JOB SHOP SCHEDULING UNDER DYNAMIC AND STOCHASTIC MANUFACTURING ENVIRONMENT

A THESIS

SUBMITTED TO THE DEPARTMENT OF INDUSTRIAL ENGINEERING

AND THE INSTITUTE OF ENGINEERING AND SCIENCES

OF BILKENT UNIVERSITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF

MASTER OF SCIENCE

By

Erhan Kutanoğlu

January, 1995

TS
157.5
.K88
1995

8010720

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.

[illegible]

Assist. Prof. İhsan Sabuncuoğlu(Principal Advisor)

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.

[illegible]

Assist. Prof. Selim Aktürk

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.

[illegible]

Assist. Prof. Selçuk Karabatı

Approved for the Institute of Engineering and Sciences:

[illegible]

Prof. Mehmet Baray
Director of Institute of Engineering and Sciences

ABSTRACT

JOB SHOP SCHEDULING UNDER DYNAMIC AND STOCHASTIC MANUFACTURING ENVIRONMENT

Erhan Kutanoğlu
M.S. in Industrial Engineering
Supervisor: Assist. Prof. İhsan Sabuncuoğlu
January, 1995

In practice, manufacturing systems operate under dynamic and stochastic environment where unexpected events (or interruptions) occur continuously in the shop. Most of the scheduling literature deals with the schedule generation problem which is only one aspect of the scheduling decisions. The reactive scheduling and control aspect has scarcely been addressed. This study investigates the effects of the stochastic events on the system performance and develops alternative reactive scheduling methods.

In this thesis, we also study the single-pass and multi-pass scheduling heuristics in dynamic and stochastic job shop scheduling environment. We propose a simulation-based scheduling system for the multi-pass heuristics. Finally, we analyze the interactions among the operational strategies (i.e, look-ahead window, scheduling period, method used for scheduling), the system conditions, and the unexpected events such as machine breakdowns and processing time variations.

Key words: Job shop scheduling, reactive scheduling, simulation.

ÖZET

DİNAMİK VE RASTSAL ÜRETİM ORTAMINDA ATÖLYE ÇİZELGELEMESİ

Erhan Kutanoğlu
Endüstri Mühendisliği Bölümü Yüksek Lisans
Tez Yöneticisi: Yrd. Doç. Dr. İhsan Sabuncuoğlu
Ocak, 1995

Pratikte, üretim sistemleri dinamik ve rastsal olayların olageldiği ortamlarda çalışmaktadır. Çizelgeleme literatürünün büyük bir kısmı, çizelgeleme kararlarının yalnızca bir tarafını oluşturan çizelge yaratma problemi üzerinde yoğunlaşmaktadır. Tepkisel çizelgeleme ve kontrol tarafı pek incelenmemiştir. Bu çalışma, sistemde olagelen rastsal olayların sistem performansı üzerindeki etkilerini inceleyecek ve alternatif tepkisel çizelgeleme yöntemleri geliştirecektir.

Bu tezde, ayrıca tek geçişli ve çok geçişli çizelgeleme yöntemleri dinamik ve rastsal atölye tipi üretim ortamında çalışılacaktır. Ek olarak, çok geçişli bir benzetim temelli çizelgeleme sistemi önerilecektir. Son olarak, operasyonal stratejiler, sistem koşulları ve makine bozulmaları gibi beklenmeyen olaylar arasındaki ilişki ve etkileşimler ortaya çıkarılacaktır.

Anahtar sözcükler. Atölye tipi üretim sisteminde çizelgeleme, tepkisel çizelgeleme, benzetim.

To my Wife

ACKNOWLEDGEMENT

I am indebted to Assistant Professor Doctor İhsan Sabuncuoğlu for her supervision, suggestions, patience and understanding throughout this thesis study.

I would like to express my thanks to Assistant Professor Doctor Selim Aktürk and Assistant Professor Doctor Selçuk Karabatı for their comments.

I cannot fully express my gratitude, love and thanks to my wife Oya for their morale support and encouragement.

My thanks also go to my friends Muhittin H. Demir, Levent Kandiller, Nurettin Kırkavak, and Süleyman Karabük who supported me throughout my studies.

Contents

List of Figures

List of Tables

Along with these primary objectives, there is also one side goal to be achieved. This side goal is to test the performance of a recently developed scheduling heuristic known as "*Bottleneck Dynamics (BD)*" in a dynamic and stochastic job shop environment. Before going into the detailed discussion of the investigations, we offer some preliminaries and definitions for further reading.

1.2 Scheduling problem defined

The scheduling problem involves accomplishing a number of things that require various resources for periods of time. The resources are capacitated, i.e. they are in limited supply. The things to be accomplished are called "*jobs*" and are composed of elementary parts called "*operations*" or "*activities*". Each operation requires certain amounts of specified resources for a specified time called "*processing time*". Resources have also elementary parts called "*machines*". There can be other types of resources such as transporters and labor. Hence, the scheduling problems contain a set of jobs to be carried out and a set of resources available to perform those jobs. Given the jobs and resources, the scheduling problem is to determine the detailed timing of the jobs within the capability of the resources [6]. Information about the resources and the jobs determine the constraints of the scheduling problem. For the resources, we need to specify the capacity of each resource. In addition, we describe each job in terms of its resource requirement, its processing times, its due date, and any technological constraints. The technological restrictions define the precedence constraints of the operations. Hence, the job shop scheduling problem is a scheduling problem in which a number of jobs, each comprising one or more operations to be performed in a specified sequence on specified machines and requiring certain processing times, are to be processed. The objective usually is to find a processing order or a scheduling rule on each machine for which a particular measure of performance is optimized [51].

There are mainly two types of scheduling problem studied in the literature:

(1) Static scheduling problem, and (2) Dynamic scheduling problem.

Definition 1.1 (Static scheduling problem) *The scheduling problem in which all jobs are assumed to be available simultaneously is called static scheduling problem.*

Definition 1.2 (Dynamic scheduling problem) *The scheduling problem in which jobs are assumed to arrive on different times, i.e., in which the set of jobs to be scheduled changes over time, is called dynamic scheduling problem.*

The dynamic problem is more difficult than its static counterpart. The problem is even more complicated when the problem includes some other complexities such as multiple machines, different visitation sequences, and uncertainties about some characteristics of the system and the jobs, etc.

In general, classic job shop scheduling studies make the following assumptions [6]:

1. Jobs consist of strictly ordered operation sequences.
2. A given operation can be performed by only one type of machine.
3. There is only one machine of each type in the shop.
4. Processing times as well as due dates are known at the time of arrival.
5. Setup times are sequence independent and can be included in the processing times.
6. Once an operation is begun on a machine, it cannot be interrupted (i.e., *preemption* is not allowed).
7. An operation may not begin until its predecessors are complete.
8. Each machine can process only one operation at a time.
9. Each machine is continuously available for production.

In this study, we relax some of these assumptions to achieve better representations of the real manufacturing environments in our model and to examine the sensitivity of the results on these assumptions.

In the next section, we review the general solution approaches for the job shop scheduling problem.

1.3 Solution approaches

There are mainly two types of approaches in the job shop scheduling literature: (1) "*optimization techniques*", and (2) "*heuristics*". The former approach is mostly proposed for the static job shop scheduling and handles only limited-size problems. Among the optimization techniques, several mixed integer programming formulations and implicit enumeration algorithms can be listed. The largest job shop scheduling problems that can currently be solved optimally are 10 job, 10-machine static problems with make-span objective (Applegate and Cook [4]). This example clearly indicates that the static problems are very difficult to solve.

The difficulty of the scheduling problems of real-life systems are further compounded by the dynamic and stochastic nature of the environment (new job arrivals, machine breakdowns, etc). For this reason, heuristic approaches are recommended for real problems.

In general, the heuristics developed for the static job shop problems use optimum-seeking approaches (Raman, Talbot and Rachamadugu [50]), improvement techniques focusing on bottleneck machines (Adams, Balas and Zawack [1]) and decomposition methods (Byeon, Wu and Storer [11]).

Definition 1.3 (Off-line schedule) *The schedule that is generated for all available jobs all at once for the entire horizon is called off-line schedule.*

Definition 1.4 (On-line schedule) *The schedule in which scheduling decisions are made one at a time and when it is needed according to the changing system conditions is called on-line schedule.*

From these definitions, most of the static scheduling algorithms can be viewed as mechanisms to generate off-line schedules. In on-line scheduling approach, the schedule is not determined all at once, but is constructed over time as events occur. Off-line scheduling can be used as an approximate approach to the dynamic scheduling problems. In this case, the dynamic problem is divided into as a series of static problems. A schedule is generated at each occurrence of an event such as a new job arrival. At these points, a static problem is solved, and the solution is implemented in a rolling horizon basis. The points of generation of new off-line schedules are called "*rescheduling points*". Examples of this approach can be found in Raman and Talbot [49] and Church and Uzsoy [14]. This application of off-line scheduling in dynamic problems is also called "*interval scheduling approach*" [39] or "*scheduling/rescheduling approach*" [14].

Among on-line scheduling methods, "*priority dispatching*" constitutes an important class and it has a major application area for dynamic scheduling problems. In this case, scheduling decisions are made in response to events occuring in the system. The general form of priority dispatching can be characterized by two responses [29]:

1. Whenever an operation becomes available for processing at a machine, if the machine is free, then engage the machine with the operation; otherwise, place the operation in the operation queue of the machine.
2. Whenever a machine becomes free, choose an operation from its queue according to a "*priority dispatching rule*" and engage the machine with the chosen operation.

Given the details of any "*priority dispatching rule*", these two responses completely specify a schedule. Hence, in the generated schedule, no machine is held idle if it has at least one job in its queue. This type of schedule is also called "*non-delay dispatching schedule*". If the machines are allowed to have idle time in anticipation of the arrival of rush jobs, then the schedule is said to have "*inserted idleness*".

There are *job-based priorities* in which the priority does not change from one operation to another. *Operation-based priorities* depend on the current operation under consideration. Some of the priority rules are *dynamic* so that their values change with the time. There are also *static rules* whose values does not depend on the current time. If the jobs have dynamic priorities, then the order of the jobs in a queue might change over time even the job content of the queue does not change.

The priority dispatching approach has some advantages such as:

- The cost of a priority dispatching is very low in terms of both computational time and storage. Also, the information needed in calculating most of the priorities is easily available.
- Feasibility of the generated schedule is easily satisfied. There is no need to consider the precedence and machine constraints explicitly.
- Since the decisions are delayed until the last moment when they are needed, adding newly arrived jobs to the schedule is not a major problem.
- Reactions to unexpected events and interruptions is extremely inexpensive, because any event can trigger new schedule (or scheduling decision) with the most up-to-date information.
- Finally, it is easy to explain the main idea behind the rules to practitioners which makes their implementations easy.

However, the priority dispatching approach has two important disadvantages:

- Best choice for the dispatching rule is heavily dependent on the objective function. No single rule is the best for all of the objectives. The selection of the best rule also depends on the operating conditions.

- The priority dispatching has a myopic view in determining the relative merits of selecting an operation. The rules are used to determine a measure of urgency for each waiting operation and to select the most urgent operation. Most of them do not consider the inherent opportunity cost that all other jobs will not be selected. Hence, their decisions are suboptimal especially in the long-run.

Several investigators propose improvement procedures for the priority dispatching approach. For example, Anderson and Nyirenda [3] combine the different dispatching rules to obtain a composite priority measure. Some researchers use the computing power and the information systems capability available today. As a result, new priority dispatching rules are developed to utilize more global information about the system such as soon-to-arrive jobs, downstream machines, and opportunity costs of other jobs in the queue, etc. The Bottleneck Dynamics (BD) rule is such a rule that is resulted from these efforts (Morton and Pentico [39]).

Iterative improvement procedures (or multi-pass algorithms) are also the outcome of these efforts to eliminate the lack of global view of priority dispatching approach. "*Multi-pass rule selection algorithms*" evaluate the performance of each rule selected from a rule set, and select the best rule to implement in the next planning horizon. The performance evaluation is performed by using computer simulation. At any decision point (e.g., at the time of a new job arrival), a new rule is selected based on the new system conditions. This approach tries to catch the global view of the off-line scheduling. It combines the powers of different dispatching rules by selecting them according to the current shop condition in a dynamic manner and by implementing them in consecutive periods [63]. The other type of the iterative improvement procedures seeks to find the best values of some parameters used in a priority dispatching rule. Here, the algorithm evaluates the values of a parameter and selects the best one to implement. Again, discrete-event simulation is used as an evaluation tool [60].

In our study, we focus on the priority dispatching approach and analyze the performances of two iterative improvement procedures. First, we investigate the performance of BD and compare it with with other rules. Second, we use the multi-pass rule selection algorithm and the lead time iteration method that tries to find best waiting time estimations of the jobs for priority calculations.

Up to now, we discuss mostly the deterministic scheduling problems and the solution approaches to those problems. In the next section, we differentiate deterministic and stochastic environments and review off-line and on-line scheduling approaches in stochastic environments.

1.4 Stochastic environment defined

In a daily operation of a real manufacturing system, a number of unexpected events and interruptions occur such as;

- Machine breakdowns
- Rush orders
- Job waiting missing input
- Job rework and recycle
- Job scrapped and replacement
- Job due date changes
- Processing time variations

These events add the stochasticity to the scheduling problem.

Definition 1.5 (Deterministic environment) *The manufacturing environment where all information about the system is known with certainty at the time of scheduling, i.e., there is no unforeseen events or disturbances, is called deterministic environment.*

Definition 1.6 (Stochastic environment) *The manufacturing environment where some random events and interruptions occur in the system from time to time is called stochastic environment.*

The off-line scheduling approach can deal with these events under stochastic and dynamic environment in two different ways:

1. "*Rescheduling*": When an unexpected event occurs in the system, a static scheduling problem is solved from scratch under the new condition.

2. "*No reaction*": When an unexpected event occurs in the system, no reaction is undertaken. The previously generated off-line schedule is kept being used no matter what the system conditions are. This approach is sometimes called "*right shift approach*" since it inherently shifts the current schedule to the right. In this case, the reaction to the event is postponed until the next rescheduling point.

No reaction approach does not follow the current system conditions. For this reason, there could be a significant difference between the generated schedule on hand, and the current progress of the system. Its performance deteriorates quickly in the environments where the events with high impacts occur frequently.

According to the rescheduling approach, rescheduling is made whenever an unexpected event occurs regardless of the effect and importance of this event. This policy has a disadvantage that the system may be in a permanent state of rescheduling if many events occur in succession. Moreover, if the scheduling process takes long time to generate the schedule, the rescheduling may not be realized real-time. Furthermore, there might be events that does not necessarily require rescheduling. For these reasons, the rescheduling approach increases the nervousness of the system by revising the schedule frequently.

In the literature, there are also several other alternatives are proposed to combine the powers of these methods:

1. "*Event-driven rescheduling*" classifies the events those that require rescheduling, and those that do not need rescheduling all the jobs. For the first type of events, a new schedule for all jobs is generated as in the rescheduling approach. For the second set of events, no reaction is shown until the next rescheduling point. An example of an event-driven approach can be found in [14].

2. "*Partial rescheduling*" does not generate a schedule for all jobs from scratch. It tries to revise some part of the schedule when an unexpected event occurs in the system. Match-up scheduling is an example of such effort [9]. In addition, switching to on-line dispatching approach in case of an unexpected event may be listed in this class [37].

3. "*Performance-driven rescheduling*" compares the actual performance measure with the expected performance obtained from the generated schedule. If the difference between these two measures exceeds some specified limit, then the rescheduling method is implemented. If the difference does not exceeds the limit, no reaction is taken even when unexpected events occur in the system. One of these approaches can be found in [31].

As discussed earlier, reactive scheduling and control is relatively easy with priority dispatching rules, since the decisions are made one at time. The dynamic and state-dependent priority dispatching rules inherently develop its reactions to the unexpected events.

Several reactive scheduling policies can easily be developed by priority dispatching approach. Rerouting the affected jobs in case of machine breakdowns is an example of such policy. There could be other policies such as increasing the priorities of the affected jobs and preempting the loaded jobs, etc.

In this thesis, we outline a methodology for reactive scheduling and control by combining event-driven rescheduling and partial rescheduling approaches. We test the reactive scheduling policies consisting of rerouting mechanisms under random machine breakdowns to validate this methodology.

1.5 Scope of the thesis

In this study, we first analyze a new priority dispatching rule known as Bottleneck Dynamics (BD). In general, BD estimates prices for delaying each operation and prices for using each resource (or machine) in the system. Trading off these prices gives a kind of benefit/cost ratio that can be used to make several decisions such as scheduling, job releasing, and routing. This rule uses global information about the job and the system, such as the estimated waiting times of the job on downstream machines in its route, number of jobs in the queues of the downstream machines, the urgencies of the jobs in the current queue and other queues, etc. Also, it explicitly incorporates the usage costs of the machines in the priority. In chapter 2, we first review the literature on on priority dispatching approach with the special emphasis on Bottleneck Dynamics studies. Then we compare the performance of BD with those of other dispatching rules. Some of these rules are first investigated in our study. We also test the different versions of BD. Hence, this study will be the first and most comprehensive study to investigate the performance of BD under various conditions since it was first proposed by Morton and Pentico [39].

In Chapter 3 (the second part of the study), we investigate the effects of machine breakdowns on the system performance. In this chapter, we develop a general framework in which we combine the powers of the rescheduling, partial rescheduling, and no reaction approaches. Specifically, we define three reactive modes each of which corresponds to one of these approaches. These three modes are called according to the type, effect, and duration of the event. The experimental study is performed to test the effectiveness of the proposed approach under machine breakdowns. Here, we utilize the BD routing principles to develop reactive scheduling policies.

Finally, we investigate two iterative improvement for priority dispatching rules in Chapter 4. This chapter presents an extended literature review on the deterministic and stochastic studies with and without look-ahead. We propose an iterative simulation-based scheduling system that uses these improvement procedures. We study the multi-pass rule selection algorithm and lead time iteration method. The proposed system uses simulation as an evaluation tool. The effectiveness of improvement procedures are measured in the simulation experiments. Here, we define forecasting horizon, scheduling period, and look-ahead window that determine the timing of the invokes of the scheduling mechanism. We analyze the interactions between these concepts and the unexpected events such as machine breakdowns and processing time variations. We draw our overall conclusions and present further research directions in Chapter 5.

Chapter 2

Dynamic Job Shop Scheduling

2.1 Introduction

As discussed in Chapter 1, heuristics are usually recommended for the dynamic job shop scheduling problems. These heuristics can be classified into two categories: single-pass (one-pass) heuristics and multi-pass heuristics. In one-pass heuristics, a single complete solution is built up in one step at a time. Most of the priority dispatching rules can be considered in this category. These one-pass heuristics may also be used repeatedly to generate more sophisticated multi-pass or search heuristics with some additional computational costs. In multi-pass heuristics, an initial schedule is generated in the first pass, and then the consecutive passes are made to improve the performance measure. In this category, we can list neighborhood search, tabu search, simulated annealing, iterative dispatching, and iterative bottleneck algorithms. A simulation-based scheduling system proposed in Chapter 4 is also a multi-pass heuristic.

In this chapter, however, we focus on the single-pass heuristics, namely priority dispatch rules.

There are numerous studies which investigate the performances of priority dispatching rules under different experimental conditions by using discrete-event simulation. These studies indicate that no single rule is the best under all possible conditions. Their performances are affected by a number of factors such as shop load level, scheduling criteria, queue length, etc. For that reason, the relevant literature contains conflicting reports about the performances of these rules.

Secondly, the majority of the existing experimental studies are performed in a uniform (or balanced) shop environment where there is no dominant bottleneck in the shop (i.e. all machines in the shop are approximately equally utilized). Hence, there is a need to test the relative performances of the rules, when there is one or more bottleneck machines in the shop, to see whether the conclusions drawn from uniform job shop case are still valid.

Finally, there are some recently proposed rules (such as CEXSPT, MDSPRO and BD) whose performances are not generally known. Especially, ATC and BD have not been adequately tested in the dynamic job shop environments. The objective of this chapter is to study several single-pass versions of ATC and BD rules and compare them with other rules under various job shop environments (i.e. load levels, uniform vs bottleneck, tardiness levels, etc).

