 0.8 | 0.66 | 0.68 | 0.64 | 0.63 | 0.61 |
| **1000** | 0.64 | 0.72 | 0.64 | 0.75 | 0.64 | 0.6 | 0.64 | 0.74 | 0.78 | 0.62 | 0.86 |
| **2000** | 0.68 | 0.64 | 0.74 | 0.77 | 0.8 | 0.74 | 0.69 | 0.68 | 0.75 | 0.73 | 0.66 |
| **5000** | 0.78 | 0.71 | 0.67 | 0.84 | 0.67 | 0.85 | 0.76 | 0.7 | 0.86 | 0.8 | 0.69 |
| **10000** | 0.8 | 0.9 | 0.75 | 0.71 | 0.79 | 0.76 | 0.75 | 0.86 | 0.87 | 0.77 | 0.89 |
| **12500** | 0.79 | 0.73 | 0.82 | 0.8 | 0.84 | 0.69 | 0.8 | 0.78 | 0.87 | 0.84 | 0.9 |
| **15000** | 0.82 | 0.83 | 0.83 | 0.69 | 0.89 | 0.66 | 0.78 | 0.78 | 0.88 | 0.88 | 0.75 |
| **20000** | 0.78 | 0.69 | 0.74 | 0.79 | 0.82 | 0.89 | 0.88 | 0.8 | 0.86 | 0.88 | 0.79 |
| **25000** | 0.84 | 0.84 | 0.88 | 0.75 | 0.89 | 0.77 | 0.85 | 0.89 | 0.86 | 0.85 | 0.84 |

| SPL | REP12 | REP13 | REP14 | REP15 | REP16 | REP17 | REP18 | REP19 | REP20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 0.62 | 0.74 | 0.87 | 0.81 | 0.83 | 0.67 | 0.71 | 0.73 | 0.75 | 0.7505 | 0.068015 |
| **100** | 0.7 | 0.82 | 0.65 | 0.8 | 0.72 | 0.72 | 0.73 | 0.8 | 0.7 | 0.726 | 0.061422 |
| **200** | 0.66 | 0.79 | 0.7 | 0.74 | 0.67 | 0.75 | 0.78 | 0.71 | 0.62 | 0.725 | 0.06245 |
| **500** | 0.66 | 0.71 | 0.68 | 0.76 | 0.66 | 0.7 | 0.63 | 0.7 | 0.64 | 0.677 | 0.046578 |
| **1000** | 0.67 | 0.65 | 0.65 | 0.67 | 0.65 | 0.62 | 0.68 | 0.73 | 0.64 | 0.6795 | 0.064683 |
| **2000** | 0.75 | 0.76 | 0.73 | 0.68 | 0.74 | 0.6 | 0.76 | 0.78 | 0.69 | 0.7185 | 0.051224 |
| **5000** | 0.74 | 0.73 | 0.79 | 0.88 | 0.7 | 0.92 | 0.72 | 0.78 | 0.7 | 0.7645 | 0.074231 |
| **10000** | 0.77 | 0.82 | 0.77 | 0.79 | 0.72 | 0.84 | 0.89 | 0.76 | 0.86 | 0.8035 | 0.058784 |
| **12500** | 0.84 | 0.76 | 0.81 | 0.74 | 0.91 | 0.89 | 0.84 | 0.79 | 0.73 | 0.8085 | 0.059936 |
| **15000** | 0.82 | 0.85 | 0.77 | 0.8 | 0.85 | 0.87 | 0.9 | 0.75 | 0.79 | 0.8095 | 0.064927 |
| **20000** | 0.85 | 0.81 | 0.86 | 0.69 | 0.79 | 0.92 | 0.93 | 0.87 | 0.93 | 0.8285 | 0.07125 |
| **25000** | 0.78 | 0.88 | 0.98 | 0.88 | 0.8 | 0.82 | 0.75 | 0.8 | 0.93 | 0.844 | 0.058974 |

**Table C.6: 90% utilization, tight due-dates replication mean tardiness values (Ri: replication i, SPL: scheduling period length, DR set: SPT, MDD, ODD)**

| SPL | REP#1 | REP#2 | REP#3 | REP#4 | REP#5 | REP#6 | REP#7 | REP#8 | REP#9 | REP#10 | REP#11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 2.02 | 2.53 | 2.41 | 1.86 | 2.38 | 2.18 | 2.01 | 2.26 | 1.74 | 2.51 | 2.61 |
| **100** | 1.68 | 1.7 | 1.62 | 1.41 | 1.45 | 1.9 | 1.58 | 1.61 | 1.23 | 1.47 | 1.52 |
| **200** | 1.52 | 1.09 | 1.04 | 1.57 | 1.59 | 1.08 | 1.01 | 0.97 | 1.68 | 1.13 | 1.19 |
| **500** | 1.47 | 0.78 | 1.37 | 1.44 | 1.11 | 1.11 | 2.15 | 1.16 | 1.39 | 1.18 | 1.56 |
| **1000** | 1.26 | 0.82 | 1.38 | 1.25 | 1.47 | 1.36 | 1.17 | 1.04 | 1.04 | 1.51 | 1.43 |
| **2000** | 1.07 | 1.45 | 1.68 | 1.42 | 1.03 | 1.06 | 0.97 | 1.54 | 1.54 | 1.45 | 1.3 |
| **7500** | 1.13 | 1.26 | 1.28 | 0.93 | 1.38 | 1.34 | 1.27 | 0.85 | 1.18 | 1.26 | 1.28 |
| **10000** | 1.41 | 0.76 | 1.41 | 1.23 | 1.1 | 1.6 | 0.92 | 1.19 | 1.13 | 1.2 | 0.74 |
| **15000** | 1.44 | 1.66 | 1.52 | 1.25 | 1.46 | 1.13 | 1.32 | 1.57 | 1.56 | 1.57 | 1.17 |
| **20000** | 0.99 | 1.34 | 1.55 | 1.16 | 1.28 | 1.12 | 1.31 | 1.35 | 1.7 | 1.65 | 1.27 |
| **25000** | 1 | 1.09 | 0.95 | 1.15 | 1.47 | 2.03 | 1.34 | 1.25 | 1.18 | 1.93 | 1.17 |

| SPL | REP#12 | REP#13 | REP#14 | REP#15 | REP#16 | REP#17 | REP#18 | REP#19 | REP#20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **50** | 1.57 | 2.21 | 2.09 | 2.23 | 2.04 | 1.68 | 1.8 | 1.93 | 2.3 | 2.118 | 0.295913 |
| **100** | 1.66 | 2.26 | 1.46 | 1.83 | 1.9 | 1.56 | 1.58 | 1.97 | 1.34 | 1.6365 | 0.241449 |
| **200** | 1.9 | 1.77 | 1.5 | 1.69 | 1.32 | 1.14 | 1.22 | 1.03 | 1.43 | 1.3435 | 0.290304 |
| **500** | 1.34 | 1.35 | 1.68 | 1.08 | 1.28 | 1.6 | 1.17 | 1.44 | 1.15 | 1.3405 | 0.284854 |
| **1000** | 1.85 | 1.19 | 1.47 | 0.91 | 1.34 | 1.44 | 1.27 | 1.35 | 1.6 | 1.3075 | 0.240129 |
| **2000** | 1 | 1.69 | 1.42 | 1.28 | 1.28 | 0.79 | 1.35 | 1.44 | 0.99 | 1.2875 | 0.255855 |
| **7500** | 1.06 | 1.05 | 1.1 | 0.73 | 1.31 | 1.44 | 1.26 | 1.39 | 1.4 | 1.195 | 0.191874 |
| **10000** | 1.4 | 1.01 | 1.82 | 1.4 | 1.37 | 1.07 | 1.22 | 1.17 | 0.78 | 1.1965 | 0.278289 |
| **15000** | 1.1 | 1.42 | 1.24 | 1.47 | 2.02 | 1.38 | 1.19 | 0.95 | 1.15 | 1.3785 | 0.243965 |
| **20000** | 1.22 | 1.96 | 1.56 | 1.24 | 1.21 | 0.97 | 1.35 | 1.14 | 1.28 | 1.3325 | 0.244538 |
| **25000** | 1.48 | 1.48 | 1.95 | 0.96 | 1.12 | 0.99 | 1.23 | 1.25 | 1.35 | 1.3185 | 0.325063 |