2.1.1 Scheduling Problem Defined

In this study, we consider the dynamic job shop (JS) problem with a primary performance measure as average weighted tardiness (WT). The results of number of tardy jobs (or percent of tardy jobs, PT) and average conditional weighted tardiness (CWT) are also reported in this chapter. The scheduling environment considered in this study is a dynamic reentrant job shop with the assumptions outlined in Chapter 1. Here, the following additional assumptions are also made:

- Each job has pre-specified routing with randomly sequenced machines and predetermined processing times on each machine in its route;
- Each job is released to the shop upon arrival;
- There is no transportation time between operations;

In addition to these assumptions, the job arrivals are dynamic and a job can visit any machine more than once (i.e., "*reentrant shop*") but not consecutively. The shop contains M machines and each job i has m_i operations with processing times p_{ij}, $j = 1, 2, \ldots, m_i$. There is a delay penalty, or tardiness weight, of w_i per unit time if job i is completed after its due date d_i. This weight includes customer badwill, cost of lost sales and charged orders and rush shipping costs. Then the WT objective is given by

$$WT = \frac{\sum_{i=1}^{n} w_i T_i}{n}$$

where n is the number of jobs completed during a specified horizon and the tardiness of job i is $T_i = max\ \{0, C_i - d_i\}$ in which C_i is the completion time of job i. If all the jobs have equal weights, then the objective function becomes unweighted (mean) tardiness. If the objective is to minimize PT, then we consider

$$PT = \frac{\sum_{i=1}^{n} \delta(T_i)}{n} \times 100$$

where $\delta(T_i) = 1.0$, if $T_i > 0$, and $\delta(T_i) = 0$ otherwise. The CWT objective can be expressed by

$$CWT = \frac{\sum_{i=1}^{n} w_i T_i}{NT}$$

where NT is the number of tardy jobs among n jobs (also it can be computed by $NT = PT \times n/100$).

In this study, we compare the tardiness performances of some conventional priority dispatching rules with those of recently developed rules in a simulated JS environment.These rules and their definitions are given in Table 2.1. Our emphasis will be on the new rules such as Apparent Tardiness Cost (ATC) and Bottleneck Dynamics (BD). The long run performances of different versions of ATC and BD are tested. Inserted idleness methods and different resource pricing schemes are studied for the first time in a dynamic JS environment. The next section reviews the relevant literature on the JS scheduling problem along with the discussion of the priority dispatching rules tested. Section 2.3 gives the system considerations and experimental conditions. Computational results are presented in Section 2.4. The chapter ends with concluding remarks in Section 2.5.

2.2 Literature Review

As discussed in the previous section, there are a number of scheduling rules in the literature some of which are listed in Table 2.1. Most of them are simple, known and used for many years. In this section, these rules are reviewed in detail.

2.2.1 Conventional Priority Dispatching Rules

The priority dispatching rules used in the dynamic job shop scheduling are classified by Ramasesh [51] according to the information content of the rules as follows:

- arrival times (e.g. FCFS, etc.)
- processing times (e.g. SPT, etc.)
- due date information
  - allowance-based (e.g. EDD, etc.)
  - slack-based (e.g. SLACK, etc.)
  - ratio-based (e.g. CR, S/RPT, S/OPN, A/OPN, etc.)

- cost or value added (e.g. Maximum weight, etc.)
- combination of one or more above (e.g. WSPT, MOD, ODD, OSLACK, etc.)

Among these rules, FCFS is generally used as a benchmark in the literature. A flow allowance of a job is the time between the release date and the due date, $A_i = d_i - r_i$. Under the allowance-based priority rules, the remaining allowance of a job i is calculated as $A_i(t) = d_i - t$ at time t. Since t is the same for all waiting jobs at a dispatching decision point, the simplest version of the allowance based priority is the earliest due date rule (EDD). The global slack time of a job is the remaining time for the waiting after remaining work is deducted from allowance. Hence, the global slack of job i waiting for operation j is $S_{ij}(t) = A_i(t) - P_{ij}$ where P_{ij} is the total remaining processing time or remaining work from current operation j to the end of the last operation. The simplest slack-based priority rule is the minimum slack time rule (SLACK), which gives priority to the smallest $S_{ij}(t)$.

The ratio-based priorities use some forms of a ratio for their implementation. For instance critical ratio rule (CR) assigns the highest priority to the job with the smallest $A_i(t)/P_{ij}$. While Rohleder and Scudder [52] and Scudder, et al. [54] show that the CR rule performs well for the net present value objective in the forbidden early shipment scheduling environment, Kim and Bobrowski [32] find out that CR is a good performer in job shop scheduling with sequence-dependent setup times. Another ratio-based rule is slack per remaining processing time (S/RPT) with the priority index $S_{ij}(t)/P_{ij}$. The priority index of the slack per remaining operation rule (S/OPN) is calculated as $S_{ij}(t)/m_{ij}$, where m_{ij} is the remaining number of operations from operation j to the last operation. The S/RPT rule sees the job with longer remaining processing time more urgent, while the S/OPN rule gives higher priority to the job with more number of operations remaining. Also we notice that S/RPT schedule is equal to the CR schedule in the sense that their priority indexes yield the same sequence, i.e. $S/RPT_{ij}(t) = CR_{ij}(t) - 1.0$.

Table 2.1: Priority dispatching rules. (The priorities are calculated for job i waiting for operation j at machine k at time t)

Priority Rule	Description
FCFS (First Come First Served)	$FCFS_{ij} = a_{ij}$
WSPT (Weighted Shortest Processing Time)	$WSPT_{ij} = \frac{w_i}{p_{ij}}$
WLWKR (Weighted Least Work Remaining)	$WLWKR_{ij} = \frac{w_i}{\sum_{q=j}^{m_i} p_{iq}}$
EDD (Earliest Due Date)	$EDD_i = d_i$
MDD (Modified Due Date)	$MDD_{ij}(t) = max\left\{ d_i, t + \sum_{q=j}^{m_i} p_{iq} \right\}$
SLACK (Least Slack)	$SLACK_{ij}(t) = d_i - t - \sum_{q=j}^{m_i} p_{iq}$
CR (Critical Ratio)	$CR_{ij}(t) = \frac{d_i - t}{\sum_{q=j}^{m_i} p_{iq}}$
S/RPT (Slack per Remaining Processing Time)	$S/RPT_{ij}(t) = \frac{d_i - t - \sum_{q=j}^{m_i} p_{iq}}{\sum_{q=j}^{m_i} p_{iq}}$
S/OPN (Slack per Remaining Operation)	$S/OPN_{ij}(t) = \frac{d_i - t - \sum_{q=j}^{m_i} p_{iq}}{m_i - j + 1}$
MDSPRO (Modified Dynamic Slack per Remaining Operation)	$M_{ij}(t) = \begin{cases} \frac{d_i - t - \sum_{q=j}^{m_i} p_{iq}}{m_i - j + 1} & \text{if } d_i - t - \sum_{q=j}^{m_i} p_{iq} > 0 \\ (m_i - j + 1)(d_i - t - \sum_{q=j}^{m_i} p_{iq}) & \text{otherwise.} \end{cases}$
ODD (Operation Due Date)	$ODD_{ij} = r_i + \frac{d_i - r_i}{\sum_{q=j}^{m_i} p_{iq}} \times \sum_{q=1}^{j} p_{iq}$

Priority dispatching rules (cont'd)

Priority Rule	Description
OSLACK (Operation Slack)	$OSLACK_{ij}(t) = r_i + \frac{d_i - r_i}{\sum_{q=j}^{m_i} p_{iq}} \times \sum_{q=1}^{j} p_{iq} - t - p_{ij}$
OCR (Operation Critical Ratio)	$OCR_{ij}(t) = \frac{r_i + \frac{d_i - r_i}{\sum_{q=j}^{m_i} p_{iq}} \times \sum_{q=1}^{j} p_{iq} - t}{p_{ij}}$
MOD (Modified Operation Due Date)	$MOD_{ij}(t) = max \left\{ r_i + \frac{d_i - r_i}{\sum_{q=j}^{m_i} p_{iq}} \times \sum_{q=1}^{j} p_{iq}, t + p_{ij} \right\}$
CEXSPT	(See Note 1)
COVERT (Cost Over Time)	$COVERT_{ij}(t) = \frac{w_i}{p_{ij}} \times \left(\frac{h \sum_{q=j}^{m_i} W_{iq} - \left(d_i - t - \sum_{q=j}^{m_i} p_{iq} \right)^+}{h \sum_{q=j}^{m_i} W_{iq}} \right)^+$
ATC (Apparent Tardiness Cost)	$ATC_{ij}(t) = \frac{w_i}{p_{ij}} \times exp \left(- \frac{\left(d_i - \sum_{q=j+1}^{m_i} (W_{iq} + p_{iq}) - p_{ij} - t \right)^+}{K p_{avg}} \right)$
BD (Bottleneck Dynamics)	$BD_{ij}(t) = \frac{w_i \times exp \left(- \frac{\left(d_i - \sum_{q=j+1}^{m_i} (W_{iq} + p_{iq}) - p_{ij} - t \right)^+}{K p_{avg}} \right)}{\sum_{q=j}^{m_i} R_{k(q)} p_{iq}}$

Note 1: CEXSPT rule partitions the original queue into 3 queues which are late queue, i.e. $S_{ij}(t) = d_i - t - \sum_{q=j}^{m_i} p_{iq} < 0$, operationally late queue (behind the schedule), i.e. $O_{ij}(t) = d_{ij} - t - p_{ij} = r_i + \frac{d_i - r_i}{\sum_{q=j}^{m_i} p_{iq}} \times \sum_{q=1}^{j} p_{iq} - t - p_{ij} < 0$, and ahead of schedule queue, i.e. $O_{ij}(t) \geq 0$. Then the rule selects SPT job from queue 1, if this job does not create a new late job with $Sij(t) < 0$. If it does, then a new SPT job is selected from queue 2, if it does not create a new operationally late job in queue 3. If it does, then a new SPT job is selected from queue 3.

However, the ratio-based priority rules have also some drawbacks in implementations, because negative ratios are difficult to interpret. When the remaining allowance or slack time is negative, these rules behave contrary to their intent. For example, the intent of the S/OPN rule is to give relatively higher priority to jobs which have more remaining operations because they will encounter more opportunities for queuing delay. But, when the slack is negative it tends to misbehave by giving priority to jobs with few remaining operations. For these reasons, Kanet [27] solves the "anomaly" in S/OPN rule by proposing modified dynamic slack per remaining operation rule (MDSPRO) as

$$MDSPRO_{ij}(t) = \begin{cases} \dfrac{S_{ij}(t)}{m_{ij}} & \text{if } S_{ij}(t) > 0 \\ \\ m_{ij}S_{ij}(t) & \text{otherwise.} \end{cases}$$

Then the MDSPRO rule selects the job with the smallest index to process.

All these rules except MDSPRO are extensively tested in scheduling studies. The results indicate that the rules are sensitive to the conditions of the system. For instance EDD, SLACK, S/RPT perform well in shops with light loads, but deteriorate in congested shops, whereas SPT performs well in congested shops with tight due dates, but fails for date-related criteria in shops with light loads and loose due dates [17], [61].

Another way to use the number of remaining operations is to utilize operation milestones called operation due dates (ODD). ODD breaks up a job's flow allowance into as many pieces as the number of operations in the job. Although there are several ways of assigning the ODDs, the work content method is found to yield best tardiness measures [5]. In this method, the initial flow allowance of a job is distributed to the operations proportional to the operation processing time. The operational version of allowance-based approach leads to the earliest operation due date (ODD). The same analogy for slack-based approach is the minimum operational slack rule (OSLACK). The operational ratio-based approach is smallest operation critical ratio (OCR). Kanet and

Hayya [28] compare the mean tardiness performances of the rules based on operational values with the job-based counter-parts, and show that operational rules are better than the job-based rules.

Baker and Bertrand [7] develop a dispatching rule known as modified due date (MDD) for the single machine shop. In this rule, a job's original due date serves as the modified due date until the job's slack becomes zero when its earliest finish time acts as the modified due date. This represents a combination of EDD and SPT rules. Baker and Kanet [8] extend the idea of the MDD rule to the multi-operation job shop. They use modified operation due date (MOD) which employs the ODDs to expedite the jobs in the system. The MOD of an operation is defined as its original ODD or its earliest finish time, whichever is larger. The rule then gives priority to the job with the smallest MOD. Experimental studies show that MOD outperforms other competing rules such as COVERT, CR, S/RPT, SLACK, and MDD at reducing unweighted tardiness at high utilizations and all but very loose levels of due date tightness. In loose due date case, the S/RPT rule produces very small values of tardiness. Baker [5] conducts some experiments to compare allowance based, slack-based, and ratio-based rules with the modified rules (MDD and MOD). The results show that slack based rules do not offer great advantage over simpler allowance-based rules, and operation-oriented rules perform better on the mean tardiness than job-based rules. The MOD rule is shown to be more robust to the changes in due date tightness and it is superior to the other rules when the due dates are not very loose in which S/OPN or A/OPN yields the minimum tardiness. Christy and Kanet [13] show that MOD is the preferred rule for the mean tardiness criterion in manufacturing systems with forbidden early shipment.

Carroll (1965) designs a dynamic priority rule (COVERT) for unweighted tardiness. The COVERT priority index represents the expected incremental tardiness cost per unit of imminent processing time, or Cost OVER Time. The expected tardiness cost is a relative measure of how much tardiness a job might experience if it is delayed by one time unit. The original COVERT can be converted into the weighted version (COVERT) since its derivation depends on the tardiness costs. Hence COVERT includes tardiness weight as a multiplier.

If job i queuing for operation j has zero or negative slack, then it is projected to be tardy by completion with an expected cost of w_i and priority index w_i/p_{ij}. If its slack exceeds some worst case estimate of the remaining waiting time over remaining operations, its expected cost is set to zero. If slack is between these extremes, then the priority linearly goes up while slack decreases. By this way, COVERT chooses the highest priority job. Carroll's experiments show that COVERT was superior to competing rules such as S/OPN and SPT in unweighted tardiness performance.

Russell et al. [53] examine the sensitivity of the COVERT rule to various operating conditions and performance measures, propose different versions of COVERT for the waiting time estimation, and test its performances (mean tardiness, mean flow time, etc.) with other scheduling rules such as EDD, SLACK, S/OPN, SPT, truncated SPT, MDD, MOD, and Apparent Urgency (AU, the very first version of ATC). The simulation experiments show the superiority of the COVERT in terms of mean tardiness and mean conditional tardiness performances.

Shultz [55] propose an expediting heuristic for the SPT rule (CEXSPT) which attempts to lessen the undesirable properties of SPT by controlling the scheduling of jobs with long processing times, and by employing job-based and operation-based due date information to expedite the late jobs (see Table 2.1). Shultz compare CEXSPT with COVERT, SPT, MOD, S/OPN and OCR. CEXSPT and COVERT produce lower unweighted tardiness values except very loose due date case where S/OPN yields the lowest. CEXSPT is the second best rule to SPT and COVERT at minimizing mean flow time and unweighted conditional tardiness respectively. In terms of proportion of tardy jobs, MOD and S/OPN yield the lowest measures at loose due dates, while SPT shows very good performance when the due dates are tight. Ye and Williams [67] explore the CEXSPT rule and make some improvements on it.

2.2.2 ATC Studies

Morton and Rachamadugu [40] develop a priority dispatching rule for the single machine WT problem, called "*Apparent Tardiness Cost (ATC)*" or R&M. The priority rule compares the slack of a job with a multiple times the average processing time of the waiting jobs, p_{avg}. The multiplier is *look ahead parameter*, K, which represents the average number of competing critical jobs. ATC gives maximum priority w_i/p_i (WSPT) if the job has negative slack, whereas it gives a portion of WSPT according to the slackness of the job if it has positive slack. Its priority increases with decreasing slack exponentially as in the following formula

$$ATC_i(t) = \frac{w_i}{p_i} \times exp\left(-\frac{(d_i - p_i - t)^+}{Kp_{avg}}\right)$$

where $(x)^+ = max\,\{0, x\}$.

The look ahead parameter, K, can be adjusted to reduce WT costs with changing utilization of the system and according to the due dates tightness level. Morton and Rachamadugu [40], Ow [46] and Vepsalainen and Morton [60] show that K fixed at 2.0 or 2.5 performs well in single machine and static flow shop scheduling problems.

Vepsalainen and Morton [59] extend ATC by adding tail lead time estimation concept to the rule for the multi-machine job shop problems. They calculate operation due date instead of original due date used in the single machine version of ATC as $d_{ij} = d_i - TL_{ij}$, where TL_{ij} represents estimated tail lead time which includes remaining work and remaining waiting times in the downstream operations. This differs from the operation due date of MOD. In MOD, initial flow allowance of a job is allocated to operation lead times in proportion to operation processing times. But in ATC, by considering finished and unfinished operations of the job, the time needed to perform the rest of the job is deducted from the due date.

Although there are several approaches to estimate the lead time, the investigators prefer to estimate it as a summation of known subsequent operation processing times and estimates of waiting times for these operations. Estimate of waiting time of job i for operation j at each remaining machine enroute is calculated as proportional to its processing time as

$$W_{ij} = bp_{ij}.$$

By this way, in multi-operation ATC, the global slack is allocated to the remaining lead time, which gives "*local resource-constrained slack*" as

$$SS_{ij}(t) = d_i - \sum_{q=j+1}^{m_i} (W_{iq} + p_{iq}) - p_{ij} - t.$$

The look ahead parameter in multi-operation ATC is selected as 3.0 in the experiments (For a full priority formula, see Table 2.1). In the priority formula, the exponential term is the activity time urgency (marginal cost of delay) if the job is currently scheduled and expected to be completed with slack SS_{ij}. By this way the urgency factor is

$$U_{ij}(t) = exp\left(-\frac{(SS_{ij}(t))^+}{Kp_{avg}}\right).$$

Results of experiments conducted by Vepsalainen and Morton [59] show that ATC is superior to COVERT and other competing rules such as EDD, S/RPT, WSPT for WT and also for the number of tardy jobs criterion.

A simulation study conducted by Vepsalainen and Morton [60] confirms the previous results. They test *priority-based estimation* and the *lead time iteration (LTI)* methods to estimate waiting times along with the "standard" estimation mentioned above. The former depends on queuing analysis. Priority-based waiting time is shorter for high-priority jobs and longer for low priority jobs in the queue. The latter is an iterative procedure (or multi-pass heuristic) to improve the WT objective: In each iteration, waiting times used in ATC (or COVERT) are obtained from previous iteration and they are used in the next iteration. By this way, by making better lead time estimation, the performance of the rule improves. The procedure is repeated until there is no improvement in the WT measures with last fixed number of iterations. The simulation results show that priority based waiting time estimation does not significantly improve

WT performance. Also it is found that ATC or COVERT with LTI improves the performance of the rules based on standard waiting time estimation up to 38%. But the use of LTI is limited to the cases when the system is static (all jobs are available at the beginning) or when the arrival times and other characteristics of the jobs are perfectly known in advance (The latter case is investigated in Chapter 4). In these cases, the model can be iterated before starting the implementation of the schedule.

Ow and Morton [47], [48] extend ATC to static early/tardy problem for single machine and flow shops. They use this extended version of ATC (EXP-ET) as an initial heuristic for the neighborhood search to find initial schedule and also as an evaluation function for filtered beam search. In these studies, EXP-ET produces very low early/tardy costs in comparison with the ATC and EDD. In addition, in moderate computation times, EXP-ET is improved by beam search as 15% maximum deviation from optimal value or lower bound found by preemptive relaxation of the problem.

Morton and Ramnath [41] add the inserted idleness concept to the ATC rule. By this way, some active schedules are examined myopically. In the first part of the study, they test this version of ATC (X-ATC) and standard ATC against other dispatching rules in a dynamic single machine shop in WT performance. While ATC outperforms all other rules, X-ATC improves the non-delay ATC performance further. In the second part of the study, they use these rules as a first phase heuristics of neighborhood search and tabu search to generate initial schedules. The results show that X-ATC provides best final schedule after the implementation of these search methods as it was the best performer in the first phase.

Kanet and Zhou [29] develop a decision theory approach (MEANP) consisting estimation of total costs for job shop scheduling problem and test it against COVERT and ATC in single machine case. While MEANP produces low values of unweighted tardiness and fraction of tardy jobs as compared with FCFS, SPT, COVERT, MOD and ATC; COVERT and MOD are found to be better than ATC. They recommend MOD for practical applications since it is the simplest of the three and it has no parameters to be estimated.

Bengu [10] analyzes the behavior of the ATC performance for varying values of the look-ahead parameter at different combinations of tardiness factor and due date range. The system considered is a flexible flow line shop. Both deterministic case and stochastic case with machine breakdowns are considered. The results show that K should be larger in both of the cases for small due date range and high tardiness factors. But in this study, while the flexible flow line includes more than one machine and more than one queuing point, the ATC implementation is like a single machine scheduling that produces permutation schedule. This version of ATC does not reflect the dynamic intent of the ATC rule.

2.2.3 Bottleneck Dynamics Studies

In standard applications of ATC, the only cost that the rule considers is about the processing time information of the current operation of the job. In other words, it is assumed that there is no cost for the other jobs in current queue and downstream machines' queues. Hence, there is no price for the downstream machines. Also ATC and other rules implicitly assume that the price of capacity exists only when the capacity is fully utilized leading to the conclusion that a machine that is not fully utilized (for example 80%) has zero price. Morton et al. [38] show that this is not true due to the non-stationarity of demand. Consequently, they develop the early version of bottleneck dynamics (BD) which they call SCHED-STAR. The proposed model is a scheduling system which uses cost-benefit analysis to make scheduling decisions based on net present value (NPV) of revenues and costs. NPV objective includes tardiness costs, direct costs, inventory holding costs and revenues from sales. SCHED-STAR calculates each resource prices by busy period analysis [42] and then it gives priorities to the jobs according to the calculated rate of return. The model starts with the initial estimates of prices and lead times. Jobs are released into the system from a pool when its computed rate of return is higher than a pre-specified threshold value. After each iteration the NPV is evaluated, lead times and prices are reestimated by using waiting times and busy periods in previous iteration and simulation proceeds to the next iteration. By this way the resource price iteration (RPI) and LTI are combined in the same module. They test the model in different types of shops including job shop and in different conditions. SCHED-STAR is the best performer as compared with the different versions of COVERT, CR, and EXP-ET. However, in this study the problem sizes studied are very small. Each problem consists of at most 50 operations all available at the beginning of the scheduling period.

In another study, Lawrence and Morton [35] test resource pricing heuristics with and without LTI in static resource-constrained multi-project scheduling problem with tardy costs. They develop a resource price-based priority rule similar to that of SCHED-STAR by taking into account several resource pricing schemes. They include five types of resource pricing methods. *Uniform pricing* assumes that all resources are of equal importance giving them 1.0 as a resource price. *Resource load pricing* estimates prices as proportional to total resource load. *Bottleneck resource pricing* uses OPT-like idea and identifies the bottleneck resource with largest load, and gives it a scaled price 1.0, while other resources are assigned prices of zero. *Busy period resource pricing* is mainly based on busy period analysis which is developed by Morton and Singh [42]. *Empirical resource pricing* measures prices by successively relaxing resource constraints and observing the effects on total tardiness. The change in objective function gives a relative measures of resource prices. They divide the use of information in activity costing as myopic and global. *Myopic costing* considers only prices of current resources which process the current activity and the cost is static. *Global costing* takes into account all prices of resources which process downstream activities. Global costs are dynamic, and are estimated as the sum of resource costs for all unfinished intra-project activities at current time. The first part of the study shows that global pricing dominates its myopic counterparts for all resource pricing rules in WT performance. Five global pricing rules with and without LTI and 20 benchmark rules are run in different experimental conditions. They find out that all pricing heuristics dominate all the benchmark rules, and performances of different pricing heuristics are statistically indistinguishable. In addition, it is shown in this study that the LTI reduces the WT of all pricing rules by 18% on the average.

Recently, Morton and Pentico [39] bring together all the pieces of their previous studies and develop a new rule with resource pricing scheme which is called Bottleneck Dynamics (BD). In general, BD is a method which estimates prices (or delay costs) for delaying each possible activity (or operation) and prices for using each resource in a shop. Trading off these heuristic prices (or costs) gives a kind of benefit/cost ratio that can be used to make scheduling decisions.

At a scheduling decision point, for each operation of the jobs in a certain queue, an activity price (AP) is calculated. In BD, the AP of an operation represents the costs saved per unit time by expediting the operation by a unit time. Also there are resource prices for each resources (mainly machines). Resource price gives an estimated extra costs if the resource is breakdown or is used for one time unit. If the resource price of a machine k at time t is known $(R_k(t))$ and if the operation j of job i is processed on machine k, then the resource usage of an activity for processing is $R_k(t)p_{ij}$. The BD principle in such a case is that if a net saving is desired at the end of an expediting decision, then it is necessary to expedite the job on its downstream operations. By this way, its completion time will be smaller. In general, job i has operations $j, \ldots, m_i$ as yet unstarted and that will be processed on machine $k(q), q = j, \ldots, m_i$. If the current operation j is expedited by one unit time, its total resource usage is over the remaining downstream operations. Hence, the total usage of a job while expediting is given by

$$\sum_{q=j}^{m_i} R_{k(q)}(t)p_{iq}.$$

Therefore, the BD priority is calculated by trading off the activity price gained by expediting a job and total resource usage of the job. Then, if we denote the activity price of operation j of job i at time t by $AP_{ij}(t)$, the BD priority is calculated as

$$BD_{ij}(t) = \frac{AP_{ij}(t)}{\sum\limits_{q=j}^{m_i} R_{k(q)}(t)p_{iq}}.$$

By this way, BD prioritizes the jobs with larger activity prices, while penalizing the jobs with longer processing times on bottleneck machines on their route. Next, we discuss the pricing mechanisms.

Activity Pricing

Since AP represents the costs saved per unit time by expediting an operation of job by one unit of time, decrease in customer dissatisfaction can be estimated. Then AP is directly related to the estimated lateness or tardiness of the job and its weight. Hence, the estimated activity price of operation j of job i that will be processed on machine k at time t is given by

$$AP_{ij}(t) = w_i U_{ij}(t),$$

where $U_{ij}(t)$ is a function of estimated lateness and is calculated as in ATC. (In fact, $U_{ij}(t)$ is named as urgency factor, and it leads to that $AP_{ij}(t)$ takes some portion of weight w_i, by ranging between 0.0 and 1.0).

If the expected completion time of job i at time t is shown by $C_{ij}(t)$, then the estimated lateness $EL_{ij}(t)$ is calculated as

$$EL_{ij}(t) = C_{ij}(t) - d_i.$$

$C_{ij}(t)$ can be estimated by the sum of tail lead time, current operation processing time and the current time. Then

$$C_{ij}(t) = t + p_{ij} + TL_{ij}.$$

From here, we see that $EL_{ij}(t) = -SS_{ij}(t)$ as in ATC. By this way, $EL_{ij}(t)$ can be used in urgency factor calculation as $SS_{ij}(t)$ is used in urgency as in ATC. Then,

$$U_{ij}(t) = exp\left(-\frac{(-EL_{ij}(t))^+}{Kp_{avg}}\right).$$

If the estimated lateness decreases, the urgency gets larger. If the estimated lateness is already 0 or negative then urgency factor takes its maximum value as 1.0 (leading to activity price = weight of the job).

Resource Pricing

As defined in the previous sections, the resource price gives an estimated extra costs if the resource is breakdown or is used for a purpose for one time unit. Therefore, if the resource is shut down for some time, all the jobs in the queue and the jobs that will arrive before the next idle status of the machine are delayed. The time between two consecutive idle status of the machine is called "*busy period*" of the machine. Hence, if the resource has some jobs in its queue and new arrivals occur before they are finished, the current busy period is the time up to the point that the resource will become first idle. Then the extra cost to the system is the sum of all activity prices of the jobs in queue plus the activity prices of the jobs that arrive before busy period ends. If there are $N_k(t)$ jobs in the current busy period of resource k, the fundamental resource price for resource k can be written as

$$R_k(t) = \sum_{i=1}^{N_k(t)} AP_{ij}(t).$$

However, in a real shop, we cannot know exact $N_k(t)$ and activity prices of the jobs. Also, the price will vary over time and roughly proportional to the current busyness of the resource. Since the resource price is a function of current busy period and it is very difficult to exactly know the busy period, there is a need to estimate the busy period.

Morton and Pentico [39] suggest some alternative ways of implementing busy period analysis in the estimation of resource prices. One of the *static* and moderately simple methods estimates the long run average busy period by queuing theory approximation. This is as follows:

$$R_k(t) = (wU)_{avg} \frac{\rho_k}{(1.0 - \rho_k)^2}.$$

In this formula $(wU)_{avg}$ is the average delay price of activities at the resource and ρ_k is the long term utilization rate of resource k. The last part of the formula expresses the unconditional average length of the busy period for an M/M/1 queue as a function of the utilization rate.

A *dynamic* counterpart of the previous one is based on the actual jobs in the queue, rather than average queue as shown below:

$$R_k(t) = \sum_{i=1}^{L_k(t)} w_i U_{ij}(t) + (wU)_{avg} L_k(t) \frac{\rho_k}{1.0 - \rho_k}$$

where $L_k(t)$ is the current queue length.

The dynamic and iterative pricing method can also be used in estimating the resource price as in SCHED-STAR. This method uses iterative approach as in the lead time iteration. It consists of running the full scheduling problem by some heuristic as a first step. After analyzing the actual busy periods, weights and urgencies for each resource at each time an activity begins processing, it uses these to calculate each resource price. In this method, there is a termination rule which uses a convergence criterion. The procedure is repeated with newly estimated prices until the convergence criterion is satisfied (e.g., if there is no significant amount of improvement in the performance measure in the last x iterations, the algorithm will stop). However, usage of this approach is very limited. In addition to the drawbacks of LTI, the usage of RPI needs high computation requirement. For example, in SCHED-STAR study, the largest system considered consists of 10 jobs each with 5 operations and 5-machine static job shop [38]. By this way, the analysis of each busy period of each machine after each iteration can be more tractable.

Also, there are some versions of resource pricing that do not depend on busy period analysis. Uniform pricing, resource load pricing and bottleneck pricing are some of them. While uniform pricing assumes that all resources have equal importance and equal resource price as 1.0, in resource load pricing and bottleneck pricing, the utilizations of the machines determine the scaled resource prices.

Inserted Idleness

Normally, BD does not allow the inserted idleness for late-arriving hot jobs. As Morton and Ramnath [41] specify, for any regular objective, no operation may be considered to be scheduled next on a given machine k, unless it is currently available in the queue, or else will arrive before the current time (t) plus the processing time of the shortest job in the queue (p_{mink}). In this case, we consider these jobs to dispatch, but we reduce the priorities of such soon-to-arrive jobs in proportion to the inserted idleness. The proportionality β goes up linearly with the utilization of the machine k. By this way, we extend the set of candidate jobs from the set of jobs currently available in the queue by adding the jobs that will arrive in the near future. If we denote the arrival time of job i to resource k for operation j as a_{ij}, then the priorities of such jobs are revised according to the following formula

$$BD_{ij}(t)' = BD_{ij}(t) \times \left(1.0 - \beta\frac{(a_{ij} - t)^{+}}{p_{mink}}\right).$$

This means that the original priority of a not-yet-arrived job is degraded by a term proportional to the idleness involved as a fraction of the minimum waiting job. Experiments conducted by Morton and Ramnath [41] show that $\beta = 1.3 + \rho_k$ is appropriate for single machine shops. If this degraded priority of a soon-to-arrive job is still greater than the priorities of the jobs in the queue then the machine is kept idle until this job arrives to the machine.

2.3 Experimental Study

In simulation experiments, similar experimental design used by Vepsalainen and Morton [59], [60] is conducted. We simulate a reentrant CJS with continuously available 10 machines. Jobs arrive continuously according to the Poisson process. The jobs have fixed number of operations selected from a discrete uniform distribution from 1 to 10. The operations are randomly processed through machines. Two types of shop is simulated: The first one is *uniform job shop* where almost every machine is equally utilized with operation processing times drawn from a uniform distribution U[1,30]. The other shop is *bottleneck job shop* where one or more machines are bottleneck with long processing times. The relative speeds of three machines are up to 30% faster than average, and three up to 30% slower. By this way some machines are highly utilized, some are low utilized. Job weights are drawn from U[1,30]. Due dates are assigned randomly over a full range of flow allowances, with an average of 6.0, 4.5, 3.0 times the mean total job processing time for relatively loose, medium, and tight due date setting, respectively. By this way, three different levels of due date tightness are set in the experiments.

The average utilization of the shop is determined by calibrating the arrival rate of the jobs. In uniform job shop case, arrival rate is adjusted to achieve approximately 60% utilization on the average in low level, 80% utilization in medium level, 90% in high level, and 95% in very high level. In bottleneck shop, the utilization of the bottleneck determines the system load level. In this case, arrival rate is adjusted to achieve 60% utilization on bottleneck machine in low level, 80% utilization in medium level and 95% in high level.

The system is simulated by using SIMAN simulation language with some additional C subroutines linked in UNIX environment. We use method of batch means to compare the results. We determine a conservative warm-up period for the system as 2,500 job-completion. We make 10 batches with 1,000 jobs each resulting to 12,500 jobs per run. In a way, this is the first controlled experiment in that the long term performance of ATC and BD are investigated. The priority dispatching rules given in Table 2.1 are used in the experiments.

Initially, some pilot experiments are conducted to select appropriate parameter values for COVERT, ATC and BD. As a result of these simulation runs, COVERT is used with parameters of b = 2.0 and h = 2.0. In original ATC, waiting time estimation parameter (b) is set to 2.0. The look-ahead parameter (K) is adjusted according to the job shop type and utilization level. In the uniform shop K is fixed at 2.0 in the low utilization level, K = 3.0 in the medium utilization level, K = 4.0 at high utilization, and it is 5.0 in very high utilization level. In the bottleneck shop, K is 1.5 in the low and medium levels of utilization, and it is fixed at 3.0 in the high utilization level. For ATC with inserted idleness (X-ATC), the inserted idleness parameter β is selected as $2.0 + \rho_k$. For the inserted idleness, the jobs that are currently processed on the machines and whose next operations are the machine under consideration are considered as soon-to-arrive jobs. The same parameter values in ATC are used for BD and X-BD.

Five different resource pricing schemes are also utilized for BD and X-BD rules. These are:

- *Dynamic resource pricing* based on queuing approximation with the original waiting time estimation as $W_{ij} = bp_{ij}$ (BD1 and X-BD1),
- Static resource pricing based on queuing approximation with the original waiting time estimation as $W_{ij} = bp_{ij}$ (BD2 and X-BD2),
- *Static resource pricing* with $R_k = 1.0$ for all k (*uniform resource pricing*, BD3 and X-BD3), (If it is assumed that all jobs have equal urgencies such as 1.0, and this pricing method is used then the rule transforms into weighted least work remaining (WLWKR));
- *Dynamic resource pricing* based on queuing approximation with the waiting time estimation developed by Kanet and Zhou [29] (BD4 and X-BD4). The waiting time for job i waiting for operation $j(i)$ at machine k with queue length $L_k(t)$ is estimated by

$$W_{ij(i)} = \frac{\sum\limits_{l=1,l\neq i}^{L_k(t)} p_{lj(l)}}{2}.$$

- *Bottleneck resource pricing* with $R_g = 1.0$, where $\rho_g = max_k\{\rho_k\}$, and $R_k = 0.0, k = 1,\ldots,m, k \neq g$ (BD5, X-BD5).

These are all global resource pricing methods as in the definition of Lawrence and Morton [35]. The myopic counter-part of these rules is naturally the ATC rule.

2.4 Computational Results

For the purpose of easy presentation of the results, we divide the scheduling rules into 5 groups:

- FCFS, WSPT, WLWKR rules that do not contain any due date information;
- The job-based rules that use due date information such as allowance, slack, or ratio (EDD, MDD, SLACK, CR, S/OPN, MDSPRO);
- The operation-based rules that utilize due date information (ODD, OSLACK, OCR, MOD, CEXSPT) and COVERT;
- Various versions of the BD rule with different resource pricing schemes and ATC;
- The inserted idleness versions of the ATC and BD rules;

First, the performances of the ATC and BD rules are compared for uniform and bottleneck job shop cases in the next section. The results of other rules are discussed in Section 2.4.2. The cross comparison of selected versions of the ATC and BD rules and other dispatch rules are given in Section 2.4.3.

2.4.1 The Comparison of the ATC and BD Rules

Average Weighted Tardiness Performances

The average weighted tardiness performances of the various versions of ATC and BD in the uniform and bottleneck job shop cases are presented in Tables 2.2 and 2.3.

First, the uniform job shop case is considered. When the performances of noninserted versions of ATC and BD are compared, almost no difference is seen between the rules in low utilizations regardless of due date tightness levels. However, as the utilization of the system increases, the ATC rule performs better than the BD rules. This difference becomes even more significant when the load of the system is very high. Same observations are made when the inserted idleness case is considered. This finding is quite surprising because the BD rule uses more global information such as information on downstream operations and machines. But, the results indicate that the resource pricing and global information are not always so helpful contrary to our expectations.

It is also observed that inserted idleness improves the performances of the rules significantly except high and very high utilization cases. The improvement achieved by implementing inserted idleness over the non-inserted versions changes from 1.4% and 16.8% in low and medium utilizations. In general, the improvement on the BD rules is greater than that of ATC. For this reason, in low utilizations, while ATC and BD1 yield almost same WT performance, X-BD1 is better than X-ATC in the same conditions. However, when the system is heavily loaded, there is no improvement, moreover inserted idleness method makes the standard ATC and BDs' performances worse. When the performances of different resource pricing schemes are compared, it is observed that the rules based on dynamic resource pricing (BD1, BD4, X-BD1 and X-BD4) perform well except very high loads in which case the bottleneck resource pricing (BD5 and X-BD5) is better than the others. In general, the performances of static resource pricing and uniform resource pricing are poor in the experiments.

These results are similar when the bottleneck shop is considered. But the differences in the performances between ATC and BD is smaller than those in the uniform shop case. In low and medium loads of the system, some versions of the BD rule are superior to the ATC rule particularly in inserted idleness case. Even in high utilization case, the BD's performance is comparable with the performance of ATC especially in loose and moderate due date settings. In bottleneck job shop case, the bottleneck resource pricing dominates the other resource pricing methods.

Percent of Tardy Jobs and Conditional Weighted Tardiness

The percent of tardy jobs and conditional weighted tardiness performances of different types of ATC and BD are shown in Tables 2.4 and 2.5 and Tables 2.6 and 2.7, respectively.

When the PT performances are compared in uniform job shop case, ATC and BD yield almost the same performance. The inserted idleness does not improve the performances of ATC and BD either. Again, static resource pricing based on queuing approximation is a poor performer among the rules. Bottleneck resource pricing surprisingly yields worse PT performances even in very high utilizations. The similar observations are made for the bottleneck job shop case.

The relative conditional weighted tardiness performances of ATC and BD in uniform job shop case are also similar to those of the weighted tardiness case. Again, the inserted idleness is most effective in low and medium utilizations. X-BD1 and X-BD4 outperforms other resource pricing methods and ATC in low and medium levels of utilization. At high and very high utilizations ATC and X-ATC are superior. The similar results can be drawn from the bottleneck shop case.