# APPENDIX D

# Results for Monitoring Period Length and β-parameter

**Table D.1: 80% utilization, for MPL = 250 replication mean tardiness values (Ri: replication i, MPL: monitoring period length)**

| β | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 0.61 | 0.64 | 0.64 | 0.68 | 0.68 | 0.66 | 0.65 | 0.56 | 0.58 | 0.6 | 0.71 |
| **0.5** | 0.63 | 0.63 | 0.61 | 0.6 | 0.67 | 0.63 | 0.6 | 0.67 | 0.62 | 0.75 | 0.68 |
| **0.8** | 0.67 | 0.7 | 0.6 | 0.72 | 0.69 | 0.61 | 0.56 | 0.56 | 0.58 | 0.61 | 0.68 |
| **1** | 0.61 | 0.66 | 0.54 | 0.7 | 0.7 | 0.74 | 0.67 | 0.62 | 0.63 | 0.7 | 0.66 |
| **1.4** | 0.69 | 0.55 | 0.79 | 0.64 | 0.6 | 0.74 | 0.67 | 0.72 | 0.65 | 0.54 | 0.67 |
| **1.8** | 0.69 | 0.58 | 0.65 | 0.63 | 0.69 | 0.62 | 0.69 | 0.69 | 0.71 | 0.66 | 0.7 |
| **2.4** | 0.67 | 0.7 | 0.62 | 0.66 | 0.75 | 0.57 | 0.67 | 0.73 | 0.7 | 0.68 | 0.62 |
| **2.6** | 0.67 | 0.76 | 0.7 | 0.64 | 0.67 | 0.71 | 0.65 | 0.68 | 0.59 | 0.67 | 0.73 |

| β | R12 | R13 | R14 | R15 | R16 | R17 | R18 | R19 | R20 | Mean | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 0.59 | 0.7 | 0.69 | 0.66 | 0.63 | 0.66 | 0.63 | 0.7 | 0.58 | 0.6425 | 0.044589 |
| **0.5** | 0.58 | 0.67 | 0.67 | 0.76 | 0.59 | 0.7 | 0.54 | 0.75 | 0.62 | 0.6485 | 0.059496 |
| **0.8** | 0.63 | 0.7 | 0.67 | 0.62 | 0.66 | 0.72 | 0.66 | 0.71 | 0.67 | 0.651 | 0.051493 |
| **1** | 0.62 | 0.77 | 0.59 | 0.6 | 0.61 | 0.68 | 0.71 | 0.62 | 0.57 | 0.65 | 0.058938 |
| **1.4** | 0.57 | 0.57 | 0.64 | 0.67 | 0.67 | 0.64 | 0.71 | 0.73 | 0.62 | 0.654 | 0.066523 |
| **1.8** | 0.58 | 0.72 | 0.66 | 0.76 | 0.72 | 0.59 | 0.75 | 0.65 | 0.58 | 0.666 | 0.055763 |
| **2.4** | 0.67 | 0.59 | 0.76 | 0.6 | 0.79 | 0.69 | 0.63 | 0.71 | 0.71 | 0.676 | 0.058616 |
| **2.6** | 0.71 | 0.66 | 0.69 | 0.68 | 0.7 | 0.63 | 0.65 | 0.62 | 0.79 | 0.68 | 0.047016 |

**Table D.2: 80% utilization, for MPL = 500 replication mean tardiness values (Ri: replication i, MPL: monitoring period length)**

| β | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 0.74 | 0.7 | 0.57 | 0.7 | 0.75 | 0.76 | 0.71 | 0.57 | 0.63 | 0.6 | 0.74 |
| **0.5** | 0.74 | 0.7 | 0.68 | 0.7 | 0.69 | 0.61 | 0.74 | 0.7 | 0.74 | 0.67 | 0.69 |
| **0.8** | 0.6 | 0.68 | 0.71 | 0.64 | 0.73 | 0.74 | 0.65 | 0.62 | 0.68 | 0.61 | 0.69 |
| **1** | 0.73 | 0.63 | 0.61 | 0.7 | 0.66 | 0.64 | 0.69 | 0.59 | 0.6 | 0.64 | 0.67 |
| **1.4** | 0.71 | 0.74 | 0.72 | 0.7 | 0.61 | 0.71 | 0.69 | 0.69 | 0.66 | 0.7 | 0.69 |
| **1.8** | 0.75 | 0.63 | 0.62 | 0.64 | 0.66 | 0.66 | 0.68 | 0.65 | 0.71 | 0.64 | 0.74 |
| **2.4** | 0.61 | 0.73 | 0.66 | 0.66 | 0.73 | 0.73 | 0.6 | 0.74 | 0.7 | 0.59 | 0.65 |
| **2.6** | 0.65 | 0.63 | 0.81 | 0.69 | 0.69 | 0.57 | 0.61 | 0.61 | 0.75 | 0.67 | 0.66 |

| β | R12 | R13 | R14 | R15 | R16 | R17 | R18 | R19 | R20 | Mean | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 0.61 | 0.78 | 0.73 | 0.66 | 0.69 | 0.63 | 0.73 | 0.73 | 0.69 | 0.686 | 0.064105 |
| **0.5** | 0.6 | 0.69 | 0.59 | 0.64 | 0.62 | 0.67 | 0.63 | 0.7 | 0.72 | 0.676 | 0.046611 |
| **0.8** | 0.79 | 0.63 | 0.61 | 0.76 | 0.57 | 0.61 | 0.67 | 0.68 | 0.63 | 0.665 | 0.058714 |
| **1** | 0.55 | 0.71 | 0.65 | 0.68 | 0.66 | 0.66 | 0.7 | 0.7 | 0.62 | 0.6545 | 0.045593 |
| **1.4** | 0.66 | 0.69 | 0.58 | 0.68 | 0.69 | 0.71 | 0.64 | 0.74 | 0.6 | 0.6805 | 0.043827 |
| **1.8** | 0.61 | 0.65 | 0.65 | 0.66 | 0.72 | 0.71 | 0.68 | 0.66 | 0.7 | 0.671 | 0.039189 |
| **2.4** | 0.69 | 0.73 | 0.62 | 0.64 | 0.73 | 0.65 | 0.56 | 0.7 | 0.68 | 0.67 | 0.05458 |
| **2.6** | 0.83 | 0.65 | 0.63 | 0.72 | 0.66 | 0.75 | 0.71 | 0.66 | 0.74 | 0.6845 | 0.066923 |