Table 2.2: Average Weighted Tardiness Measures of the ATC and BD rules, Uniform Job Shop Case

(a) Loose Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
ATC	146.83	198.23	417.26	880.76
BD1	147.24	206.22	507.03	1248.17
BD2	146.95	230.41	620.58	1589.43
BD3	149.28	214.32	560.91	1373.63
BD4	145.12	209.39	558.37	1214.82
BD5	149.24	206.84	494.83	1102.76
X-ATC	131.95	182.09	406.94	885.33
X-BD1	122.52	178.02	467.35	1167.05
X-BD2	124.58	201.49	667.85	1592.00
X-BD3	124.20	190.08	580.22	1419.00
X-BD4	120.43	182.42	531.79	1212.38
X-BD5	124.95	184.74	495.71	1086.60

(b) Medium Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
ATC	201.09	317.29	691.79	1227.60
BD1	202.67	339.47	806.21	1662.30
BD2	202.35	391.67	1125.48	2341.86
BD3	205.65	377.83	993.37	1938.20
BD4	196.79	352.97	872.54	1686.36
BD5	206.13	357.79	865.29	1491.06
X-ATC	181.34	304.42	683.26	1227.07
X-BD1	170.32	308.73	798.98	1577.28
X-BD2	175.49	372.83	1155.94	2373.23
X-BD3	174.97	345.57	982.55	1892.91
X-BD4	167.76	315.80	839.76	1654.84
X-BD5	178.17	336.13	853.37	1550.78

(c) Tight Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
ATC	327.71	625.56	1172.47	1779.35
BD1	332.95	661.49	1349.67	2290.35
BD2	349.98	822.36	1771.50	3063.80
BD3	344.14	774.69	1594.66	2609.79
BD4	324.24	677.21	1405.39	2324.97
BD5	344.85	705.53	1435.66	2166.39
X-ATC	306.92	601.89	1155.25	1758.84
X-BD1	291.22	629.34	1263.90	2256.51
X-BD2	315.84	773.91	1776.67	3124.66
X-BD3	305.60	746.43	1609.20	2616.07
X-BD4	278.72	633.94	1347.79	2267.84
X-BD5	317.59	695.99	1426.72	2240.86

Table 2.3: Average Weighted Tardiness Measures of the ATC and BD rules, Bottleneck Job Shop Case

(a) Loose Due Dates

Rule	Utilization Level		
	Low	Medium	High
ATC	130.25	159.09	292.93
BD1	129.82	159.86	383.61
BD2	130.03	162.23	430.91
BD3	132.12	161.06	337.87
BD4	126.88	159.80	369.08
BD5	131.51	160.04	304.83
X-ATC	117.32	144.86	280.16
X-BD1	109.04	134.73	353.42
X-BD2	111.55	137.88	417.36
X-BD3	109.77	135.57	308.77
X-BD4	106.41	130.31	348.38
X-BD5	110.88	138.05	290.09

(b) Medium Due Dates

Rule	Utilization Level		
	Low	Medium	High
ATC	177.87	230.07	472.12
BD1	177.22	230.65	645.58
BD2	178.83	241.34	739.14
BD3	178.64	241.44	546.01
BD4	173.19	227.57	660.90
BD5	177.84	237.56	498.64
X-ATC	161.40	213.49	455.50
X-BD1	152.18	198.92	581.16
X-BD2	152.64	207.65	719.44
X-BD3	152.64	202.76	529.36
X-BD4	147.49	196.14	576.24
X-BD5	153.68	208.37	480.75

(c) Tight Due Dates

Rule	Utilization Level		
	Low	Medium	High
ATC	277.19	402.18	770.27
BD1	281.47	407.35	1009.19
BD2	288.23	442.14	1259.12
BD3	286.38	440.16	982.33
BD4	273.98	408.68	1037.11
BD5	287.11	429.39	868.73
X-ATC	261.80	386.81	763.86
X-BD1	245.26	370.22	1017.59
X-BD2	253.68	408.56	1236.73
X-BD3	249.20	399.68	928.53
X-BD4	236.66	371.60	952.79
X-BD5	256.01	400.36	829.51

Table 2.4: Average Percent of Tardy Jobs Measures of the ATC and BD rules, Uniform Job Shop Case

(a) Loose Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
ATC	12.23	15.36	24.22	31.61
BD1	12.17	15.56	23.96	33.94
BD2	12.36	16.30	26.35	36.60
BD3	12.41	15.76	25.16	33.97
BD4	12.33	16.37	25.62	32.40
BD5	12.41	15.77	25.95	34.72
X-ATC	11.65	14.98	23.69	31.86
X-BD1	11.29	14.71	23.42	33.36
X-BD2	11.36	15.49	27.23	36.75
X-BD3	11.28	15.29	25.13	33.76
X-BD4	11.21	15.59	25.05	31.75
X-BD5	11.40	15.10	25.80	34.41

(b) Medium Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
ATC	16.81	24.07	34.49	40.05
BD1	16.61	23.59	33.73	41.45
BD2	16.67	24.93	37.39	45.58
BD3	16.68	24.10	35.41	41.76
BD4	16.63	26.18	35.69	41.11
BD5	16.76	25.26	36.95	43.72
X-ATC	15.88	23.64	34.12	39.73
X-BD1	15.34	22.70	33.40	41.35
X-BD2	15.58	24.52	37.07	45.13
X-BD3	15.53	23.44	34.84	41.06
X-BD4	15.45	24.40	34.93	39.77
X-BD5	15.69	24.74	37.28	43.67

(c) Tight Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
ATC	27.23	40.06	48.92	52.36
BD1	26.92	38.17	47.82	52.64
BD2	27.18	40.40	49.20	54.79
BD3	27.01	39.64	47.61	52.50
BD4	28.14	41.61	49.24	52.32
BD5	27.59	41.56	51.99	55.46
X-ATC	26.70	39.22	48.35	52.12
X-BD1	26.03	37.24	46.00	52.00
X-BD2	26.48	39.24	48.36	54.34
X-BD3	25.91	38.62	47.10	51.30
X-BD4	26.74	40.03	47.73	50.69
X-BD5	26.66	40.98	51.08	55.93

Table 2.5: Average Percent of Tardy Jobs Measures of the ATC and BD rules, Bottleneck Job Shop Case

(a) Loose Due Dates

	Utilization Level		
Rule	Low	Medium	High
ATC	11.47	12.96	18.81
BD1	11.52	12.89	19.24
BD2	11.44	12.98	20.16
BD3	11.53	13.00	18.71
BD4	11.43	13.21	19.96
BD5	11.67	12.92	18.77
X-ATC	11.05	12.24	17.84
X-BD1	10.59	11.96	18.53
X-BD2	10.71	11.94	19.38
X-BD3	10.58	11.84	17.83
X-BD4	10.49	11.95	19.17
X-BD5	10.71	12.02	18.21

(b) Medium Due Dates

	Utilization Level		
Rule	Low	Medium	High
ATC	15.59	18.69	25.97
BD1	15.61	18.79	26.70
BD2	15.49	18.89	27.67
BD3	15.57	18.57	26.36
BD4	15.33	19.66	28.28
BD5	15.60	19.23	27.39
X-ATC	14.98	18.31	25.93
X-BD1	14.70	17.63	26.13
X-BD2	14.62	17.62	27.11
X-BD3	14.58	17.20	25.52
X-BD4	14.51	18.39	27.29
X-BD5	14.78	18.19	26.38

(c) Tight Due Dates

	Utilization Level		
Rule	Low	Medium	High
ATC	24.15	31.74	39.90
BD1	24.11	30.88	39.08
BD2	24.33	31.46	40.50
BD3	24.59	31.19	39.42
BD4	24.47	33.81	41.22
BD5	24.58	31.91	41.53
X-ATC	23.76	31.27	39.69
X-BD1	22.81	29.78	38.87
X-BD2	23.06	30.62	40.19
X-BD3	22.92	29.72	38.40
X-BD4	23.09	32.26	39.88
X-BD5	23.38	31.49	40.51

Table 2.6: Average Conditional Weighted Tardiness Measures of the ATC and BD rules, Uniform Job Shop Case

(a) Loose Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
ATC	1198.05	1291.09	1673.66	2703.27
BD1	1210.77	1324.14	2035.58	3463.59
BD2	1187.20	1415.07	2266.64	4201.65
BD3	1201.20	1359.74	2142.80	3961.16
BD4	1175.89	1280.09	2085.01	3596.65
BD5	1202.38	1309.96	1838.62	3048.62
X-ATC	1129.86	1213.92	1670.96	2680.81
X-BD1	1082.53	1212.33	1916.10	3312.25
X-BD2	1096.47	1300.87	2340.97	4180.15
X-BD3	1101.46	1239.08	2220.89	4114.65
X-BD4	1075.35	1172.23	2043.63	3672.87
X-BD5	1094.34	1223.74	1834.10	3048.12

(b) Medium Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
ATC	1197.11	1320.79	1965.27	3021.52
BD1	1221.52	1438.09	2328.85	3876.64
BD2	1214.69	1567.62	2938.64	5034.36
BD3	1234.19	1564.44	2752.31	4601.78
BD4	1183.76	1348.57	2386.00	4009.12
BD5	1230.81	1416.73	2277.93	3353.61
X-ATC	1143.08	1288.00	1959.10	3038.31
X-BD1	1109.97	1356.63	2324.49	3706.23
X-BD2	1125.92	1520.04	3037.87	5141.60
X-BD3	1126.90	1471.12	2763.57	4567.56
X-BD4	1087.96	1293.01	2349.62	4082.47
X-BD5	1134.85	1353.21	2228.90	3498.59

(c) Tight Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
ATC	1204.01	1556.07	2370.92	3372.87
BD1	1235.75	1727.38	2788.32	4294.22
BD2	1286.77	2026.43	3559.95	5536.28
BD3	1274.13	1945.66	3316.00	4945.36
BD4	1153.17	1624.80	2821.31	4389.52
BD5	1248.02	1690.09	2726.17	3867.89
X-ATC	1149.28	1528.22	2361.05	3340.98
X-BD1	1117.69	1681.08	2714.36	4271.36
X-BD2	1192.03	1962.00	3642.77	5696.79
X-BD3	1177.74	1925.75	3375.92	5073.29
X-BD4	1042.20	1576.60	2794.69	4414.80
X-BD5	1189.91	1690.19	2760.48	3964.07

Table 2.7: Average Conditional Weighted Tardiness Measures of the ATC and BD rules, Bottleneck Job Shop Case

(a) Loose Due Dates

	Utilization Level		
Rule	Low	Medium	High
ATC	1134.18	1225.88	1535.01
BD1	1125.70	1237.18	1907.86
BD2	1132.18	1248.05	2048.03
BD3	1144.10	1236.20	1767.18
BD4	1108.64	1207.77	1772.30
BD5	1125.00	1237.14	1601.38
X-ATC	1060.76	1182.78	1550.72
X-BD1	1029.73	1123.11	1815.47
X-BD2	1040.12	1152.29	2052.37
X-BD3	1036.98	1142.20	1692.49
X-BD4	1016.07	1089.79	1754.26
X-BD5	1037.09	1147.91	1566.70

(b) Medium Due Dates

	Utilization Level		
Rule	Low	Medium	High
ATC	1140.55	1230.82	1801.68
BD1	1135.40	1227.28	2341.52
BD2	1151.91	1275.50	2594.52
BD3	1145.43	1297.65	2051.01
BD4	1126.98	1160.27	2279.40
BD5	1138.72	1234.89	1804.41
X-ATC	1077.78	1166.00	1738.67
X-BD1	1033.92	1126.25	2162.09
X-BD2	1042.99	1176.97	2576.48
X-BD3	1046.24	1175.27	2053.52
X-BD4	1016.08	1069.29	2069.00
X-BD5	1037.53	1141.80	1802.02

(c) Tight Due Dates

	Utilization Level		
Rule	Low	Medium	High
ATC	1147.97	1262.94	1923.08
BD1	1167.00	1314.44	2555.09
BD2	1185.05	1398.59	3085.41
BD3	1162.22	1404.45	2487.11
BD4	1119.06	1204.89	2500.91
BD5	1167.15	1338.70	2084.14
X-ATC	1101.59	1230.90	1915.84
X-BD1	1074.91	1235.75	2587.68
X-BD2	1099.03	1326.63	3057.99
X-BD3	1085.29	1336.26	2406.20
X-BD4	1023.83	1144.94	2375.87
X-BD5	1093.84	1263.71	2039.76

2.4.2 Conventional Rules

Weighted Tardiness Performances

The average weighted tardiness performances of the conventional rules are shown in Tables 2.8 and 2.9. The results indicate that FCFS is the worst of the all rules tested in all due date tightness and utilization levels. WSPT is better than WLWKR in all conditions, which shows that myopic policies in such an environment is more useful in the sense that considering the processing times of the downstream operations do not improve the performance of myopic counterpart which takes into account only imminent processing time information (This also supports our previous result drawn for ATC and BD).

In low utilizations, the performances of the job-oriented rules and operation based rules are indistinguishable. However, when the utilization increases the CR and S/OPN rules dominate other job-based rules if the due dates are loosely set. As the due dates get also tighter, MDD performs better than the other job based rules. MDSPRO does not improve the performance of S/OPN, moreover it is the worst performer among the job-based rules.

When the performances of the operation based rules are compared, MOD is the preferred rule. The performance of CEXSPT is not encouraging. Its performance is comparable only when the utilization is very high. As in the previous studies, the operational rules perform better than the job-based counter-parts. Also the results show that the performances of the job-based and operation-based rules are very sensitive to the system conditions. As the utilization of the system increases or the due dates tighten, the performances of them get worse sharply. The WSPT rule is more robust as compared with these rules. When the performance of WSPT is worse than the those of almost all job- and operation-based rules in low load level, it performs better than these rules in high congestion levels. This effect is magnified as the due dates get tighter.

However, COVERT yield very good performances as compared with the performances of the other rules mentioned above. As the utilization of the system increases there is no great deterioration in the performance of COVERT as in the case of job- and operation-based rules. For this reason, the difference between the performances of job- and operation-based rules and the COVERT becomes significant as the utilization increases and due dates tighten.

Almost similar observations are made for the bottleneck shop case except that the differences in the performances of the rules become smaller.

Percent of Tardy Jobs and Conditional Weighted Tardiness

The average percent of tardy jobs performances of conventional rules are shown in Tables 2.10 and 2.11. The average conditional performances of the rules are presented in Tables 2.12 and 2.13.

For the percent of tardy jobs performance, except the low and medium utilizations with loose due dates, WSPT and even WLWKR are better than job- and operation-based rules. As far as the job-based priority rules are considered, MDD yields consistently better PT performances, while the MOD rule performs robustly well among the operation-based rules. The performances of all of the job- and operation-based rules deteriorate sharply as the due dates tighten and the load of the system increases. Again, COVERT produce good performances. When the bottleneck job shop is considered, the results are similar.

The comparison of the conditional weighted tardiness performances yields similar results as with the weighted tardiness performances. WSPT is again the best in the first group. In job-based rules CR yields lowest CWT performances, while MDD's performance is the worst. This is interesting, since the CR rule is the worst performer for the PT performance in almost all cases, whereas the MDD rule outperforms others in the group in the PT performance. Among operation-based rules, CEXSPT and OCR perform well. The WSPT rule is superior to these rules as the due dates get tighter. Again, COVERT is better than the best of the first three groups in all conditions. The similar results can be drawn from the bottleneck shop case.

Table 2.8: Average Weighted Tardiness Measures of the Conventional Rules, Uniform Job Shop Case

(a) Loose Due Dates

Rule	Utilization Level			
	Low	Medium	High	V. High
FCFS	270.84	909.03	2650.36	5934.37
WSPT	201.77	437.43	909.15	1463.69
WLWKR	230.42	591.03	1263.53	2159.42
EDD	150.12	255.42	1435.63	4362.42
MDD	151.30	260.97	1278.18	4165.99
SLACK	150.65	228.41	1402.89	4217.80
CR(=S/RPT)	149.70	220.41	1050.91	3628.22
S/OPN	148.75	215.68	1038.72	4031.36
MDSPRO	147.32	215.68	1330.43	4747.07
ODD	147.90	255.00	1194.84	3543.51
OSLACK	150.00	260.37	1263.76	3711.24
OCR	149.40	256.30	1054.95	2921.96
MOD	148.56	239.61	979.99	2534.76
CEXSPT	153.88	300.01	1071.15	2818.08
COVERT	147.64	202.95	418.37	882.89

(b) Medium Due Dates

Rule	Utilization Level			
	Low	Medium	High	V. High
FCFS	365.92	1210.14	3388.76	7043.76
WSPT	269.52	565.76	1086.49	1692.42
WLWKR	311.86	781.16	1527.71	2524.15
EDD	209.49	584.55	2503.51	5868.97
MDD	213.10	554.51	2134.46	5804.12
SLACK	207.75	573.30	2458.51	5936.65
CR(=S/RPT)	207.57	476.48	2170.83	5528.48
S/OPN	206.37	496.26	2277.52	5866.23
MDSPRO	205.08	533.90	2524.33	6322.18
ODD	205.56	521.37	2161.46	5212.80
OSLACK	206.90	545.58	2216.47	5403.91
OCR	209.57	490.54	1780.26	4145.71
MOD	204.69	438.85	1507.93	3520.78
CEXSPT	216.49	545.36	1681.82	3876.33
COVERT	201.43	323.40	692.39	1259.55

(c) Tight Due Dates

Rule	Utilization Level			
	Low	Medium	High	V. High
FCFS	555.79	1751.74	4404.52	8485.90
WSPT	404.93	808.86	1400.58	2064.31
WLWKR	471.61	1073.63	2018.03	2957.97
EDD	376.59	1353.00	3924.62	7681.09
MDD	378.99	1228.61	3439.29	7140.49
SLACK	362.47	1397.13	4010.64	7693.57
CR(=S/RPT)	357.33	1248.93	3507.85	7331.51
S/OPN	361.15	1275.03	3718.98	7767.25
MDSPRO	360.02	1392.13	4464.32	8708.66
ODD	354.09	1193.66	3631.92	7284.26
OSLACK	368.25	1249.41	3773.29	7366.64
OCR	353.00	1060.71	2965.05	5657.25
MOD	346.61	929.37	2532.29	4835.45
CEXSPT	362.71	1039.91	2632.79	5049.83
COVERT	333.33	609.73	1144.14	1774.94

Table 2.9: Average Weighted Tardiness Measures of the Conventional Rules, Bottleneck Job Shop Case

(a) Loose Due Dates

Rule	Utilization Level		
	Low	Medium	High
FCFS	198.77	440.14	1840.92
WSPT	159.16	267.11	567.31
WLWKR	173.83	327.47	747.13
EDD	130.87	168.17	1021.45
MDD	130.65	167.31	625.64
SLACK	129.34	162.21	1014.89
CR(=S/RPT)	132.18	164.93	830.97
S/OPN	131.79	163.29	877.52
MDSPRO	132.63	165.52	986.99
ODD	131.75	170.56	1006.97
OSLACK	131.84	172.42	1008.45
OCR	130.53	170.66	876.96
MOD	131.46	168.11	620.60
CEXSPT	135.01	190.68	691.14
COVERT	130.26	161.16	299.74

(b) Medium Due Dates

Rule	Utilization Level		
	Low	Medium	High
FCFS	268.89	590.68	2314.89
WSPT	213.05	351.98	706.50
WLWKR	235.15	428.83	924.49
EDD	179.79	270.74	1594.93
MDD	181.24	271.57	1070.52
SLACK	177.96	267.39	1622.07
CR(=S/RPT)	181.55	259.95	1361.97
S/OPN	181.76	255.06	1419.99
MDSPRO	181.07	261.57	1684.21
ODD	180.91	272.35	1586.62
OSLACK	182.23	279.27	1638.88
OCR	180.90	268.53	1343.85
MOD	179.76	258.78	989.14
CEXSPT	184.22	298.82	1023.26
COVERT	177.83	231.62	469.47

(c) Tight Due Dates

Rule	Utilization Level		
	Low	Medium	High
FCFS	408.33	880.85	3036.75
WSPT	319.32	514.85	948.50
WLWKR	356.46	627.01	1225.80
EDD	296.16	596.26	2568.43
MDD	298.39	579.17	1920.52
SLACK	293.25	593.98	2644.68
CR(=S/RPT)	293.27	544.32	2307.03
S/OPN	292.14	558.95	2408.56
MDSPRO	295.51	586.67	2933.85
ODD	291.43	563.15	2482.22
OSLACK	299.21	586.78	2602.49
OCR	290.37	526.77	2083.65
MOD	287.93	498.38	1583.10
CEXSPT	294.84	544.09	1624.25
COVERT	281.66	405.26	779.52