**Table D.3: 90% utilization, for MPL = 500 replication mean tardiness values (Ri: replication i, MPL: monitoring period length)**

| β | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 0.73 | 1.45 | 1.03 | 0.93 | 1.53 | 0.89 | 1.08 | 1.19 | 0.96 | 0.81 | 1.2 |
| **0.5** | 0.96 | 1.35 | 1.38 | 1.12 | 1.23 | 1.01 | 1.1 | 0.89 | 0.77 | 1.28 | 1.02 |
| **0.8** | 1.36 | 1.07 | 1.34 | 1.41 | 1.53 | 0.9 | 1.11 | 1 | 1 | 1.14 | 1.36 |
| **1** | 1.09 | 1.65 | 0.89 | 1.03 | 1.24 | 1.18 | 1.12 | 1.34 | 0.88 | 1.4 | 1.49 |
| **1.4** | 0.8 | 1.39 | 1.44 | 1.02 | 1.04 | 1.17 | 1.04 | 1.18 | 1.12 | 1.63 | 1.61 |
| **1.8** | 1.08 | 1.03 | 1.03 | 1.13 | 0.66 | 1.21 | 1.42 | 0.91 | 1.61 | 1.03 | 1.08 |
| **2.4** | 1.2 | 1.4 | 1.41 | 1 | 1.02 | 1.3 | 0.92 | 1.22 | 1.55 | 1.28 | 1.32 |
| **2.6** | 1.07 | 1.29 | 1.09 | 0.78 | 1.06 | 1.4 | 1.3 | 1.19 | 1.34 | 1.21 | 1.24 |

| β | R12 | R13 | R14 | R15 | R16 | R17 | R18 | R19 | R20 | MEAN | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 1.84 | 1.12 | 0.97 | 1.2 | 1.38 | 0.88 | 1.19 | 1.05 | 0.74 | 1.1085 | 0.279554 |
| **0.5** | 1.05 | 0.92 | 1.21 | 1.36 | 1.22 | 1.06 | 1.36 | 0.85 | 1.29 | 1.1215 | 0.187119 |
| **0.8** | 0.62 | 0.98 | 1.08 | 1.32 | 0.8 | 1.27 | 1.31 | 1.05 | 0.88 | 1.1265 | 0.232815 |
| **1** | 1.1 | 1.44 | 1.31 | 1.16 | 1.38 | 1.43 | 1.07 | 1.21 | 1.09 | 1.225 | 0.202264 |
| **1.4** | 1.31 | 1.02 | 1.03 | 1.16 | 1.6 | 1.04 | 1.58 | 1.34 | 1.13 | 1.2325 | 0.241309 |
| **1.8** | 1.14 | 1.18 | 1.43 | 1.83 | 0.83 | 1.05 | 1.17 | 0.98 | 1.44 | 1.162 | 0.271828 |
| **2.4** | 1.07 | 0.9 | 1.06 | 0.89 | 1.23 | 1.22 | 1.69 | 0.83 | 1.1 | 1.1805 | 0.228484 |
| **2.6** | 1.24 | 1 | 1.33 | 1.2 | 1.34 | 1.23 | 1.12 | 1.05 | 1.04 | 1.176 | 0.149821 |

**Table D.4: 90% utilization, for MPL = 2500 replication mean tardiness values (Ri: replication i, MPL: monitoring period length)**

| β | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 0.99 | 0.82 | 1.69 | 1.46 | 1.18 | 1.38 | 1.13 | 1.48 | 1.43 | 1.48 | 1.4 |
| **0.5** | 0.85 | 1.34 | 1.09 | 0.91 | 1.21 | 0.97 | 1.8 | 1.18 | 1.49 | 1.34 | 1.48 |
| **0.8** | 1.15 | 1.58 | 1.16 | 1.56 | 1.05 | 0.91 | 1.5 | 1.42 | 0.96 | 0.96 | 1.36 |
| **1** | 0.933 | 0.84 | 0.83 | 1.32 | 1.09 | 1.49 | 0.91 | 1.4 | 1.19 | 1.17 | 1.16 |
| **1.4** | 1.16 | 1.16 | 1.1 | 1.04 | 1.8 | 1.17 | 1.45 | 1.32 | 1.43 | 1.44 | 1 |
| **1.8** | 1.44 | 0.99 | 1.32 | 1.04 | 1.14 | 1.4 | 1.5 | 1.15 | 1.31 | 1.08 | 1.29 |
| **2.4** | 1.44 | 0.87 | 1.21 | 0.86 | 1.38 | 1.5 | 1.39 | 1.2 | 1.28 | 1.28 | 1.4 |
| **2.6** | 1.77 | 1.16 | 1.45 | 1.27 | 1.2 | 1.18 | 1.25 | 1.28 | 1.42 | 1.04 | 1.61 |

| β | R12 | R13 | R14 | R15 | R16 | R17 | R18 | R19 | R20 | MEAN | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 1.41 | 1.42 | 1.06 | 1.35 | 1.13 | 1.23 | 0.95 | 1.29 | 1.06 | 1.267 | 0.220576 |
| **0.5** | 0.79 | 1.64 | 1.26 | 1.19 | 0.91 | 1.53 | 1.22 | 1.62 | 1.34 | 1.258 | 0.282965 |
| **0.8** | 1.32 | 1.13 | 1.05 | 1.44 | 0.89 | 0.9 | 1.6 | 0.9 | 1.66 | 1.225 | 0.271303 |
| **1** | 1.17 | 0.95 | 1.4 | 1.3 | 1.39 | 0.77 | 0.99 | 1.09 | 1.26 | 1.13265 | 0.215036 |
| **1.4** | 1.02 | 1.3 | 1.15 | 1.44 | 1.1 | 1.4 | 1.39 | 1.08 | 1.3 | 1.2625 | 0.20026 |
| **1.8** | 1 | 1.4 | 0.77 | 1.32 | 1.26 | 1.26 | 1.47 | 1.37 | 1.2 | 1.2355 | 0.188553 |
| **2.4** | 1.5 | 1.24 | 0.85 | 1.48 | 1.33 | 1.33 | 1.08 | 1.45 | 1.44 | 1.2755 | 0.2103 |
| **2.6** | 1.31 | 1.17 | 0.96 | 1.27 | 1.23 | 1.24 | 1.05 | 1.24 | 0.99 | 1.2545 | 0.196294 |

**Table D.5: 90% utilization, for MPL = 3750 replication mean tardiness values (Ri: replication i, MPL: monitoring period length)**

| β | R1 | R2 | R3 | R4 | R5 | R6 | R7 | R8 | R9 | R10 | R11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 1.13 | 1.02 | 1.13 | 1.29 | 1.46 | 1.17 | 1.06 | 1.14 | 1.29 | 1.06 | 1.3 |
| **0.5** | 1.27 | 1.21 | 1.42 | 0.86 | 1.17 | 1.04 | 1.28 | 0.67 | 1.23 | 1.47 | 0.94 |
| **0.8** | 1.26 | 1.27 | 1.52 | 1.06 | 1.02 | 1.27 | 1.24 | 1.22 | 1.23 | 1.57 | 1.25 |
| **1** | 1.41 | 1.09 | 1.42 | 0.91 | 1.18 | 1.26 | 1.04 | 1.14 | 1.55 | 1.07 | 0.78 |
| **1.4** | 1.56 | 1.11 | 1.02 | 1 | 1.5 | 1.2 | 1.54 | 1.11 | 1.25 | 1.32 | 1.11 |
| **1.8** | 1.44 | 0.79 | 0.79 | 1.28 | 0.94 | 0.97 | 1.39 | 1.53 | 1.51 | 1.48 | 1.44 |
| **2.4** | 0.94 | 1.46 | 1.05 | 1.35 | 1.48 | 1.17 | 1.1 | 1.48 | 1.66 | 1.35 | 1.29 |
| **2.6** | 1.22 | 1.06 | 1.57 | 1.23 | 1.39 | 1.2 | 1.02 | 1.49 | 1.41 | 0.94 | 1.22 |