Table 2.10: Average Percent of Tardy Jobs Measures of the Conventional Rules, Uniform Job Shop Case

(a) Loose Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
FCFS	16.14	28.44	46.42	63.14
WSPT	15.04	22.91	29.67	33.03
WLWKR	15.62	24.85	31.80	35.61
EDD	12.59	19.74	51.75	77.37
MDD	12.49	18.06	41.17	61.28
SLACK	12.46	19.07	52.94	78.67
CR(=S/RPT)	12.93	20.41	52.34	82.14
S/OPN	12.44	17.77	48.56	81.98
MDSPRO	12.41	18.07	49.51	77.94
ODD	12.65	20.03	47.03	76.00
OSLACK	12.85	20.45	48.26	77.57
OCR	13.08	22.63	49.19	75.58
MOD	12.48	17.15	33.01	51.26
CEXSPT	14.80	29.46	52.55	73.60
COVERT	12.49	17.56	28.25	39.97

(b) Medium Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
FCFS	21.64	36.98	56.78	72.21
WSPT	19.95	29.69	36.53	39.99
WLWKR	21.11	31.65	38.54	42.41
EDD	17.44	36.97	69.45	85.83
MDD	17.17	31.08	53.97	67.43
SLACK	17.41	38.02	71.27	87.95
CR(=S/RPT)	17.95	38.41	75.20	90.05
S/OPN	17.65	37.12	72.70	89.87
MDSPRO	17.73	37.39	70.10	87.03
ODD	17.69	35.89	69.06	87.29
OSLACK	17.68	37.17	69.70	88.11
OCR	18.30	38.12	67.77	85.62
MOD	16.88	28.25	48.04	63.72
CEXSPT	20.71	41.23	64.93	80.88
COVERT	17.29	26.64	39.99	49.23

(c) Tight Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
FCFS	32.14	52.78	70.60	82.00
WSPT	29.87	40.81	47.35	51.10
WLWKR	30.91	42.45	49.42	52.28
EDD	30.45	61.29	83.13	91.90
MDD	28.80	49.94	64.95	71.97
SLACK	31.16	64.95	85.34	92.81
CR(=S/RPT)	32.23	67.14	86.56	93.98
S/OPN	31.13	65.55	86.53	93.90
MDSPRO	31.33	64.54	86.15	93.33
ODD	30.53	62.01	85.46	93.59
OSLACK	31.43	63.32	85.91	93.75
OCR	32.13	62.44	83.83	92.00
MOD	28.17	48.45	66.52	76.16
CEXSPT	32.67	59.26	77.32	86.55
COVERT	28.35	42.67	54.92	61.45

Table 2.11: Average Percent of Tardy Jobs Measures of the Conventional Rules, Bottleneck Job Shop Case

(a) Loose Due Dates

	Utilization Level		
Rule	Low	Medium	High
FCFS	13.79	19.76	35.67
WSPT	13.21	17.39	23.33
WLWKR	13.57	18.27	24.94
EDD	11.66	13.88	32.49
MDD	11.49	13.41	24.97
SLACK	11.58	13.69	33.03
CR(=S/RPT)	11.96	14.44	34.58
S/OPN	11.76	13.73	32.45
MDSPRO	11.78	14.04	32.03
ODD	11.72	14.25	33.74
OSLACK	11.71	14.29	33.71
OCR	11.94	15.15	34.25
MOD	11.58	13.38	22.51
CEXSPT	13.00	18.48	35.99
COVERT	11.63	13.70	20.87

(b) Medium Due Dates

	Utilization Level		
Rule	Low	Medium	High
FCFS	18.54	26.31	44.60
WSPT	17.47	22.65	29.38
WLWKR	18.07	23.72	31.21
EDD	15.74	21.62	46.75
MDD	15.57	20.25	36.14
SLACK	15.82	22.21	48.34
CR(=S/RPT)	16.42	23.19	49.79
S/OPN	15.90	21.57	48.11
MDSPRO	15.80	22.08	48.66
ODD	15.96	22.26	47.53
OSLACK	16.07	22.88	48.59
OCR	16.22	23.51	46.90
MOD	15.63	19.66	32.43
CEXSPT	17.19	26.83	45.75
COVERT	15.77	19.81	29.25

(c) Tight Due Dates

	Utilization Level		
Rule	Low	Medium	High
FCFS	27.44	38.72	58.06
WSPT	26.10	33.11	40.24
WLWKR	26.88	34.67	41.85
EDD	25.95	39.87	64.85
MDD	24.85	34.57	50.82
SLACK	25.86	41.62	67.52
CR(=S/RPT)	26.70	42.57	70.07
S/OPN	26.18	41.92	68.98
MDSPRO	26.30	42.01	69.06
ODD	25.54	40.46	66.61
OSLACK	26.29	41.55	67.78
OCR	26.42	40.86	65.20
MOD	24.72	33.57	48.86
CEXSPT	26.98	40.72	59.05
COVERT	24.95	32.93	43.61

Table 2.12: Average Conditional Weighted Tardiness Measures of the Conventional Rules, Uniform Job Shop Case

(a) Loose Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
FCFS	1678.89	3172.21	5550.06	9036.51
WSPT	1339.30	1904.41	3029.22	4388.99
WLWKR	1473.84	2376.20	3943.15	6045.17
EDD	1192.19	1294.20	2517.14	5319.38
MDD	1213.16	1445.21	2873.67	6658.48
SLACK	1208.87	1202.30	2401.46	5038.34
CR(=S/RPT)	1155.71	1089.54	1804.32	4198.97
S/OPN	1194.38	1213.49	1907.44	4694.74
MDSPRO	1186.96	1197.16	2346.90	5531.11
ODD	1166.61	1268.37	2281.33	4303.90
OSLACK	1166.54	1266.90	2349.57	4438.78
OCR	1141.52	1132.14	1966.86	3563.50
MOD	1188.96	1390.32	2735.29	4530.68
CEXSPT	1038.02	1019.87	1895.94	3610.59
COVERT	1181.02	1159.61	1437.41	2119.84

(b) Medium Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
FCFS	1686.11	3243.93	5802.92	9486.40
WSPT	1348.82	1900.75	2944.15	4196.27
WLWKR	1474.39	2461.24	3943.23	5941.18
EDD	1202.65	1552.20	3415.34	6650.64
MDD	1243.95	1765.12	3806.82	8571.94
SLACK	1195.63	1489.80	3257.92	6600.44
CR(=S/RPT)	1159.85	1235.68	2745.46	6052.75
S/OPN	1171.25	1323.33	2951.23	6436.40
MDSPRO	1160.86	1410.22	3338.87	6949.55
ODD	1163.04	1438.50	2936.14	5773.52
OSLACK	1171.45	1450.87	2997.90	5964.45
OCR	1149.22	1278.78	2477.43	4662.27
MOD	1214.00	1542.49	2976.54	5288.37
CEXSPT	1047.31	1309.08	2485.35	4645.21
COVERT	1165.79	1217.11	1691.78	2505.55

(c) Tight Due Dates

	Utilization Level			
Rule	Low	Medium	High	V. High
FCFS	1725.62	3291.81	6127.35	10162.27
WSPT	1353.76	1976.83	2936.85	4012.14
WLWKR	1522.26	2523.57	4063.74	5646.17
EDD	1236.24	2180.65	4628.56	8274.70
MDD	1314.97	2440.02	5240.76	9920.11
SLACK	1163.69	2124.51	4616.84	8208.35
CR(=S/RPT)	1110.00	1839.66	3985.67	7758.00
S/OPN	1161.61	1923.21	4226.85	8234.85
MDSPRO	1151.61	2117.56	5066.74	9189.47
ODD	1160.09	1902.37	4169.53	7713.69
OSLACK	1173.35	1952.88	4315.35	7790.93
OCR	1099.27	1684.09	3462.67	6058.07
MOD	1231.05	1899.03	3731.03	6243.02
CEXSPT	1110.69	1736.23	3328.78	5762.90
COVERT	1176.28	425.45	2061.41	2859.44

Table 2.13: Average Conditional Weighted Tardiness Measures of the Conventional Rules, Bottleneck Job Shop Case

(a) Loose Due Dates

	Utilization Level		
Rule	Low	Medium	High
FCFS	1445.69	2199.61	4809.55
WSPT	1203.81	1527.13	2422.85
WLWKR	1281.68	1778.61	2975.70
EDD	1120.09	1212.50	2542.98
MDD	1134.73	1245.80	2302.61
SLACK	1117.11	1187.85	2500.99
CR(=S/RPT)	1106.95	1145.91	1959.98
S/OPN	1118.83	1195.17	2163.66
MDSPRO	1124.43	1180.84	2411.15
ODD	1122.16	1195.24	2477.96
OSLACK	1124.65	1203.86	2475.06
OCR	1090.44	1124.77	2184.88
MOD	1133.24	1251.93	2580.47
CEXSPT	1036.87	1036.19	1802.34
COVERT	1116.03	1172.75	1409.01

(b) Medium Due Dates

	Utilization Level		
Rule	Low	Medium	High
FCFS	1448.96	2218.22	4892.83
WSPT	1218.11	1545.59	2399.70
WLWKR	1300.24	1796.64	2946.15
EDD	1140.40	1247.37	3058.14
MDD	1161.56	1339.52	2839.80
SLACK	1125.58	1202.06	3051.90
CR(=S/RPT)	1105.75	1125.91	2495.32
S/OPN	1142.60	1179.99	2643.59
MDSPRO	1146.42	1181.76	3097.23
ODD	1132.13	1216.40	3022.71
OSLACK	1132.62	1212.22	3048.57
OCR	1116.57	1139.14	2605.83
MOD	1148.43	1306.70	2955.59
CEXSPT	1074.07	1108.04	2159.64
COVERT	1127.69	1169.61	1586.54

(c) Tight Due Dates

	Utilization Level		
Rule	Low	Medium	High
FCFS	1485.66	2250.33	5048.67
WSPT	1223.21	1547.54	2352.95
WLWKR	1326.05	1800.11	2919.82
EDD	1141.75	1469.77	3796.33
MDD	1200.09	1659.66	3735.74
SLACK	1134.14	1401.32	3769.25
CR(=S/RPT)	1100.30	1261.20	3183.96
S/OPN	1116.67	1312.53	3365.21
MDSPRO	1124.54	1369.42	4064.76
ODD	1141.91	1373.92	3568.45
OSLACK	1138.12	1388.50	3685.93
OCR	1099.73	1274.38	3060.05
MOD	1164.14	1470.39	3174.10
CEXSPT	1092.47	1325.76	2709.38
COVERT	1127.50	1224.95	1775.71

2.4.3 Cross Comparisons of the Rules

Weighted Tardiness Performances

In low utilizations, inserted idleness versions of BD with dynamic resource pricing and bottleneck resource pricing (X-BD1, X-BD4, X-BD5) draw the lower envelope for WT performance. In this case, performances of COVERT and ATC are very similar. In medium utilizations with loose and moderately tight due dates, inserted idleness versions of ATC and BD1 outperform other rules. In high and very high utilizations, COVERT and ATC produce similar performance measures. In very high utilizations, the performances of the BD rules are even worse than that of WSPT. But they are still better than job-based and operation-based rules significantly.

But the results are different in bottleneck job shop case. In this case, except high utilizations the inserted idleness versions of dynamic and bottleneck resource pricing are better than all other rules. In high utilizations, X-ATC performs better than the other rules.

Percent of Tardy Jobs and Conditional Weighted Tardiness

For percent of tardy jobs performances in uniform shops, except very high utilization case, the performances of X-ATC and X-BD1 are better than the others. In very high utilizations ATC, X-ATC, X-BD4 perform well. In bottleneck job shop case, the inserted idleness versions of ATC and BD5 are consistently better than other rules.

In uniform shops, almost all the X-BD rules yield better CWT performances in low utilizations. In higher loads, the COVERT rule is better than ATC and BD versions. In bottleneck case, again in low utilizations X-ATC and X-BD rules are best performers. In medium utilizations, their performances are almost the same with that of COVERT. In high utilizations, COVERT is better.

2.5 Conclusions

The experimental results reported in the previous section support the earlier findings about the priority dispatching rules that use the due date information. For example, priority rules which make use of operational information such as operation due dates and operation processing times are consistently better than the job-based rules. In addition, it is observed that WSPT is a relatively good performer in tardiness performances as well as in flow time measures especially in congested shops with tight due dates. Other results can be summarized as follows:

- The complex and composite rules designed for weighted tardiness objectives, such as COVERT, ATC and BD, are very effective in tardiness related performances as compared with the job-based and operation-based rules such as MDD, MOD, etc. Although MOD is reported as a good performer in unweighted tardiness measure in previous studies, it is outperformed by COVERT, ATC, and BD in our study.

- The standard BD rules are not significantly better than the COVERT and ATC rules. Especially, as the utilization of the shop increases, the relative performances of the BD rules get worse. It is surprising because BD uses more global information than ATC and other rules. However, it does not make good decisions in dynamic job shop environment. These results contradict with those of Lawrence and Morton [35]. Their experiments in multi-project scheduling show that BD with global resource pricing is better than myopic counterparts. But their system is static, and their scheduling problem is different: they consider resource-constrained multi-project scheduling problem.

- Inserted idleness improves the BD performances significantly in low and medium utilizations. For example, for weighted tardiness measures in uniform job shop, the improvement achieved by implementing inserted idleness over the standard versions of ATC and BDs increases up to

16.80%. But in high and very high utilizations, there is no improvement achieved by inserted idleness, moreover it worsen the performance of some resource pricing schemes. For this reason, in low and medium utilization, inserted idleness versions of BD can be suggested with the implementation of dynamic or bottleneck resource pricing schemes. In this research, the jobs that are considered for the inserted idleness are the jobs that are currently processing on the machines and whose next operations are the machine under consideration. But this may only be small expansion on the candidate job set. To achieve further improvement on the performances, all the jobs in the system must be considered and their arrival times of these jobs to the machine should be accurately estimated.

- The results of bottleneck shop are somewhat different from those of uniform shop case. Some of them are reported in computational results section. The performances of some standard ATC and BD with proper resource pricing (dynamic or bottleneck) produce lower tardiness measures, especially in bottleneck shop in most of the cases. Therefore, resource pricing distinguishes the resources as bottleneck and non-bottleneck. Furthermore, the global nature of BD includes these resource prices of downstream operations with their operation processing times to the priority index. The results indicate that this mechanism is more useful in a bottleneck shop than in uniform shop. It can be expected that as the load difference between the bottleneck machines and the average machines increases this effect is magnified.

- As Morton and Pentico [39] state, the lead time estimation is the most important issue in ATC and BD. When the tail lead time of the operations is accurately estimated, the local resource constrained slack or estimated lateness of the job is also well estimated. By this way, the ATC and BD rules detects the most urgent job accurately. In our study, we see that the performances of these rules are very sensitive to the waiting time estimation parameter (b). We test some b values, and select the best among them. But this could not be sufficient to achieve good results. For this reason, different lead time estimation methods can be incorporated to the ATC and BD scheduling. In this study, one of them (BD4 and X-BD4 that depend on Kanet and Zhou's waiting time estimation [29]) is firstly tested with other pricing methods. The results of this version are good, but it is not enough to make accurate estimations and to get high-quality performances. Also there are some other methods to accurately estimate the waiting times. One of them is the multi-pass version of these rules known as the lead time iteration as studied in some of the previous studies. These studies show that the LTI improves the performances of ATC and BD significantly [60], [35]. But the implementation of LTI requires either a static system or a perfect knowledge about the future job arrivals and its characteristics in dynamic shops. In Chapter 4, the ATC and BD rules are further tested with and without LTI in a more realistic scheduling environment.

- The multi-pass iterative simulations of the systems are shown to be very effective to improve the performance measures. In these methods, before passing the implementation of one schedule, some iterative simulations of the system are conducted to select the best schedule producing best performance. In some of the previous studies, with iterative simulations, the candidate rules are tested for some period, and then the best rule is implemented up to the next rescheduling point in time [63], [24], [25]. These types of multi-pass scheduling heuristics might be a base of neural network based or expert systems based approaches [62], [12]. At this point, the scheduling and rescheduling issues come to scene. The length of the rescheduling intervals (known as the forecasting horizon) and the length of the implementation period (known as the scheduling period) are two important parameters in such a study. When we consider all of these, the rolling horizon concept is an important subject to deal with. The rolling horizon device is very helpful in improving the performances of the system especially in dynamic and stochastic environments [45]. In Chapter 4, we investigate these issues in detail.

Chapter 3

Reactive Scheduling

3.1 Introduction

In Chapter 1, we have defined static and dynamic scheduling problems and discuss the general solution approaches to these problems. One of these approaches, which is priority dispatching, has been already studied in detail in Chapter 2 for dynamic and deterministic job shop scheduling problems.

In this chapter, we combine event-driven rescheduling and partial rescheduling approaches (discussed in Section 1.4) to propose a reactive scheduling and control methodology. We classify the unexpected events into three types, each of which requires either no reaction, partial rescheduling or rescheduling. We investigate several reactive scheduling policies (or rerouting mechanisms) using priority dispatching approach. Next section reviews the relevant literature. The proposed reactive scheduling and control system is presented in Section 3.3. Section 3.4 summarizes the experimental conditions. This is followed by the discussion of the experimental results in Section 3.5. Finally, conclusions are drawn in Section 3.6.

3.2 Literature Review

The bulk of the published literature on the job shop scheduling problem deals with the task of schedule generation. Reactive scheduling and control issues have not been adequately addressed in the literature. Most of the previous work uses either rescheduling algorithms or utilizes priority dispatching.

Muhlemann, et al. [43] investigate the frequency of scheduling in a dynamic job shop environment where the processing time variations and machine breakdowns may occur randomly. At each scheduling point, static schedule for current jobs is generated by a dispatching rule. Six rescheduling frequencies are tested, ranging from 4 to 40 hours. New jobs are added to the schedule at the rescheduling points. As anticipated, performance generally deteriorates when the rescheduling period increases. More frequent revision of the schedule makes the schedule more up-to-date and the schedule follows the system dynamics at right times. Since the revision of the schedules are made by using priority dispatching the response time is not a problem. The only problem is the increase in the system nervousness. The experiments also show that the shortest processing time (SPT) rule is the best in the overall performance when rescheduling is less frequent. For short rescheduling periods, however, the truncated SPT and some composite rules are found superior. In these experiments, the priority dispatching is used for generating static schedules for current jobs, and new arrivals until the next rescheduling point are out of consideration. In these cases, since the rescheduling does not incur excess time, the rescheduling can be made as a dynamic dispatching by considering new arrivals and selecting maximum priority job from available jobs when a machine becomes available.

Schedule revisions only after significant unexpected events or interruptions, such as machine breakdowns, are investigated by Yamamoto and Nof [66]. They propose a three-phase scheduling/rescheduling scheme: (1) *planning phase*: part-mix assignment, initial scheduling, machine loading. (2) *control phase*: machine loading execution, progress monitoring, and testing for abnormal status. (3) *rescheduling phase*: rescheduling, revising the machine loading, and resorting to the control phase. This scheme is investigated on several systems. Three alternative scheduling/rescheduling algorithms are used in the experiments: (1) a branch and bound method to search a schedule tree, used both for scheduling and rescheduling; (2) same branch and bound method for scheduling, but for rescheduling only time-shifting is applied on the same original sequence; (3) a procedure based on priority dispatching rules, both for scheduling and rescheduling. Rescheduling is triggered in all algorithms whenever a random machine breakdown occurs. In the experiments the make-span is used as the performance measure. The experimental results show that scheduling/rescheduling procedure based on branch and bound generates an initial schedule with lower make-span values than the schedules generated by priority dispatching rules. When used for both scheduling and rescheduling it also yields better performance measure than the fixed schedule and priority dispatching procedures. Furthermore, better results are obtained by the fixed scheduling (only time shifting is allowed) than by the priority dispatching, possibly due to the robustness of the initial schedule generated by branch and bound. This implies that if the original schedule is well-planned, too frequent rescheduling may be unnecessary. Therefore, Yamamoto and Nof [66] conclude that the schedule must be revised by rescheduling at points in time when the current progress of the system deviates from the generated schedule over a prespecified limit. But as they state, the determination of this limit is an important issue because this method will prevent too frequent, not necessarily beneficial, and often disruptive schedule revisions.

Jain and Foley [26] investigate the effects of the machine breakdowns in a flexible manufacturing environment. In this study, it is assumed that there is a base schedule at the beginning and the objective is to follow the planned schedule as closely as possible. The unexpected event considered is machine breakdown and two on-line reactive scheduling policies are compared: (1) rerouting the jobs scheduled to broken machine to alternative machines, and (2) holding the interrupted jobs with high priority until interruption is removed. The experiments conducted on different levels of machine breakdown and in different utilization levels show that rerouting always outperforms the holding the jobs.

Dutta [16] develops a knowledge-based scheduling and control system for a flexible manufacturing system. The proposed method monitors the system and takes corrective actions in case of unexpected events. The study considers three types of events that occur randomly: (1) machine breakdowns, (2) new job arrivals, and (3) dynamic increase in job priority. At first, an initial static schedule is generated by using static job-based priorities. In case of unexpected event, the control system takes one of the corrective actions according to the system conditions: (1) rerouting the affected jobs to the alternative machines, and (2) preempting scheduled jobs. The experiments show that good performances can be achieved by taking any one of the corrective actions.

Nof and Grant [44] propose a adaptive/predictive scheduling and control mechanism that includes five basic components: (1) *Scheduler* generates a schedule of operations for a given set of factory orders based on the performance and for given objectives. (2) *Monitor* monitors the current progress of the actual execution of the generated schedule, and the ongoing new demands on the factory. (3) *Comparator* compares the actual execution and demands with the planned schedule, i.e., comparing the findings of the monitor with the plan of the scheduler. (4) *Resolver* decides and selects for given criteria how to respond to the results of the comparator, what adaptation strategy to follow during the next period, mainly proceed with the current schedule or resort to a higher level rescheduling. (5) *Adaptor* adapts the current schedule if the decision of the resolver is to respond by some automatic recovery procedure, or by rescheduling.

The initial experiments conducted to test the feasibility and effect of the proposed system consider two types of environment. First environment is a small manufacturing cell and the source of disruption is the processing time variation. In this case, two types of recovery are used based on the deviation of performance of the actual progress from the planned performance generated by the scheduler. When the deviation is in the limit of the first tolerance fence (for example $\pm 10\%$ of the expected performance), the time shifting of the current schedule is applied. When the deviation is detected during the monitoring, rescheduling of all uncompleted orders is triggered (This is an example of performance-driven rescheduling defined in Chapter 1). The experimental results show that this approach is useful and monitoring interval is an important factor for the success of this approach.

In the second environment, machine breakdown and unexpected order arrival are simulated. Again some tolerance fences are defined to select the appropriate recovery policy: When the performance is viewed as normal, no recovery is applied. If the deviation from the planned performance is high, then one of three reactive policies is activated: (1) Rerouting the jobs to alternative machine, (2) order-splitting (since the production consists of two part types and they are produced in batches), and (3) rescheduling all the orders by not considering the alternative machine. The second set of experiments show that even a relatively weak recovery policy yields better performance than no recovery at all. Rescheduling is found to be better than the rerouting. As the authors state the experiments are very limited to conclude more general results. Hence, the research on the additional features is required.

Bean et al. [9] consider the rescheduling of operations in a system when disruptions prevent the use of a preplanned schedule. The proposed approach is to follow the generated schedule until an interruption occurs. In the case of an interruption, part of the schedule is reconstructed to match up with the preschedule at some future time. This approach is compared with the preplanned static scheduling (follow the initial schedule), dynamic priority dispatching with several rules and total rescheduling under different disruptions such as machine breakdowns, and due date changes. The results of the test problems demonstrate the advantages of the match-up approach.

By using the idea of match-up scheduling, Akturk and Gorgulu [2] propose a reactive scheduling algorithm against machine breakdowns. The authors develop a new rescheduling strategy and a match-up point determination procedure through a feedback mechanism to increase the schedule quality and stability. The proposed approach is compared with different alternative reactive scheduling methods. The experimental results show the superiority of the proposed approach. Also it is concluded that the initial schedule has an important effect on the rescheduling problem.

Matsuura et al. [37] investigates the problem of selection between *sequencing* and *dispatching* as a scheduling approach in a job shop environment. In the first set of experiments two approaches are compared with make-span as a performance measure under stochastic environment where random machine breakdowns, job specification changes (operation cancelation or insertion) and arrivals of rush jobs occur in the system. The approaches compared are fixed scheduling (sequencing) and priority dispatching with either FCFS or SPT. In the sequencing approach, an initial schedule is generated by branch-and-bound for the initial job set (there are five jobs initially), and this schedule is maintained regardless of the unexpected events occuring over time. In this set of experiments, when the rates of occurrences are small the sequencing is better than the dispatching. As the rate of events increases, the dispatching outperforms sequencing. From the insights gained in these pilot experiments, the investigators propose a *switching* approach where the sequencing approach is followed until the first occurrence of an event and then dispatching is applied until the end of the horizon. The experiments show that switching always produces the lower make-span performance than the dispatching policy.

Bengu [10] proposes a simulation-based scheduler that uses the up-to-date information about the current status of the system and aims to improve the performance of a scheduling rule (ATC) with the simulation under dynamic and stochastic production environment. In this study, a typical electronics assembly facility manufacturing electronics product is simulated with machine breakdowns. The aim of the use of the simulation scheduler in such an environment is to select the best look-ahead parameter value for the ATC rule with iterating the simulations. The experiments show that the value of the look-ahead parameter in ATC affects the performance of the rule, and simulation-based scheduler is a very effective way of finding a good value for this parameter.

Recently, Kim and Kim [31] propose another simulation-based real-time scheduling mechanism for a flexible manufacturing system. There are two major components, a simulation mechanism and a real-time control system. The simulation mechanism evaluates various dispatching rules for a given job set and selects the best one for a given criterion. The real-time control system periodically monitors the system and checks the system performance value. The best dispatching rule determined by means of simulation is used until the difference between the actual performance and the estimated performance exceeds a given limit (called *performance limit*); then a new simulation is performed with remaining operations, and a new rule is selected. The time between two consecutive comparisons of estimated and actual performance gives the *monitoring interval.* The performance deterioration occurs since there are urgent job arrivals and machine breakdowns. The approach classifies these events as major and minor: the urgent job arrivals and machine breakdowns with longer expected repair times are major disturbances and machine breakdowns with shorter repair times are minor events. In the case of a major event, the simulation mechanism is activated, whereas affected jobs are rerouted to the alternative machines in case of minor events. The experiments show that the monitoring interval and performance limits should be carefully designed to achieve better performances. Finally, the response time is relatively small for the simulation mechanism to apply the method in a real-time manner (The simulation-based scheduling systems will be reviewed in detail in Chapter 4).

There are other studies that do not actually investigate reactive scheduling problem, but analyze their scheduling mechanisms under stochastic events and variations. For example, He et al. [22] analyze the effects of processing time variation on the performance of the priority dispatching rules in a dynamic job shop environment. The experiments show that when the inaccuracy of processing time estimation is not large, the inaccuracy does not significantly affect the performance of the rules. The experiments also show that due-date based rules are more robust than the other rules. At higher levels of processing time variation, there is a significant deterioration of the performance of the rules (Processing time variation will be analyzed as a disturbance in Chapter 4). Byeon et al. [11] and Wu et al. [64] investigate the effects of processing time variations. Their approaches do not explicitly include any reactive scheduling policy. Instead, they construct some critical part of the schedule in advance and complete the remaining part according to the dynamic events such as processing time variations occurring in time. Karabuk [30] compares on-line and off-line scheduling approaches under deterministic and stochastic environments. He considers processing time variation and machine breakdowns. The results show the superiority of the off-line approach even in a stochastic environments. Some of the Artificial Intelligence (AI) based scheduling systems include reactive scheduling and control models. Among them ISIS developed by Fox and Smith [18] and OPIS proposed by Smith et al. [56] are AI based systems that combine scheduling and control aspects of the problem.

Finally, Morton and Pentico [39] propose a three-level reactive scheduling/control system. Since the current study is based on their proposal, the detailed discussion is deferred to the next section.

3.3 Reactive Scheduling

3.3.1 Observations

From the literature review, we can make following observations:

- The majority of the studies treat the reactive scheduling problem as a rescheduling process. Hence, rescheduling is made regardless of the effect and importance of the event. In this case, the system might be in a permanent state of rescheduling if many events occur in succession. Also, this rescheduling approach increases the system nervousness by revising the schedule frequently. Also, rescheduling every time in response to every change may not be economical. Moreover, if the scheduling scheme is off-line and the time needed to generate a schedule is long with this scheme, the rescheduling may not be realized in real-time. For these reasons, we should separate events that really require rescheduling from those that do not need rescheduling so that rescheduling is implemented in a controlled manner.

- No reaction approach is not a proper decision either. Because the existing schedule does not follow the current system conditions. There could be a significant gap between the generated schedule and the current progress of the system. If the unexpected events occur frequently with long durations, the procedure does not catch the events that really require rescheduling. Also, some events can easily invalidate the current schedule.

- In the existing studies, the long-run performances of the reactive policies are not measured. However, the short-term winners may not be very effective in the long-run.

- In general, researchers use a restrictive set of factors in their experimental studies. There is no single study that investigates reactive scheduling policies at various levels of utilization, due date tightness, etc. In addition, some of the studies do not consider the factors that directly affect the performances of the reactive scheduling approaches such as duration and frequency of the unexpected events.

- In none of these studies, the material handling system (MHS) is considered. However, some properties of the MHS such as load and speed might become very important especially when the rerouting mechanisms are considered as reactive scheduling policies.

Hence, the purpose of this study is to develop a reactive scheduling methodology and to test reactive policies proposed according to this methodology.

3.3.2 Scheduling System

The scheduling system under consideration is assumed to have two levels: (1) *Schedule generation (predictive scheduling)*, and (2) *Schedule adjustment and control (reactive scheduling)*.

For the first stage, there are at least two ways to generate schedules: *off-line* *scheduling*, and *on-line scheduling.* Their advantages and disadvantages have been already discussed in detail in Chapter 1. In our study, we use on-line priority dispatching as a schedule generation approach.

In the second stage, schedules or scheduling decisions are revised in response to unforeseen events. First, an appropriate reactive mode should be selected. The selected reactive mode develops the necessary low-cost corrective actions to response these events and is called according to the type, duration, and effect of the event. We can use three reactive modes recommended by Morton and Pentico [39]:

- *Dispatch mode* for minor events.
- *Mid-reactive correction mode* for mid-level interruptions.
- *Major-reactive correction mode* for major interruptions.

The first mode watches the system and uses the current schedule. This corresponds to no reaction approach described in the first section. Minor events that require dispatch mode might be small processing time variations and minor machine breakdowns. The mid-reactive correction mode makes local modifications and revisions in the current schedule. By this way, partial scheduling is considered as a reactive policy. Job recycles for rework, job waiting for missing input, moderate-level machine breakdowns might call this mode. In the last mode, all schedule is regenerated from scratch. Major order cancelations, long machine breakdowns, bottleneck machine downs might be considered in the third type. In fact this three-mode approach combines the event-driven approach, which is a combination of no reaction and rescheduling methods, and the partial rescheduling approaches discussed previously.

3.3.3 Experimental Considerations

In this section, we test the effectiveness of the first two modes of the reactive scheduling model proposed by Morton and Pentico [39] under machine breakdown. The major reactive mode is not considered because it is equivalent to generating job priorities at stage 1 of the scheduling system. To differentiate the event types, we vary the frequency and the duration of breakdowns in the experiments.

For this purpose, single-pass version of *Bottleneck Dynamics* (BD) dispatching is used as a scheduling heuristic [39]. Recall that the BD rule has been discussed in detail in Chapter 2. This heuristic estimates the cost of delaying each operation (*activity price*) and estimates costs of using each resource (*resource price*) and trades off these prices leading to a benefit/cost ratio. BD uses the ratio as a priority to dynamically schedule the jobs. It is also flexible to incorporate the changes in the system and it is possible to modify activity and resource prices to take into account unexpected events. Also, there are routing rules that can be used in the reactive policies that include rerouting mechanisms. For these reasons, BD is used in this study.

In the current study, machine breakdown event is used as an interruption factor. Four alternative reactive scheduling policies are proposed and compared in the experiments. These are:

1. *No rerouting* (no reaction, NR). This corresponds to a dispatch mode.
2. *Queue rerouting* (QR). This policy reroutes the jobs in the queue of the broken machine to the alternative machines and accept the new arrivals to the broken machine.
3. *Arrival rerouting* (AR). This policy keeps the jobs in the queue of the broken machine and reroutes the jobs that will arrive during the repair time period to the alternative machines.
4. *All rerouting* (AAR). It reroutes all the jobs in the queue plus all the new arrivals to the alternative machines.

Rerouting of a job to one of the alternative machine is performed by using BD routing principles. For each alternative machine, we calculate a cost of the route that contains the machine, and select the route with the minimum cost. The cost function of the route contains the resource prices of alternative machine and downstream machines, processing times on these machines and the expected completion time of each downstream operation. By this way, BD tries to select the alternative machine which is non-bottleneck in job's route.

From the observations made in the previous section, we consider a material handling system (MHS) in the experimental study. Since MHS can affect the performances of different rerouting mechanisms, we conduct the experiments with changing MH factors such as speed and load of the MHS. The scheduling of each MH device in the MHS is made by using BD priority dispatching.

3.4 Experiments

In the experiments, a classical dynamic job shop environment is simulated as described in Chapter 2. Here, we outline basic differences in the current experimental design used in this chapter. There are 10 machines which are subject to break down. The operations are randomly processed through machines. Operation processing times are drawn from a uniform distribution U[1,30]. Jobs weights are drawn from U[1,30]. Due dates are assigned randomly over a full range of flow allowances, with an average of 6.0 and 2.0 times the mean total job processing time for relatively loose, and tight due date setting, respectively. There are two utilization levels: Low (52%) and high (68%). The utilizations are calculated by subtracting the down times of the machines. Average weighted tardiness is considered as a system performance measure.

The transportation time comes from a uniform distribution and has a mean which is determined according to the mean transportation time/mean processing time ratio. This ratio can take one of the values 0.15, 0.30, and 0.35. The loaded MH device is 20% slower than the empty device and the loading and unloading times are both equal to 0.25 time units. There are 4 or 5 MH devices in the system.

Machine breakdowns are modeled by using busy time approach proposed by Law and Kelton [34]. With this approach a random uptime is generated for each machine from a busy time distribution. The machine is considered as up until its total accumulated busy time reaches the end of the generated uptime. Then it fails for a random time drawn from a down time distribution, after which an uptime is generated. Law and Kelton [34] recommends that in the absence of real data, busy time distribution is most likely to be a Gamma distribution with a shape parameter (α_b) which is equal to 0.7 and a scale parameter to be specified according to the experimental conditions. The authors also state that Gamma distribution with a shape parameter (α_d) fixed at 1.4 is appropriate for the distribution of down times. In this framework, the level of machine breakdown is measured by efficiency level which gives the long-run ratio of machine busy time to total busy and down time. In fact, this ratio is changed to generate desired levels of machine breakdowns. This is done by fixing the ratio of mean busy time to the sum of mean busy and down time. By this way, duration of each breakdown comes from

$$Gamma(\alpha_b = 1.4, \beta_b = d_{avg}/1.4)$$

and busy time between two successive breaks is drawn from

$$Gamma\left(\alpha_d = 0.7, \beta_d = d_{avg} \times \frac{\epsilon}{0.7(1-\epsilon)}\right).$$

Here, d_{avg} represents mean duration of breakdown and ϵ gives the efficiency level. The mean duration of breakdown can take values p_{avg}, $5p_{avg}$, $10p_{avg}$, $15p_{avg}$, $20p_{avg}$ where p_{avg} is the average operation processing time. The efficiency has two levels as 80% and 90%. By this way, a smaller mean duration of breakdown with the same efficiency represents higher frequency of short breakdown times.

In addition to the main machine, there are two alternative machines randomly selected from existing machines in the case of machine breakdowns. The processing time on alternative machine may be the same as on the main machine (*equal alternative machine*) or it can last 20% more time (*unequal alternative machine*).

The system is simulated by using SIMAN simulation language with linking C subroutines in UNIX environment. We use method of batch means to compare the results. We determine a warm-up period for the system as 1,200 job-completion. We make 10 batches with 1,500 jobs each, leading to 16,200-job run.

3.5 Experimental Results

First we analyze the no-material handling case (in which MHS is not modeled). Figures 3.1–3.4 show plots of experimental results for this case. Each graph depicts the average weighted tardiness performances of four reactive policies for changing mean duration of breakdown. The graphs also show the performances of the no-breakdown case on the vertical axis for 0 duration of breakdown point. As moving from left to right on the horizontal axis, while the mean duration of breakdown increases from p_{avg} to $20p_{avg}$, the overall utilization and the efficiency of the system is almost the same as aimed. Therefore, the movement from left to right does not only imply that the durations of the breakdowns become longer but also indicate that machines break down in lower frequency to achieve the same efficiency level. By this way, the utilization and the efficiency level is the same for all plots in each graph.

As seen from these figures, when there are machine breakdowns, the system performance deteriorates regardless of the level of interruption. This deterioration is high for long mean durations of breakdown and for low efficiency. This effect is magnified when the due dates are tight and the system load is high.

The results show that the relative performances of the reactive policies are not affected by the due date tightness level. All of the plots of the policies show upward shift and their order remains the same.