| β | R12 | R13 | R14 | R15 | R16 | R17 | R18 | R19 | R20 | MEAN | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0.2** | 1.2 | 1.18 | 1.49 | 1.45 | 1.26 | 1.37 | 1.01 | 1.2 | 1.2 | 1.2205 | 0.142699 |
| **0.5** | 0.78 | 1.2 | 0.87 | 1.53 | 1.51 | 1.54 | 1.62 | 1.64 | 1.39 | 1.232 | 0.291613 |
| **0.8** | 1.39 | 1.41 | 1.59 | 1.35 | 1.2 | 1.41 | 0.94 | 1.22 | 1.45 | 1.2935 | 0.172391 |
| **1** | 1.31 | 1.45 | 0.98 | 1.36 | 1.34 | 0.89 | 0.76 | 1.25 | 1.25 | 1.172 | 0.227679 |
| **1.4** | 1.22 | 1.28 | 0.95 | 1.47 | 1.08 | 1.17 | 1.25 | 1.26 | 1.42 | 1.241 | 0.181685 |
| **1.8** | 1.65 | 1.51 | 1.48 | 0.75 | 1.65 | 1.31 | 1.28 | 0.87 | 1.29 | 1.2675 | 0.301241 |
| **2.4** | 1.44 | 1.38 | 0.92 | 1.09 | 1.53 | 1.36 | 1.49 | 0.78 | 1.37 | 1.2845 | 0.235271 |
| **2.6** | 1.27 | 1.24 | 1.52 | 1.45 | 1.17 | 1.48 | 1.35 | 1.25 | 1.23 | 1.2855 | 0.170586 |

# APPENDIX E

# Results for Scheduling with a Static Learning Tree

**Table E.1: 80% Utilization, MPL=250, β=0.2, DR set {MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.87 | 0.88 | 0.9 | 0.86 | 0.9 | 0.86 | 0.88 | 0.84 | 0.89 | 0.93 | 0.94 | 0.99 |
| LearnPerf | 0.81 | 0.78 | 0.82 | 0.89 | 0.81 | 0.81 | 0.8 | 0.76 | 0.75 | 0.83 | 0.83 | 0.84 |
| BestPerf | 0.63 | 0.66 | 0.65 | 0.66 | 0.62 | 0.64 | 0.65 | 0.6 | 0.67 | 0.71 | 0.63 | 0.66 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.94 | 0.84 | 0.99 | 0.82 | 0.95 | 0.82 | 0.89 | 0.9 | 0.8945 | 0.049891989 |
| LearnPerf | 0.83 | 0.81 | 0.81 | 0.75 | 0.81 | 0.73 | 0.79 | 0.73 | 0.7995 | 0.039930861 |
| BestPerf | 0.71 | 0.63 | 0.67 | 0.63 | 0.69 | 0.56 | 0.63 | 0.66 | 0.648 | 0.035183728 |

**Table E.2: 80% Utilization, MPL=500, β=1, DR set {MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.89 | 0.88 | 0.89 | 0.95 | 0.9 | 0.88 | 0.91 | 0.89 | 0.94 | 0.92 | 0.92 | 0.9 |
| LearnPerf | 0.86 | 0.86 | 0.89 | 0.9 | 0.88 | 0.88 | 0.83 | 0.83 | 0.91 | 0.9 | 0.87 | 0.89 |
| BestPerf | 0.64 | 0.66 | 0.66 | 0.67 | 0.67 | 0.66 | 0.68 | 0.66 | 0.67 | 0.686 | 0.676 | 0.656 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.97 | 0.93 | 0.89 | 0.89 | 0.94 | 0.81 | 0.79 | 0.83 | 0.896 | 0.045002924 |
| LearnPerf | 0.92 | 0.89 | 0.89 | 0.85 | 0.92 | 0.81 | 0.9 | 0.84 | 0.876 | 0.031355349 |
| BestPerf | 0.666 | 0.656 | 0.636 | 0.646 | 0.686 | 0.596 | 0.646 | 0.616 | 0.655 | 0.021980853 |

**Table E.3: 80% Utilization, MPL=1000, β= -, DR set {MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.89 | 0.87 | 0.89 | 0.93 | 0.91 | 0.89 | 0.92 | 0.89 | 0.92 | 0.92 | 0.9 | 0.89 |
| LearnPerf | 0.88 | 0.88 | 0.89 | 0.91 | 0.88 | 0.88 | 0.91 | 0.89 | 0.92 | 0.92 | 0.89 | 0.89 |
| BestPerf | 0.67 | 0.66 | 0.68 | 0.68 | 0.68 | 0.67 | 0.7 | 0.67 | 0.7 | 0.71 | 0.68 | 0.68 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.93 | 0.88 | 0.92 | 0.88 | 0.92 | 0.84 | 0.87 | 0.85 | 0.8955 | 0.02584875 |
| LearnPerf | 0.93 | 0.88 | 0.92 | 0.88 | 0.91 | 0.83 | 0.88 | 0.86 | 0.8915 | 0.023680994 |
| BestPerf | 0.7 | 0.67 | 0.7 | 0.68 | 0.7 | 0.63 | 0.67 | 0.66 | 0.6795 | 0.018771479 |

**Table E.4: 90% Utilization, MPL=500, β=0.2, DR set {MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 1.59 | 1.14 | 1.54 | 1.65 | 1.71 | 1.4 | 1.37 | 1.77 | 1.53 | 1.66 | 1.7 | 1.14 |
| LearnPerf | 1.35 | 1.37 | 1.28 | 1.34 | 1.47 | 1.29 | 1.35 | 1.63 | 1.63 | 1.53 | 1.38 | 1.17 |
| BestPerf | 1.08 | 1.11 | 1.01 | 1.05 | 1.18 | 1.01 | 1.1 | 1.36 | 1.23 | 1.19 | 1.07 | 0.9 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 1.4 | 1.52 | 1.31 | 1.34 | 1.54 | 1.72 | 1.34 | 1.52 | 1.4945 | 0.184860545 |
| LearnPerf | 1.32 | 1.39 | 1.14 | 1.39 | 1.56 | 1.42 | 1.22 | 1.43 | 1.383 | 0.134677002 |
| BestPerf | 1.06 | 1.12 | 0.8 | 1.13 | 1.29 | 1.13 | 1.06 | 1.12 | 1.1 | 0.123756977 |

**Table E.5: 90% Utilization, MPL=2500, β=1, DR set {MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 1.43 | 1.59 | 1.35 | 1.39 | 1.64 | 1.86 | 1.58 | 1.53 | 1.36 | 1.6 | 1.56 | 1.3 |
| LearnPerf | 1.49 | 1.44 | 1.37 | 1.43 | 1.64 | 1.58 | 1.42 | 1.51 | 1.45 | 1.56 | 1.56 | 1.48 |
| BestPerf | 1.15 | 1.1 | 1.02 | 1.08 | 1.2 | 1.19 | 1.11 | 1.17 | 1.12 | 1.17 | 1.17 | 1.1 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 1.4 | 1.39 | 1.88 | 1.4 | 1.57 | 1.55 | 1.66 | 1.61 | 1.5325 | 0.158575069 |
| LearnPerf | 1.45 | 1.47 | 1.48 | 1.53 | 1.3 | 1.45 | 1.51 | 1.63 | 1.4875 | 0.082454134 |
| BestPerf | 1.1 | 1.15 | 1.16 | 1.15 | 1.18 | 1.11 | 1.15 | 1.21 | 1.1395 | 0.046506932 |