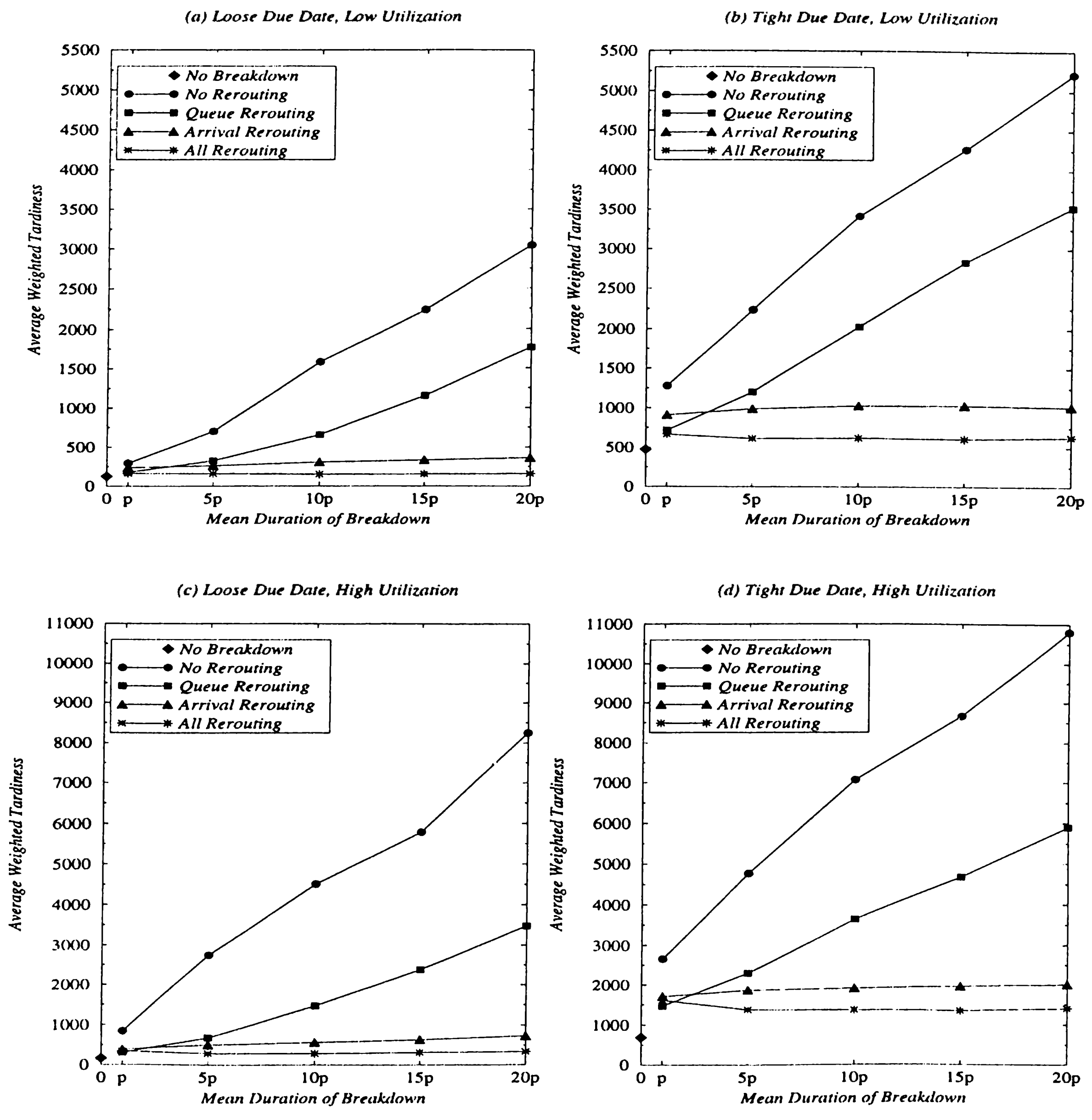

Figure 3.1: Average Weighted Tardiness versus Mean Duration of Breakdown with no Material Handling Consideration (Equal Alternative Machine, Efficiency=80%)

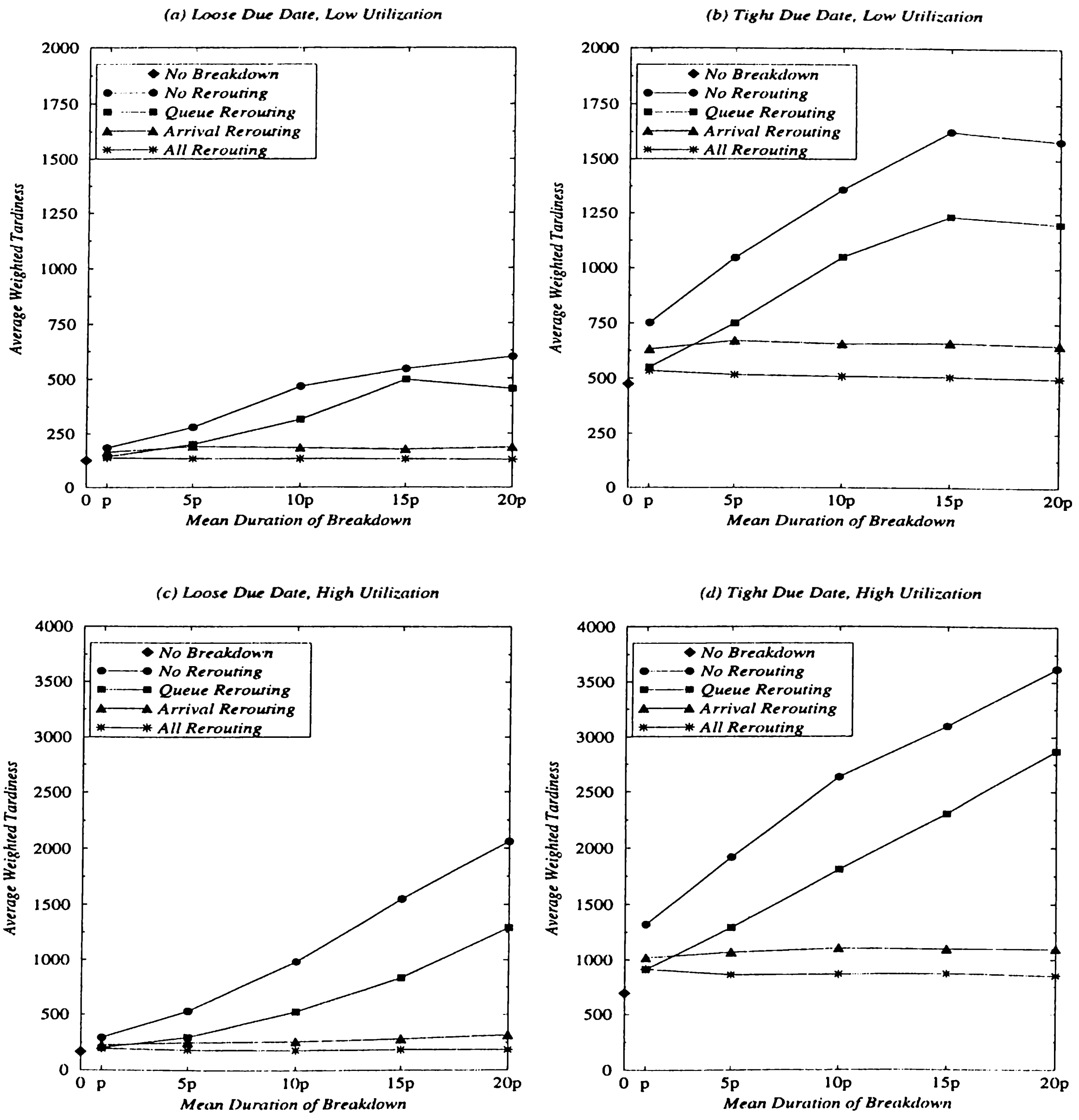

Figure 3.2: Average Weighted Tardiness versus Mean Duration of Breakdown with no Material Handling Consideration (Equal Alternative Machine, Efficiency=90%)

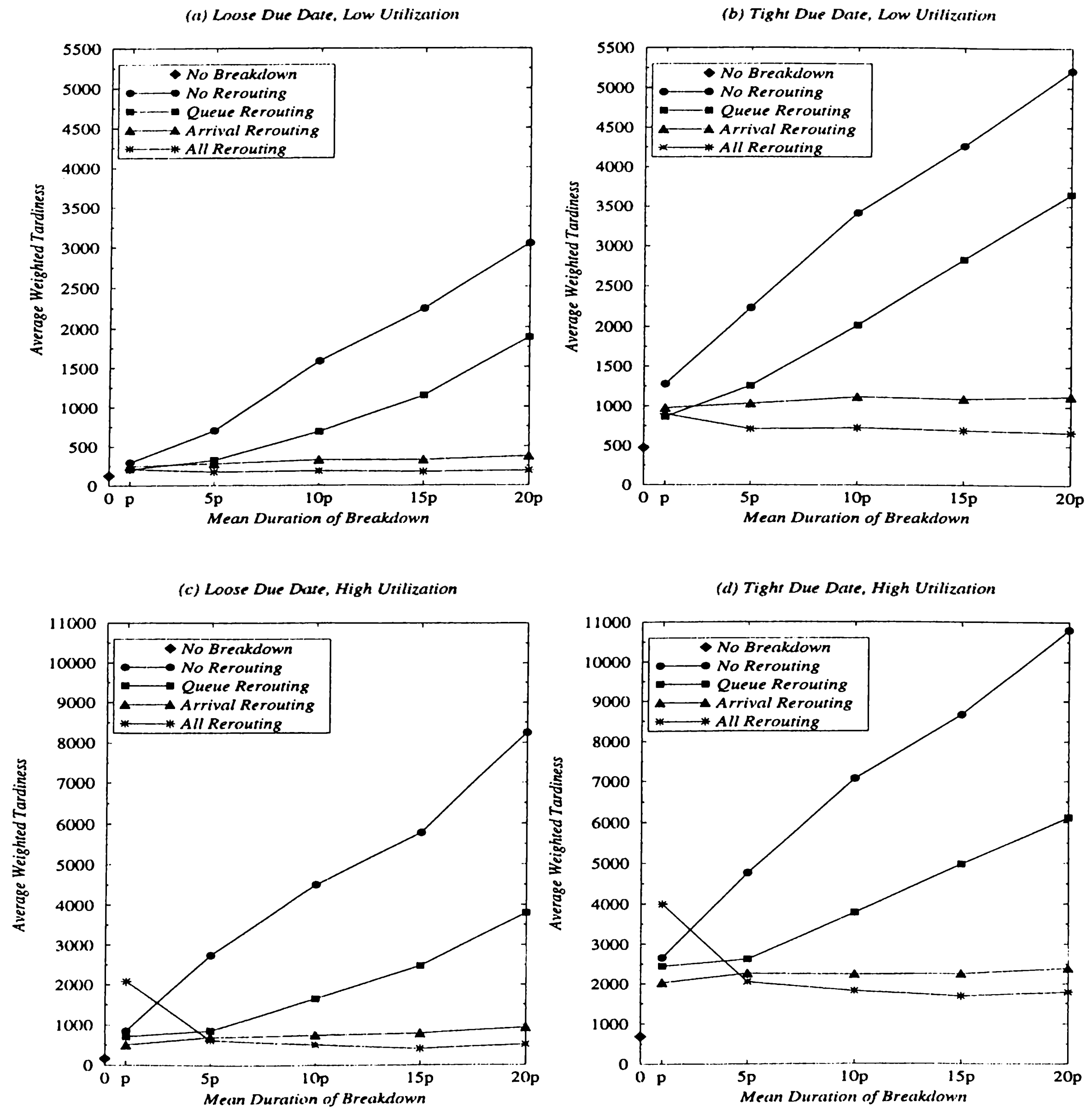

Figure 3.3: Average Weighted Tardiness versus Mean Duration of Breakdown with no Material Handling Consideration (Unequal Alternative Machine, Efficiency=80%)

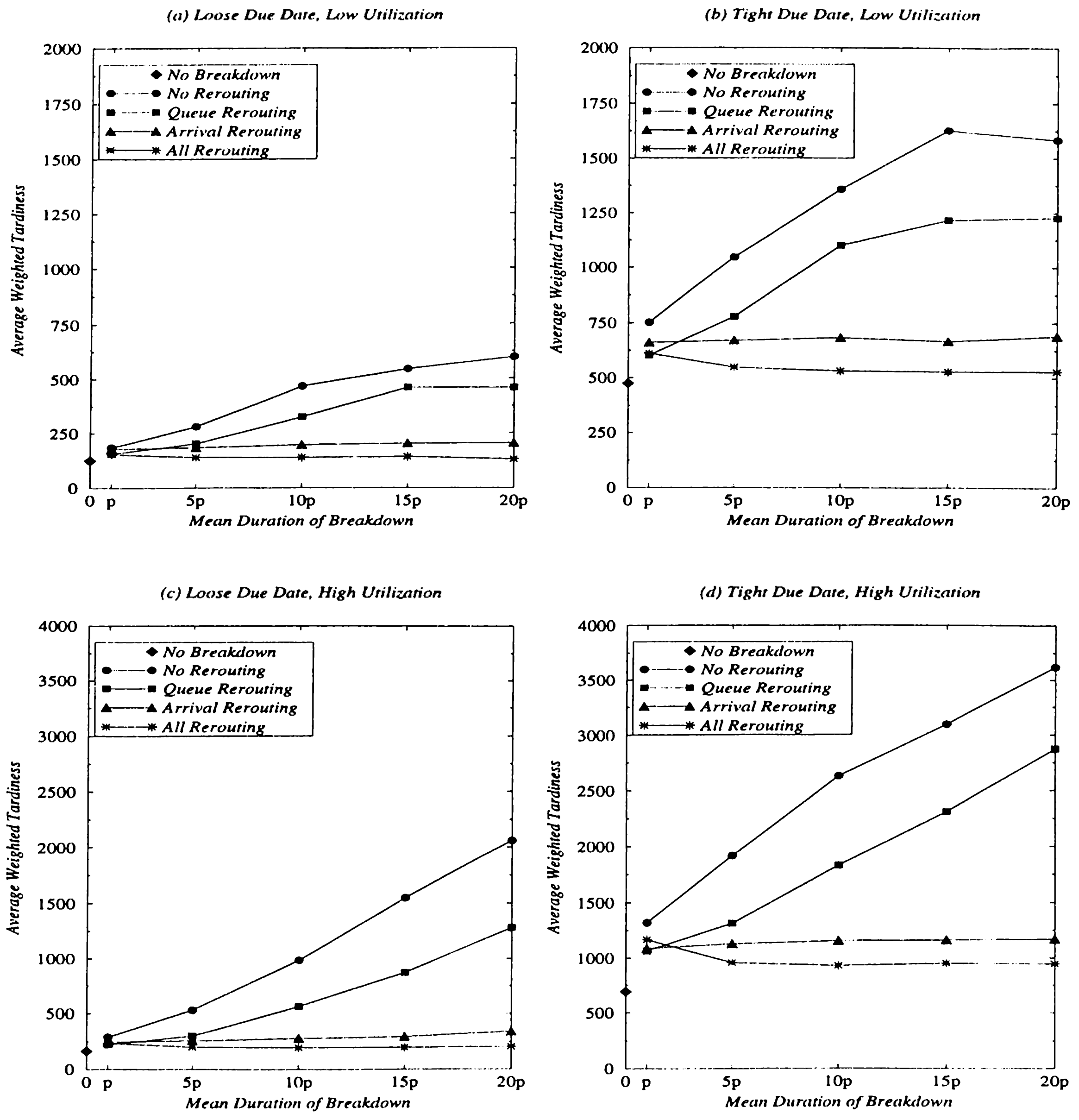

Figure 3.4: Average Weighted Tardiness versus Mean Duration of Breakdown with no Material Handling Consideration (Unequal Alternative Machine, Efficiency=90%)

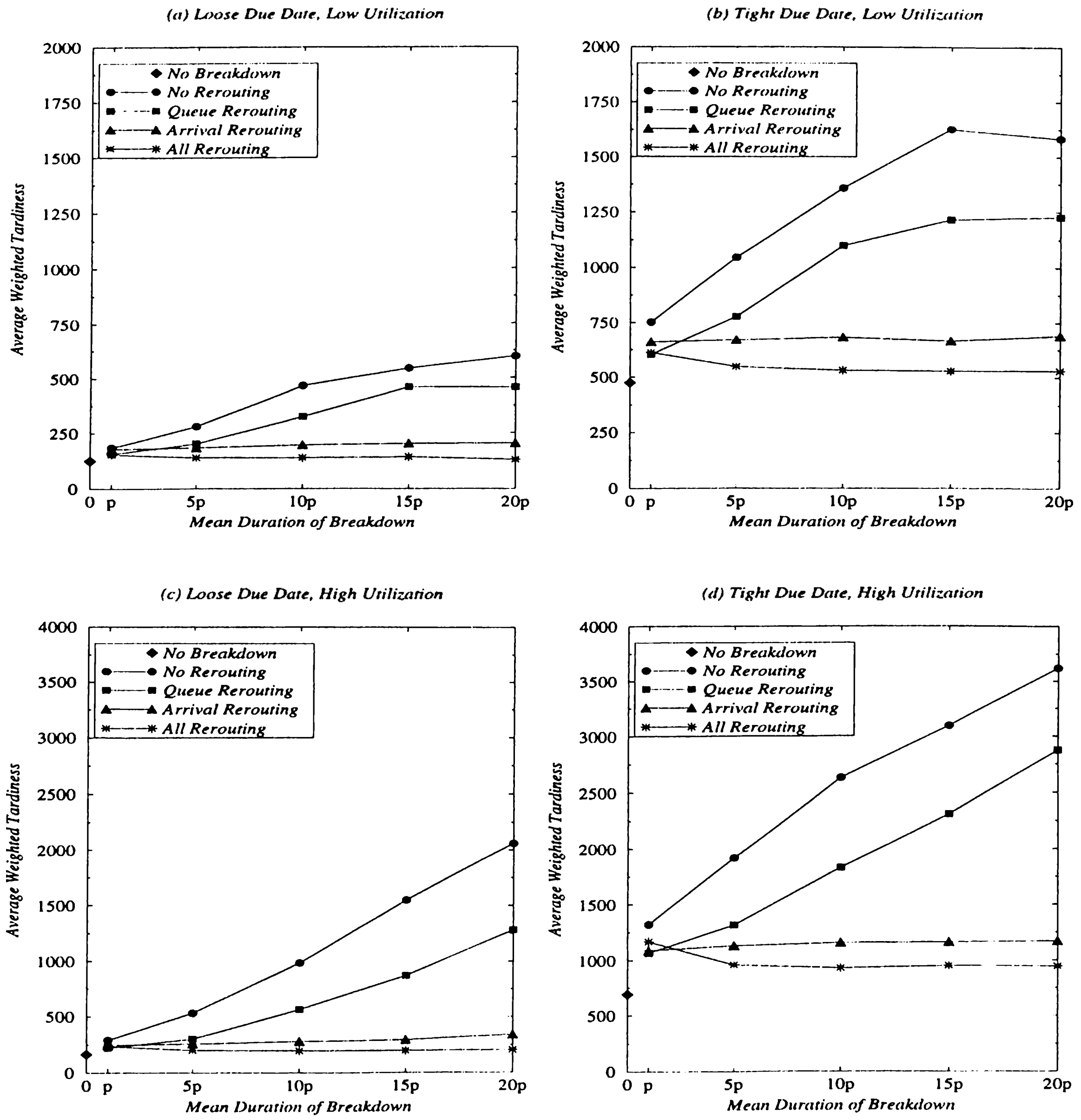

Figure 3.4: Average Weighted Tardiness versus Mean Duration of Breakdown with no Material Handling Consideration (Unequal Alternative Machine, Efficiency=90%)

Table 3.1: Average Weigted Tardiness versus Mean Duration of Breakdown
Number of MH devices=4; Transportation/Processing Time=0.30;
Tight due date; Low utilization; Equal Alt. Mach., Eff=80%

No down: 953.05	Mean Duration of Breakdown				
Policy	p_{avg}	$5p_{avg}$	$10p_{avg}$	$15p_{avg}$	$20p_{avg}$
No Rerouting	1929.84	2971.34	3988.73	4736.03	5610.80
Queue Rerouting	*	2104.56	2895.99	3547.09	4390.91
Arrival Rerouting	1550.36	1643.71	1670.29	1669.76	1705.24
All Rerouting	*	1340.63	1273.00	1240.51	1256.82

In almost every case, the AAR policy is the best. But there are some exceptions. When the alternative machine is not equal to the main machine, the AR policy yields better results if the machines break down often and they are repaired in short time (Efficiency is low, and mean duration of breakdown is p_{avg}). Moreover, the AAR policy is the worst among the policies in this case. When the overall efficiency is high (i.e. the machines are more reliable), then the same observation is made. But in this case, the AAR policy is not the worst. There is always a cross-over between p_{avg} and $5p_{avg}$ of mean duration of breakdown. After $5p$ point, all rerouting policy again draws lower envelope. These results are more significant when the utilization of the system increases.

The results also show that all rerouting and arrival rerouting policies are more robust against breakdowns. As can be seen in figures, they do not deteriorate as the mean duration of breakdown increases. However, this is not true for the NR and QR policies. They deteriorate immediately even for shorter down times. In almost all of the cases, NR is outperformed by all other policies and the QR policy is the second worst, except the case where the mean duration of breakdown is p_{avg}.

In all of the cases and in all durations of breakdown all rerouting policy is the best, except when the utilization is high and alternative machine is not equal to the main machine in 80% efficiency. In this case, when the down times are small, arrival rerouting is the best, for larger mean down times, again all rerouting outperforms other policies including arrival rerouting.