**Table E.6: 90% Utilization, MPL=7500, β= -, DR set {MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 1.5 | 1.5 | 1.4 | 1.46 | 1.52 | 1.62 | 1.44 | 1.51 | 1.52 | 1.54 | 1.55 | 1.48 |
| LearnPerf | 1.51 | 1.48 | 1.42 | 1.44 | 1.52 | 1.62 | 1.42 | 1.49 | 1.54 | 1.53 | 1.52 | 1.47 |
| BestPerf | 1.18 | 1.18 | 1.11 | 1.18 | 1.22 | 1.25 | 1.14 | 1.2 | 1.2 | 1.21 | 1.21 | 1.17 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 1.53 | 1.55 | 1.56 | 1.57 | 1.57 | 1.45 | 1.48 | 1.54 | 1.5145 | 0.052362602 |
| LearnPerf | 1.51 | 1.57 | 1.52 | 1.58 | 1.57 | 1.47 | 1.51 | 1.52 | 1.5105 | 0.052060188 |
| BestPerf | 1.17 | 1.23 | 1.23 | 1.21 | 1.23 | 1.18 | 1.2 | 1.22 | 1.196 | 0.033308763 |

**Table E.7: 80% Utilization, MPL=250, β=0.2, DR set {MOD, MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.47 | 0.49 | 0.52 | 0.48 | 0.49 | 0.47 | 0.48 | 0.51 | 0.51 | 0.53 | 0.48 | 0.5 |
| LearnPerf | 0.41 | 0.41 | 0.43 | 0.44 | 0.41 | 0.39 | 0.4 | 0.42 | 0.43 | 0.44 | 0.41 | 0.42 |
| BestPerf | 0.36 | 0.35 | 0.38 | 0.38 | 0.32 | 0.35 | 0.35 | 0.28 | 0.35 | 0.41 | 0.37 | 0.39 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.51 | 0.48 | 0.52 | 0.49 | 0.51 | 0.46 | 0.47 | 0.48 | 0.4925 | 0.029464519 |
| LearnPerf | 0.43 | 0.43 | 0.43 | 0.41 | 0.43 | 0.38 | 0.39 | 0.4 | 0.4155 | 0.019967078 |
| BestPerf | 0.39 | 0.37 | 0.35 | 0.33 | 0.4 | 0.35 | 0.36 | 0.35 | 0.3595 | 0.01731291 |

**Table E.8: 80% Utilization, MPL=500, β=1, DR set {MOD, MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.43 | 0.42 | 0.45 | 0.45 | 0.42 | 0.44 | 0.46 | 0.44 | 0.45 | 0.44 | 0.45 | 0.42 |
| LearnPerf | 0.4 | 0.44 | 0.45 | 0.45 | 0.44 | 0.42 | 0.41 | 0.4 | 0.46 | 0.45 | 0.44 | 0.41 |
| BestPerf | 0.35 | 0.37 | 0.38 | 0.38 | 0.37 | 0.37 | 0.38 | 0.36 | 0.39 | 0.39 | 0.38 | 0.37 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.48 | 0.44 | 0.44 | 0.43 | 0.47 | 0.41 | 0.44 | 0.42 | 0.44 | 0.017770466 |
| LearnPerf | 0.45 | 0.44 | 0.44 | 0.43 | 0.43 | 0.38 | 0.43 | 0.4 | 0.4285 | 0.021830688 |
| BestPerf | 0.39 | 0.37 | 0.38 | 0.37 | 0.4 | 0.35 | 0.37 | 0.36 | 0.374 | 0.013138934 |

**Table E.9: 80% Utilization, MPL=1000, β= -, DR set {MOD, MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.43 | 0.42 | 0.44 | 0.47 | 0.44 | 0.44 | 0.45 | 0.43 | 0.46 | 0.45 | 0.44 | 0.43 |
| LearnPerf | 0.43 | 0.42 | 0.44 | 0.44 | 0.44 | 0.42 | 0.45 | 0.42 | 0.45 | 0.44 | 0.44 | 0.43 |
| BestPerf | 0.38 | 0.37 | 0.38 | 0.4 | 0.38 | 0.38 | 0.4 | 0.38 | 0.4 | 0.39 | 0.39 | 0.38 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.44 | 0.42 | 0.46 | 0.43 | 0.46 | 0.42 | 0.43 | 0.45 | 0.4405 | 0.014680815 |
| LearnPerf | 0.45 | 0.42 | 0.46 | 0.43 | 0.45 | 0.4 | 0.42 | 0.45 | 0.435 | 0.015043796 |
| BestPerf | 0.39 | 0.37 | 0.39 | 0.37 | 0.4 | 0.36 | 0.38 | 0.37 | 0.383 | 0.011742859 |

**Table E.10: 90% Utilization, MPL=500, β=0.2, DR set {MOD, MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.81 | 0.64 | 0.67 | 0.59 | 0.74 | 0.66 | 0.77 | 0.79 | 0.67 | 0.89 | 0.6 | 0.65 |
| LearnPerf | 0.67 | 0.53 | 0.54 | 0.48 | 0.61 | 0.54 | 0.65 | 0.65 | 0.54 | 0.76 | 0.49 | 0.55 |
| BestPerf | 0.56 | 0.6 | 0.67 | 0.48 | 0.51 | 0.67 | 0.52 | 0.62 | 0.58 | 0.54 | 0.63 | 0.44 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.61 | 0.64 | 0.54 | 0.69 | 0.7 | 0.67 | 0.67 | 0.68 | 0.684 | 0.082359929 |
| LearnPerf | 0.5 | 0.54 | 0.45 | 0.58 | 0.58 | 0.56 | 0.56 | 0.58 | 0.568 | 0.072663608 |
| BestPerf | 0.47 | 0.48 | 0.34 | 0.46 | 0.52 | 0.43 | 0.44 | 0.44 | 0.52 | 0.088317609 |

**Table E.11: 90% Utilization, MPL=2500, β=1, DR set {MOD, MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.6 | 0.62 | 0.45 | 0.57 | 0.74 | 0.78 | 0.68 | 0.63 | 0.66 | 0.7 | 0.66 | 0.5 |
| LearnPerf | 0.59 | 0.54 | 0.47 | 0.53 | 0.74 | 0.68 | 0.52 | 0.61 | 0.55 | 0.66 | 0.66 | 0.58 |
| BestPerf | 0.55 | 0.5 | 0.42 | 0.48 | 0.6 | 0.59 | 0.51 | 0.57 | 0.52 | 0.57 | 0.57 | 0.5 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.5 | 0.59 | 0.68 | 0.6 | 0.57 | 0.65 | 0.71 | 0.76 | 0.6325 | 0.087891979 |
| LearnPerf | 0.55 | 0.57 | 0.58 | 0.63 | 0.4 | 0.55 | 0.61 | 0.73 | 0.5875 | 0.082454134 |
| BestPerf | 0.5 | 0.55 | 0.56 | 0.55 | 0.58 | 0.51 | 0.55 | 0.61 | 0.5395 | 0.046506932 |

**Table E.12: 90% Utilization, MPL=7500, β= -, DR set {MOD, MDD, ODD, SPT}**

| Performances | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 | R# 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.58 | 0.58 | 0.58 | 0.56 | 0.6 | 0.63 | 0.57 | 0.59 | 0.6 | 0.6 | 0.59 | 0.58 |
| LearnPerf | 0.58 | 0.58 | 0.57 | 0.57 | 0.6 | 0.62 | 0.56 | 0.58 | 0.6 | 0.6 | 0.6 | 0.57 |
| BestPerf | 0.56 | 0.54 | 0.54 | 0.53 | 0.57 | 0.59 | 0.54 | 0.55 | 0.56 | 0.57 | 0.56 | 0.54 |

| Performances | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average | Std. Dev. |
|---|---|---|---|---|---|---|---|---|---|---|
| MultiPass | 0.59 | 0.63 | 0.62 | 0.61 | 0.61 | 0.58 | 0.59 | 0.61 | 0.595 | 0.019056702 |
| LearnPerf | 0.59 | 0.61 | 0.61 | 0.61 | 0.6 | 0.58 | 0.59 | 0.61 | 0.5915 | 0.016944181 |
| BestPerf | 0.56 | 0.58 | 0.58 | 0.58 | 0.57 | 0.55 | 0.55 | 0.57 | 0.5595 | 0.016693838 |

**Table E.13: 80% Utilization, MPL=250, β=0.2, DR set {MDD, ODD, SPT}**

| Dispatching Rule Percentages | Rules | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 15% | 12% | 14% | 11% | 14% | 11% | 14% | 13% | 14% | 14% | 14% |
| | MDD | 33% | 35% | 36% | 33% | 33% | 35% | 31% | 33% | 32% | 34% | 35% |
| | ODD | 51% | 52% | 49% | 54% | 52% | 52% | 53% | 52% | 53% | 51% | 50% |
| | | | | | | | | | | | | |
| Learning | SPT | 10% | 13% | 11% | 11% | 11% | 9% | 12% | 10% | 10% | 10% | 11% |
| | MDD | 52% | 54% | 55% | 54% | 52% | 57% | 53% | 50% | 55% | 58% | 55% |
| | ODD | 37% | 32% | 32% | 34% | 36% | 33% | 33% | 40% | 34% | 32% | 33% |
| | | | | | | | | | | | | |
| Best | SPT | 8% | 9% | 9% | 9% | 10% | 9% | 9% | 9% | 9% | 11% | 9% |
| | MDD | 52% | 55% | 55% | 57% | 51% | 53% | 53% | 55% | 54% | 55% | 55% |
| | ODD | 38% | 35% | 34% | 33% | 38% | 37% | 36% | 35% | 36% | 33% | 35% |

| Dispatching Rule Percentages | Rules | R# 12 | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 13% | 13% | 13% | 14% | 13% | 13% | 14% | 13% | 14% | 13% |
| | MDD | 34% | 33% | 33% | 33% | 34% | 35% | 34% | 34% | 32% | 34% |
| | ODD | 51% | 53% | 52% | 52% | 52% | 50% | 51% | 52% | 53% | 52% |
| | | | | | | | | | | | |
| Learning | SPT | 10% | 10% | 11% | 9% | 9% | 9% | 10% | 8% | 10% | 10% |
| | MDD | 52% | 54% | 52% | 57% | 57% | 56% | 50% | 55% | 57% | 54% |
| | ODD | 37% | 34% | 35% | 33% | 33% | 33% | 38% | 36% | 31% | 34% |
| | | | | | | | | | | | |
| Best | SPT | 10% | 10% | 9% | 9% | 9% | 10% | 8% | 9% | 9% | 9% |
| | MDD | 52% | 52% | 53% | 53% | 55% | 55% | 51% | 54% | 55% | 54% |
| | ODD | 36% | 37% | 37% | 37% | 35% | 34% | 40% | 36% | 35% | 36% |

**Table E.14: 80% Utilization, MPL=500, β=1, DR set {MDD, ODD, SPT}**

| Dispatching Rule Percentages | Rules | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 14% | 13% | 13% | 13% | 15% | 12% | 14% | 13% | 14% | 14% | 14% |
| | MDD | 35% | 34% | 36% | 33% | 33% | 36% | 31% | 34% | 33% | 34% | 35% |
| | ODD | 50% | 52% | 49% | 53% | 51% | 51% | 54% | 52% | 51% | 50% | 49% |
| | | | | | | | | | | | | |
| Learning | SPT | 14% | 14% | 18% | 14% | 15% | 13% | 15% | 13% | 15% | 13% | 16% |
| | MDD | 38% | 39% | 37% | 39% | 39% | 43% | 38% | 40% | 43% | 37% | 39% |
| | ODD | 47% | 46% | 44% | 46% | 44% | 42% | 45% | 45% | 40% | 48% | 44% |
| | | | | | | | | | | | | |
| Best | SPT | 13% | 13% | 14% | 13% | 14% | 13% | 14% | 14% | 14% | 15% | 14% |
| | MDD | 36% | 39% | 40% | 40% | 37% | 39% | 37% | 40% | 39% | 39% | 39% |
| | ODD | 49% | 46% | 45% | 45% | 47% | 46% | 48% | 45% | 46% | 45% | 45% |

| Dispatching Rule Percentages | Rules | R# 12 | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 13% | 13% | 14% | 14% | 14% | 14% | 13% | 13% | 15% | 14% |
| | MDD | 34% | 34% | 33% | 32% | 36% | 35% | 33% | 34% | 32% | 34% |
| | ODD | 52% | 52% | 51% | 53% | 49% | 50% | 53% | 51% | 52% | 51% |
| | | | | | | | | | | | |
| Learning | SPT | 15% | 15% | 13% | 14% | 14% | 12% | 14% | 14% | 13% | 14% |
| | MDD | 35% | 36% | 38% | 31% | 40% | 43% | 36% | 38% | 34% | 38% |
| | ODD | 48% | 48% | 48% | 53% | 45% | 43% | 49% | 46% | 52% | 46% |
| | | | | | | | | | | | |
| Best | SPT | 14% | 13% | 12% | 13% | 13% | 13% | 11% | 14% | 12% | 13% |
| | MDD | 39% | 37% | 39% | 37% | 39% | 40% | 37% | 38% | 39% | 39% |
| | ODD | 46% | 48% | 47% | 48% | 46% | 46% | 51% | 47% | 48% | 47% |

**Table E.15: 80% Utilization, MPL=1000, β= -, DR set {MDD, ODD, SPT}**

| Dispatching Rule Percentages | Rules | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 14% | 13% | 14% | 12% | 15% | 12% | 15% | 13% | 14% | 14% | 14% |
| | MDD | 32% | 34% | 35% | 34% | 32% | 35% | 31% | 36% | 34% | 35% | 34% |
| | ODD | 52% | 52% | 49% | 53% | 52% | 51% | 53% | 50% | 51% | 50% | 51% |
| | | | | | | | | | | | | |
| Learning | SPT | 13% | 11% | 14% | 15% | 16% | 14% | 13% | 16% | 12% | 12% | 14% |
| | MDD | 25% | 33% | 31% | 29% | 30% | 33% | 31% | 29% | 37% | 31% | 26% |
| | ODD | 61% | 54% | 53% | 54% | 52% | 52% | 54% | 53% | 49% | 56% | 59% |
| | | | | | | | | | | | | |
| Best | SPT | 13% | 13% | 14% | 13% | 14% | 14% | 14% | 14% | 15% | 14% | 13% |
| | MDD | 31% | 34% | 34% | 36% | 32% | 32% | 33% | 34% | 33% | 34% | 34% |
| | ODD | 55% | 52% | 51% | 50% | 52% | 53% | 51% | 51% | 51% | 50% | 51% |

| Dispatching Rule Percentages | Rules | R# 12 | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 13% | 13% | 13% | 14% | 14% | 14% | 14% | 14% | 13% | 14% |
| | MDD | 33% | 35% | 33% | 33% | 34% | 34% | 34% | 34% | 33% | 34% |
| | ODD | 52% | 51% | 52% | 51% | 51% | 51% | 51% | 51% | 52% | 51% |
| | | | | | | | | | | | |
| Learning | SPT | 16% | 16% | 13% | 16% | 14% | 15% | 14% | 12% | 14% | 14% |
| | MDD | 35% | 31% | 34% | 35% | 34% | 29% | 31% | 32% | 35% | 32% |
| | ODD | 48% | 51% | 52% | 47% | 51% | 54% | 54% | 55% | 50% | 53% |
| | | | | | | | | | | | |
| Best | SPT | 13% | 14% | 13% | 14% | 14% | 14% | 11% | 13% | 13% | 14% |
| | MDD | 34% | 33% | 33% | 33% | 34% | 34% | 33% | 34% | 33% | 33% |
| | ODD | 51% | 52% | 52% | 52% | 50% | 50% | 55% | 51% | 53% | 52% |

**Table E.16: 90% Utilization, MPL=500, β=0.2, DR set {MDD, ODD, SPT}**

| Dispatching Rule Percentages | Rules | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 3% | 3% | 3% | 2% | 2% | 3% | 1% | 2% | 2% | 2% | 3% |
| | MDD | 60% | 56% | 57% | 62% | 58% | 57% | 58% | 58% | 58% | 58% | 60% |
| | ODD | 35% | 40% | 39% | 35% | 39% | 39% | 39% | 39% | 38% | 38% | 36% |
| | | | | | | | | | | | | |
| Learning | SPT | 2% | 2% | 2% | 2% | 2% | 1% | 4% | 3% | 4% | 3% | 3% |
| | MDD | 82% | 86% | 83% | 86% | 85% | 82% | 85% | 86% | 82% | 82% | 81% |
| | ODD | 15% | 11% | 14% | 10% | 12% | 16% | 9% | 9% | 12% | 14% | 14% |
| | | | | | | | | | | | | |
| Best | SPT | 3% | 2% | 2% | 1% | 2% | 3% | 2% | 2% | 3% | 3% | 2% |
| | MDD | 82% | 83% | 79% | 83% | 83% | 82% | 84% | 82% | 82% | 80% | 83% |
| | ODD | 14% | 13% | 18% | 15% | 14% | 14% | 13% | 14% | 14% | 16% | 14% |

| Dispatching Rule Percentages | Rules | R# 12 | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 2% | 2% | 2% | 3% | 2% | 3% | 2% | 2% | 3% | 2% |
| | MDD | 56% | 59% | 57% | 58% | 56% | 56% | 57% | 58% | 59% | 58% |
| | ODD | 40% | 38% | 39% | 38% | 40% | 40% | 40% | 39% | 37% | 38% |
| | | | | | | | | | | | |
| Learning | SPT | 1% | 3% | 2% | 5% | 3% | 2% | 2% | 0% | 2% | 2% |
| | MDD | 82% | 86% | 84% | 82% | 84% | 84% | 83% | 86% | 87% | 84% |
| | ODD | 16% | 10% | 13% | 12% | 12% | 13% | 14% | 12% | 9% | 12% |
| | | | | | | | | | | | |
| Best | SPT | 2% | 3% | 2% | 3% | 2% | 3% | 4% | 1% | 2% | 2% |
| | MDD | 83% | 82% | 82% | 82% | 83% | 83% | 81% | 82% | 83% | 82% |
| | ODD | 14% | 13% | 14% | 14% | 14% | 13% | 13% | 15% | 14% | 14% |

**Table E.17: 90% Utilization, MPL=2500, β=1, DR set {MDD, ODD, SPT}**

| Dispatching Rule Percentages | Rules | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 3% | 2% | 3% | 2% | 2% | 3% | 2% | 2% | 2% | 3% | 3% |
| | MDD | 59% | 56% | 55% | 58% | 57% | 59% | 58% | 57% | 58% | 55% | 58% |
| | ODD | 36% | 41% | 41% | 38% | 39% | 37% | 39% | 39% | 38% | 41% | 37% |
| | | | | | | | | | | | | |
| Learning | SPT | 4% | 3% | 7% | 1% | 6% | 2% | 6% | 2% | 3% | 5% | 6% |
| | MDD | 63% | 65% | 61% | 65% | 59% | 63% | 66% | 64% | 64% | 60% | 63% |
| | ODD | 32% | 31% | 31% | 32% | 34% | 34% | 27% | 32% | 32% | 33% | 30% |
| | | | | | | | | | | | | |
| Best | SPT | 3% | 3% | 2% | 3% | 4% | 3% | 2% | 3% | 3% | 2% | 3% |
| | MDD | 63% | 62% | 60% | 61% | 66% | 63% | 64% | 63% | 62% | 63% | 61% |
| | ODD | 33% | 34% | 36% | 35% | 29% | 33% | 33% | 33% | 34% | 33% | 34% |

| Dispatching Rule Percentages | Rules | R# 12 | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 3% | 3% | 2% | 2% | 2% | 3% | 2% | 2% | 3% | 2% |
| | MDD | 57% | 59% | 58% | 63% | 54% | 54% | 58% | 60% | 59% | 58% |
| | ODD | 39% | 36% | 38% | 34% | 42% | 42% | 39% | 37% | 37% | 39% |
| | | | | | | | | | | | |
| Learning | SPT | 2% | 3% | 1% | 2% | 3% | 2% | 4% | 5% | 5% | 4% |
| | MDD | 63% | 67% | 67% | 66% | 62% | 66% | 63% | 62% | 58% | 63% |
| | ODD | 33% | 29% | 30% | 31% | 33% | 31% | 32% | 33% | 36% | 32% |
| | | | | | | | | | | | |
| Best | SPT | 2% | 3% | 3% | 3% | 2% | 3% | 2% | 4% | 4% | 3% |
| | MDD | 63% | 64% | 62% | 64% | 61% | 65% | 63% | 60% | 64% | 63% |
| | ODD | 33% | 31% | 34% | 31% | 35% | 31% | 33% | 34% | 31% | 33% |

**Table E.18: 90% Utilization, MPL=7500, β= -, DR set {MDD, ODD, SPT}**

| Dispatching Rule Percentages | Rules | R# 1 | R# 2 | R# 3 | R# 4 | R# 5 | R# 6 | R# 7 | R# 8 | R# 9 | R# 10 | R# 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 3% | 2% | 3% | 2% | 2% | 3% | 1% | 2% | 2% | 2% | 3% |
| | MDD | 59% | 55% | 56% | 60% | 57% | 58% | 56% | 57% | 59% | 56% | 58% |
| | ODD | 36% | 42% | 40% | 37% | 39% | 37% | 41% | 40% | 38% | 40% | 37% |
| | | | | | | | | | | | | |
| Learning | SPT | 3% | 3% | 1% | 2% | 2% | 3% | 3% | 3% | 2% | 2% | 3% |
| | MDD | 51% | 53% | 57% | 76% | 52% | 61% | 60% | 65% | 50% | 55% | 56% |
| | ODD | 44% | 43% | 40% | 21% | 44% | 35% | 35% | 30% | 47% | 42% | 40% |
| | | | | | | | | | | | | |
| Best | SPT | 3% | 3% | 1% | 3% | 2% | 3% | 2% | 2% | 1% | 2% | 3% |
| | MDD | 60% | 58% | 58% | 58% | 59% | 58% | 59% | 57% | 59% | 57% | 58% |
| | ODD | 36% | 38% | 39% | 38% | 38% | 38% | 38% | 39% | 38% | 40% | 38% |

| Dispatching Rule Percentages | Rules | R# 12 | R# 13 | R# 14 | R# 15 | R# 16 | R# 17 | R# 18 | R# 19 | R# 20 | Average |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Multi-pass | SPT | 2% | 2% | 2% | 3% | 2% | 3% | 1% | 1% | 2% | 2% |
| | MDD | 59% | 57% | 59% | 59% | 56% | 54% | 57% | 60% | 59% | 58% |
| | ODD | 37% | 39% | 38% | 37% | 41% | 41% | 40% | 37% | 38% | 39% |
| | | | | | | | | | | | |
| Learning | SPT | 1% | 1% | 4% | 2% | 2% | 4% | 2% | 4% | 2% | 2% |
| | MDD | 59% | 56% | 55% | 70% | 54% | 57% | 59% | 60% | 61% | 58% |
| | ODD | 38% | 42% | 40% | 26% | 43% | 38% | 38% | 34% | 35% | 38% |
| | | | | | | | | | | | |
| Best | SPT | 2% | 2% | 2% | 3% | 2% | 2% | 2% | 3% | 3% | 2% |
| | MDD | 59% | 57% | 58% | 59% | 58% | 57% | 57% | 56% | 58% | 58% |
| | ODD | 37% | 40% | 39% | 37% | 39% | 39% | 40% | 40% | 38% | 38% |

# APPENDIX F

# Results for Scheduling with Dynamic Learning Tree

**Table F.1: Plotted data in the charts**

| Data | X | X-Bar | X-Lower | X-Upper | Sigma | 2-Sigma | -Sigma | -2-Sigma | R | R-Bar | R-Lower | R-Upper |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 0.639 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 1.193 | 2.236 | 0 | 4.729 |
| 2 | 0.507 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 1.915 | 2.236 | 0 | 4.729 |
| 3 | 1.123 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 1.94 | 2.236 | 0 | 4.729 |
| 4 | 0.3 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 0.718 | 2.236 | 0 | 4.729 |
| 5 | 0.454 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 0.876 | 2.236 | 0 | 4.729 |
| 6 | 0.867 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 0.983 | 2.236 | 0 | 4.729 |
| 7 | 1.175 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 2.939 | 2.236 | 0 | 4.729 |
| 8 | 1.533 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 4.266 | 2.236 | 0 | 4.729 |
| 9 | 0.617 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 0.907 | 2.236 | 0 | 4.729 |
| 10 | 1.256 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 2.961 | 2.236 | 0 | 4.729 |
| 11 | 0.426 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 0.707 | 2.236 | 0 | 4.729 |
| **12** | **4.407** | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | **8.675** | 2.236 | 0 | 4.729 |
| 13 | 0.625 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 1.013 | 2.236 | 0 | 4.729 |
| 14 | 0.678 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 1.184 | 2.236 | 0 | 4.729 |
| 15 | 1.416 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 3.812 | 2.236 | 0 | 4.729 |
| 16 | 1.092 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 3.81 | 2.236 | 0 | 4.729 |
| 17 | 1.085 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 2.464 | 2.236 | 0 | 4.729 |
| 18 | 0.712 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 2.165 | 2.236 | 0 | 4.729 |
| 19 | 1.313 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 2.016 | 2.236 | 0 | 4.729 |
| 20 | 0.368 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 0.437 | 2.236 | 0 | 4.729 |
| 21 | 0.474 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 0.937 | 2.236 | 0 | 4.729 |
| 22 | 0.572 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 1.605 | 2.236 | 0 | 4.729 |
| 23 | 0.706 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 2.249 | 2.236 | 0 | 4.729 |
| 24 | 0.54 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 1.127 | 2.236 | 0 | 4.729 |
| 25 | 1.507 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 3.814 | 2.236 | 0 | 4.729 |
| 26 | 0.33 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 0.432 | 2.236 | 0 | 4.729 |
| 27 | 2.064 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 4.516 | 2.236 | 0 | 4.729 |
| **28** | **3.085** | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | **6.048** | 2.236 | 0 | 4.729 |
| 29 | 0.982 | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | 3.142 | 2.236 | 0 | 4.729 |
| **30** | **3.578** | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | **6.169** | 2.236 | 0 | 4.729 |

| **31** | **1.482** | 0.828 | -0.46 | 2.118 | 1.257 | 1.686 | 0.399 | -0.03 | **5.258** | 2.236 | 0 | 4.729 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 32 | 2.1 | 0.929 | -0.43 | 2.229 | 1.382 | 1.835 | 0.476 | 0.023 | 4.171 | 2.359 | 0 | 4.989 |
| 33 | 1.061 | 0.929 | -0.43 | 2.229 | 1.382 | 1.835 | 0.476 | 0.023 | 3.067 | 2.359 | 0 | 4.989 |
| **34** | 1.848 | 0.929 | -0.43 | 2.229 | 1.382 | 1.835 | 0.476 | 0.023 | **7.571** | 2.359 | 0 | 4.989 |
| 35 | 0.507 | 0.929 | -0.43 | 2.229 | 1.382 | 1.835 | 0.476 | 0.023 | 1.963 | 2.359 | 0 | 4.989 |
| **36** | **5.799** | 0.929 | -0.43 | 2.229 | 1.382 | 1.835 | 0.476 | 0.023 | **10.67** | 2.359 | 0 | 4.989 |
| 37 | 1.762 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 2.325 | 2.677 | 0 | 5.662 |
| 38 | 0.741 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 1.158 | 2.677 | 0 | 5.662 |
| 39 | 0.262 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 0.423 | 2.677 | 0 | 5.662 |
| 40 | 2.044 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 5.089 | 2.677 | 0 | 5.662 |
| 41 | 1.359 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 2.698 | 2.677 | 0 | 5.662 |
| 42 | 0.388 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 0.224 | 2.677 | 0 | 5.662 |
| **43** | **3.011** | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | **9.757** | 2.677 | 0 | 5.662 |
| 44 | 1.377 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 3.957 | 2.677 | 0 | 5.662 |
| 45 | 0.447 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 1.672 | 2.677 | 0 | 5.662 |
| 46 | 1.087 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 3.022 | 2.677 | 0 | 5.662 |
| 47 | 0.599 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 1.626 | 2.677 | 0 | 5.662 |
| 48 | 0.685 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 1.371 | 2.677 | 0 | 5.662 |
| 49 | 1.211 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 2.451 | 2.677 | 0 | 5.662 |
| 50 | 0.697 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 1.326 | 2.677 | 0 | 5.662 |
| 51 | 0.879 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 1.141 | 2.677 | 0 | 5.662 |
| 52 | 0.653 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 0.772 | 2.677 | 0 | 5.662 |
| 53 | 0.967 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 1.164 | 2.677 | 0 | 5.662 |
| 54 | 0.594 | 1.046 | -0.498 | 2.59 | 1.56 | 2.074 | 0.532 | 0.018 | 0.601 | 2.677 | 0 | 5.662 |

**Table F.2: Number of updates for the learning tree and the charts for DR set {MOD, MDD, ODD, SPT}**

| | Training data set: | Full | | Partial | |
|---|---|---|---|---|---|
| | (SPL, MPL, β): | (1000, 250, 0.2) | (7500, 500, 0.2) | (1000, 250, 0.2) | (7500, 500, 0.2) |
| Due date tightness: | | | | | |
| Adjusted | Learning tree updates | 313 | 315 | 255 | 239 |
| | Control chart updates | 157 | 163 | 131 | 117 |
| Not Adjusted | Learning tree updates | 503 | 504 | 252 | 254 |
| | Control chart updates | 242 | 252 | 125 | 127 |

**Table F.3: Number of updates for the learning tree and the charts for DR set {MDD, ODD, SPT}**

| | Training data set: | Full | | Partial | |
|---|---|---|---|---|---|
| | (SPL, MPL, β): | (1000, 250, 0.2) | (7500, 500, 0.2) | (1000, 250, 0.2) | (7500, 500, 0.2) |
| Due date tightness: | | | | | |
| Adjusted | Learning tree updates | 308 | 318 | 258 | 255 |
| | Control chart updates | 147 | 164 | 125 | 123 |
| Not Adjusted | Learning tree updates | 537 | 537 | 269 | 284 |
| | Control chart updates | 266 | 268 | 134 | 137 |