Table 3.2: Average Weigted Tardiness versus Mean Duration of Breakdown Number of MH devices=4; Transportation/Processing Time=0.30; Tight due date; High utilization; Equal Alt. Mach., Eff=80%

No down: 2608.42	Mean Duration of Breakdown				
Policy	p_{avg}	$5p_{avg}$	$10p_{avg}$	$15p_{avg}$	$20p_{avg}$
No Rerouting	4911.69	7821.44	10808.95	12383.90	15587.90
Queue Rerouting	*	*	*	*	*
Arrival Rerouting	3787.54	4126.01	4300.19	4292.15	4203.60
All Rerouting	*	*	*	6977.66	5413.43

Table 3.3: Average Weigted Tardiness versus Mean Duration of Breakdown Number of MH devices=4; Transportation/Processing Time=0.30; Tight due date; High utilization; Equal Alt. Mach., Eff=90%

No down: 2608.42	Mean Duration of Breakdown				
Policy	p_{avg}	$5p_{avg}$	$10p_{avg}$	$15p_{avg}$	$20p_{avg}$
No Rerouting	3393.46	4206.99	5190.09	5954.67	7063.31
Queue Rerouting	*	13402.28	7668.58	6720.89	7302.83
Arrival Rerouting	2955.99	3107.68	3188.59	3143.97	3104.84
All Rerouting	*	6588.54	4062.29	3620.61	3227.67

The results with MHS considerations are summarized in Tables 3.1–3.8. In these tables, the asterisk symbol denotes that the system explodes (saturates) because of the insufficiency of the MHS capacity. Again, it is observed that there is no significant effect of due date tightness on the relative performances of the policies. When the number of devices is high and/or transportation time/processing time ratio is small, the similar observations are made as in the no-material handling case, except with very small down times. In this case, MHS capacity is not sufficient for the queue rerouting and all rerouting policies. Consequently, the AR policy is the best performer. In longer mean down times with lower frequency of breakdowns, the MHS capacity becomes sufficient for queue and all rerouting, and all rerouting yields better performances.

When there is no too many devices and/or transportation time/processing time ratio is high, the MHS capacity is not sufficient for these policies even in larger mean down times. These effects are even magnified when the efficiency of the system is low, utilization is high, or alternative machine is not identical

Table 3.4: Average Weigted Tardiness versus Mean Duration of Breakdown
Number of MH devices=4; Transportation/Processing Time=0.30;
Tight due date; High utilization; Unequal Alt. Mach., Eff=80%

No down: 2608.42	Mean Duration of Breakdown				
Policy	p_{avg}	$5p_{avg}$	$10p_{avg}$	$15p_{avg}$	$20p_{avg}$
No Rerouting	4911.69	7821.44	10808.95	12383.90	15587.90
Queue Rerouting	*	*	*	*	*
Arrival Rerouting	4094.43	4626.07	4921.60	4863.87	4969.36
All Rerouting	*	*	*	*	8073.28

Table 3.5: Average Weigted Tardiness versus Mean Duration of Breakdown
Number of MH devices=4; Transportation/Processing Time=0.15;
Tight due date; High utilization; Unequal Alt. Mach., Eff=80%

No down: 943.27	Mean Duration of Breakdown				
Policy	p_{avg}	$5p_{avg}$	$10p_{avg}$	$15p_{avg}$	$20p_{avg}$
No Rerouting	3010.28	5379.26	6891.18	9113.84	11101.08
Queue Rerouting	*	3065.25	4339.05	5266.18	6610.42
Arrival Rerouting	2460.64	2646.03	2671.78	2602.70	2758.43
All Rerouting	*	2642.94	2206.58	2098.61	2187.05

to the main machine. Again, in these cases, the AR policy produces very good performances.

In all conditions, the QR and NR policies are not the solution for reactive scheduling. As opposed to this situation, the AR and AAR policies are more robust and their performances are very good as in no-material handling case.

3.6 Conclusions

In this chapter, reactive scheduling problems are discussed and a new methodology is proposed. Four reactive scheduling policies are tested under machine breakdowns in dynamic job shop environment to test the proposed methodology. The results show that if the MHS is ignored as in the previous studies, or the MH devices are fast enough so that the transportation time/processing time ratio is small, or there are excess number of MH devices in the system,

Table 3.6: Average Weigted Tardiness versus Mean Duration of Breakdown Number of MH devices=5; Transportation/Processing Time=0.30; Tight due date; High utilization; Equal Alt. Mach., Eff=80%

No down: 1260.96	Mean Duration of Breakdown				
Policy	p_{avg}	$5p_{avg}$	$10p_{avg}$	$15p_{avg}$	$20p_{avg}$
No Rerouting	3514.94	5807.86	7870.95	9501.10	11212.54
Queue Rerouting	*	4109.59	4962.48	5865.39	7348.54
Arrival Rerouting	2532.20	2731.07	2748.05	2779.43	2788.55
All Rerouting	*	2826.40	2316.75	2212.76	2237.45

Table 3.7: Average Weigted Tardiness versus Mean Duration of Breakdown Number of MH devices=5; Transportation/Processing Time=0.30; Tight due date; High utilization; Unequal Alt. Mach., Eff=80%

No down: 1260.96	Mean Duration of Breakdown				
Policy	p_{avg}	$5p_{avg}$	$10p_{avg}$	$15p_{avg}$	$20p_{avg}$
No Rerouting	3514.94	5807.86	7870.95	9501.10	11212.54
Queue Rerouting	*	4857.76	5204.75	6134.66	7544.43
Arrival Rerouting	2933.75	3157.99	3157.47	3112.74	3233.56
All Rerouting	*	6119.54	2912.61	2680.44	2723.76

then the all rerouting policy is preferred as a reactive policy. In other cases, arrival rerouting can be used. Hence, the arrival rerouting policy is suggested when the machines are broken often and repaired in a short time or the MHS capacity is low to compensate the extra requirements of all rerouting policy. No reaction is not seen as an appropriate strategy for reactive scheduling.

From these results, we see that the selected reactive scheduling strategy mostly depends on the several factors such as utilization and capacity of machines and MHS, duration and frequency of the unexpected events, etc. By considering the current system conditions and estimated impacts of the unexpected events, we can select the appropriate reactive policy. For this reason, we conclude that, the proposed reactive scheduling framework is quite well-designed. But the experimentation on an extended test bed is needed to compare the effects of all three reactive modes (including rescheduling with an off-line method that corresponds to major reactive mode) and to differentiate the event types in a more concrete manner.

Table 3.8: Average Weigted Tardiness versus Mean Duration of Breakdown
Number of MH devices=5; Transportation/Processing Time=0.35;
Tight due date; High utilization; Equal Alt. Mach., Eff=80%

No down: 1705.67	Mean Duration of Breakdown				
Policy	p_{avg}	$5p_{avg}$	$10p_{avg}$	$15p_{avg}$	$20p_{avg}$
No Rerouting	3975.59	6382.77	8670.37	10297.14	12941.39
Queue Rerouting	*	*	10174.44	8956.46	10050.74
Arrival Rerouting	2998.08	3266.50	3363.46	3288.11	3322.42
All Rerouting	*	*	3746.89	3156.61	3007.39

In the next chapter, we investigate the effects of unexpected events in a different environment where information about the some part of the future events is available for scheduling. In this case, we assume that we have perfect information about some of the events that will occur during a specified horizon, and we analyze the effects of timing of the scheduling decisions on the system performance.

Chapter 4

Scheduling in Stochastic Job Shops

4.1 Introduction

In Section 1.2, we have listed the assumptions made in general dynamic job shop scheduling studies. In this chapter, we relax some of these assumptions to achieve a better representation of reality in the simulation model and to examine the sensitivity of the results on these assumptions. For example, processing time variation is allowed in this study to represent the real-life more accurately, and to compare new findings with the results of the previous studies. By this way, it is possible to examine the sensitivity of the earlier results to some of these assumptions.

It is also assumed in the previous studies that job arrivals to the system occur randomly over time, i.e. arrival times are not known in advance. But in practice, some arriving jobs and their characteristics may be known for a certain period of time in the future as follows:

For example, a scheduling system can be considered as an intermediate level of a more global planning/scheduling system as the one given below:

- Long-range planning (e.g., plant expansion, plant layout)
- Middle-range planning (e.g., production smoothing, logistics)
- Short-range planning (e.g., requirements planning, shop bidding, order review/release, due date setting)
- Scheduling (e.g., job shop routing, sequencing)
- Reactive scheduling and control (e.g., rush jobs, machine breakdowns)

In such a global view, the major assumption is that the output of a one level is the input (i.e., constraints and environmental parameters) for the next level. If we consider the short-range planning, we see that the short-range planning module makes the material requirements planning (MRP), prepares job release data and due date information. By this way, the upper level lays out a forecast of job arrivals to the scheduling system. The data provided by the short-range planning module usually consists of arrival times, due dates and quantities of each job together with the operation and routing information. Hence, the scheduling system operates using master schedule of the upcoming jobs, their due dates, or other information for a certain period of time. The following definitions are offered for further reading:

Definition 4.1 (Job release data) *The necessary data provided by the short-range planning system that consists of the job characteristics (arrival times, due date, operation sequence, operation processing times, and the required machines) is called job release data.*

Definition 4.2 (Scheduling decision) *The decision regarding to the scheduling of the jobs (i.e. sequencing the jobs, determining the start time of an operation, selecting a scheduling rule, etc.) by considering the job release data, current system status and the objectives is called scheduling decision.*

Definition 4.3 (Scheduling decision point) *The point in time at which the scheduling decision is made is called scheduling decision point.*

Definition 4.4 (Forecasting horizon) *The time period for which we have the most accurate information about the events that will occur is called forecasting horizon.*

Definition 4.5 (Look-ahead window) *The time period that the scheduling method looks ahead into the future at a scheduling decision point is called look-ahead window.*

Definition 4.6 (Scheduling period, or Planning horizon) *The time period between two successive scheduling (or rescheduling) decision points is called scheduling period.*

As illustrated in Figure 4.1, the job release data is provided by the short-range planning module to the scheduling level. The scheduling level makes the scheduling decisions using this data set and the current shop status information collected from the physical system. The scheduling decision may be in the form of a generation of a fixed schedule (i.e. making strict decisions on the start and finish times of each operations of all jobs). It can also be in the form of selecting a scheduling rule, or finding the best parameter value of a scheduling algorithm. In our case, at these decision points, we select a priority dispatching rule for the next scheduling period by considering the current shop status and the job release data. Moreover, in this study, lead time estimation parameters of the scheduling method (ATC) are improved at these decision points.

The relations between the forecasting horizon, the look-ahead window and the scheduling period are illustrated in Figure 4.2. Typically, the forecasting horizon represents the time period which is covered by the job release data. A decision is made by using all or some part of this data set. The look-ahead window can have two extreme lengths. The length of the window can be zero in which case the scheduling decision is made using only the currently available jobs, or it can also equal to the length of the forecasting horizon in which case we use all of the available data. Hence, the forecasting horizon restricts the length of the look-ahead window. In practice, the latter method is usually used

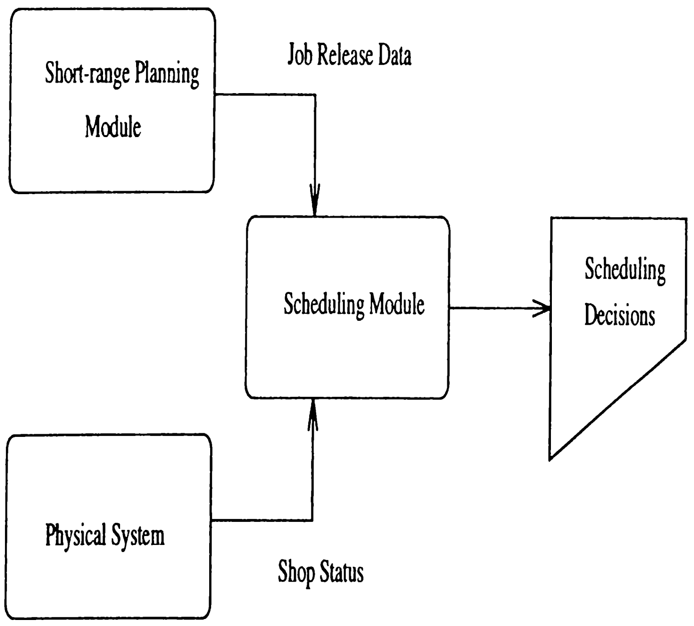

Figure 4.1: Schematic view of the interactions between scheduling levels and the other components

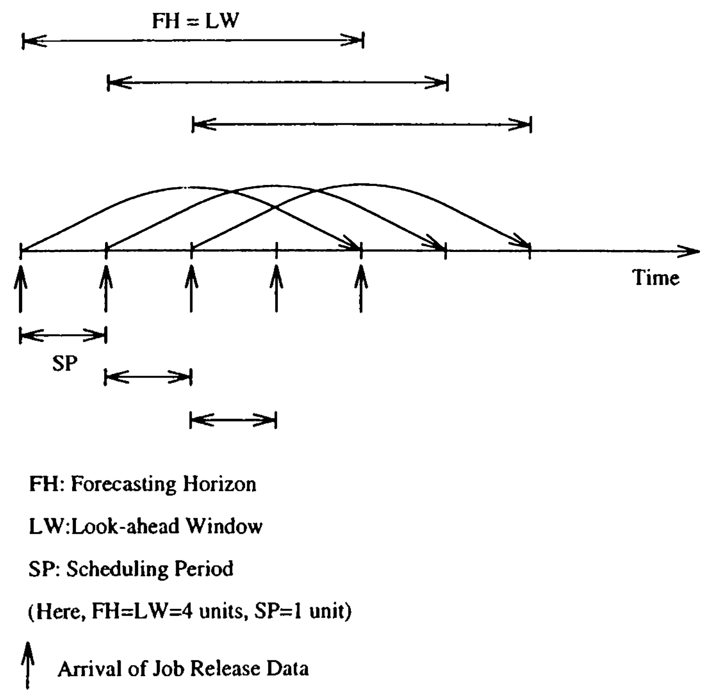

Figure 4.2: Schematic view of the relations between forecasting horizon, scheduling period, and look-ahead window

to utilize all the information and consider the effects of the current decisions on the future events. However, shorter look-ahead windows can also be used. Since the length of the look-ahead window affects the computation time required by the decision making process, longer look-ahead windows can sometimes bring untolerable computational burden. In such cases, even though we have more information, we can use some part of the data set to reduce the computational time required for a decision.

The scheduling period which is sometimes called *planning horizon* determines the frequency of the scheduling. The shorter the period, the greater the number of decisions made in a given forecasting horizon. In the literature, a decision point is usually taken to be the arrival time of the new job release data. In this case, the interarrival time of the job release data determines the length of the scheduling period. If the the arrival of the job release data from the short-range planning module is periodic, then the scheduling period has a specified constant length. In other words, the operation of the decision making process is synchronized by the arrival of job release data. This is logical, since if we do not receive any new information and the events occur in system as expected, then we do not need to revise our previous decisions. If we use shorter scheduling periods, we increase the frequency of the revision of the scheduling decisions. This increases the system nervousness, but we follow the real-life system status more closely and up-to-date.

For instance, if a 4-week job release data is provided by the short-range planning module every week, then we can make scheduling decisions for the next 4 weeks and implement only the decisions of the first week. At this point (at the end of the first week), we obtain the new job release data (fresh data) and make a new scheduling decision for the next 4 weeks in a rolling horizon scheme.

Note that, at all decision points, we assume that the job release data is perfect (i.e., jobs will arrive at times specified in the job release data, the jobs will have prespecified characteristics, and all machines are continuously available). However, in practice, many unexpected events or unforeseen interruptions can occur that our previous decisions may need to be revised again. These events can be listed as follows:

- Processing time variations
- Machine breakdowns
- Arrival of new jobs
- Job rework and recycle
- Job scrapped and replacement
- Job due date changes

To some extent, these unexpected events are inevitable in real-life systems. These events, once they occur, do not only affect the system performance, but also upset the scheduling decisions generated previously (the scheduling decisions may even become infeasible to be implemented). At this point, what is needed is reactive scheduling and control. This control system can take corrective actions to revise the previously generated scheduling decisions, or can invoke a new decision making process to generate new set of decisions by using the new shop status information.

In this study, we consider the first two of these events: procesing time variation and machine breakdowns. In general, scheduling decisions are usually made by using best estimates of the processing times. But the processing times differ from their estimated values. By allowing processing time variations in our study, we relax the assumption of deterministic processing times (Assumption number 4 in Section 1.2).

The second event is the machine breakdown. Again, we usually make scheduling decisions by assuming that the machines are continuously available. But in practice, the machines are all fallible. The consideration of machine breakdowns corresponds to the relaxation of the assumption that the machines never fail for processing (Assumption number 9 in Section 1.2).

In this study, we also analyze the effects of the look-ahead window and scheduling period on the system performance. Specifically, we investigate whether the system performance can be improved by properly selecting the values of these paramters. Our initial expectation is that the longer look-ahead windows with shorter scheduling periods produces better performances.

To study these issues, we propose an iterative simulation-based scheduling system which utilizes the discrete-event simulation as a real-time decision making tool.

In the next section we review the relevant literature. In this review, the emphasis will be on simulation-based studies. In Section 4.3 the proposed scheduling and control approach will be described in detail. This is followed by the the experimental design and system considerations in Section 4.4. The computational results are presented in Section 4.5. Finally, this chapter will end with the summary and concluding remarks in Section 4.6.

4.2 Literature Review

Yamamoto [65] claims that proposed scheduling methods have not been very useful in solving the real-life scheduling problems even though considerable research effort has been conducted in regard to job shop scheduling problems. The author lists three reasons for this inability:

- Since the scheduling problems are in combinatorial nature, computational time required to solve the practical problems is too long. This makes the proposed methods unpractical for real-time applications.
- The proposed models have too many assumptions in contrast to the need for considerations on the variety of system characteristics.
- In the real shop conditions, unexpected events and variations can easily invalidate the current schedule. Hence, a significant differences can be seen between a schedule and the real progress of the jobs in the shop.

Under these observations, the following remarks are made: First, when a scheduling decision is needed, it must be made as soon as possible in real-time, there may not be a plenty of time to solve the problem optimally. A solution approach that operates fast is needed in practice. The model should also represent important characteristics of the systems. Second, the system has to deal with unexpected events as they occur over time. In case of unexpected events, the traditional approach is to reschedule all the jobs on-hand by considering the new state of the system. This approach, in a way, decomposes the real-life dynamic and stochastic scheduling problem into a series of static and deterministic scheduling problems. If we want to correct the differences between the planned schedule and the real progress, and maintain the control by the schedule, we cannot avoid continuous rescheduling. Such a practice is usually undesirable from the standpoint of the shop management. In this case, several researchers proposed scheduling/rescheduling approaches in a more controlled manner.

In the next section, we first discuss some conceptual studies on the simulation-based scheduling and control. Then we classify the approaches proposed in the literature according to their considerations of stochastic events. The studies that do not consider the unexpected events other than dynamic job arrivals will be called "*deterministic studies*" and the studies that explicitly investigate the unexpected events will be called "*stochastic studies*". Also, we separate the studies those that use look-ahead methods by assuming the existence of a forecasting horizon ("*with look-ahead*") and those that consider only jobs currently on-hand ("*without look-ahead*"). We present review for each class separately.

4.2.1 Conceptual Studies on Simulation Approaches

As the investigators confront with the problems that are outlined in the previous section, they have proposed alternative approaches to overcome these issues. One of these approaches that has been proposed in the literature is simulation. Hence, there is a growing area of scheduling that uses the simulation as a real-time scheduling and control tool. The one of the first conceptual studies is by Davis and Jones [15]. The authors propose a framework for addressing real-time scheduling problems in manufacturing environments using discrete-event simulation and mathematical decomposition to break down production scheduling problems into a hierarchical decision structure. A production planner provides input for an inter-process coordinator (IPC), which then directs the individual process controllers (PCs). While PCs contain more detailed information regarding process control, the IPC has more aggregate system information. In this case, a mathematical programming method is not feasible, due to the complex constraints and stochastic nature of the process. Furthermore, there are conflicting objectives. Therefore, the authors propose a simulation mechanism that is responsible for the simulation of each scheduling alternative (priority dispatch rules, routing alternatives, etc). The simulation is integrated with the shop floor information in order to obtain the current system status at the time of execution. Development of a production schedule involves the simulation of each rule. Compromise analysis is then performed by comparing the simulation results to select the best rule for the implementation.

In another study, Harmonosky [19] discusses the implementation issues for using simulation for real-time scheduling, control and monitoring. In this study, discrete event simulation is proposed as a real-time decision making tool. In the proposed system, there are two operational modes: (1) monitoring mode when the model is directly linked to the real system and in this mode, the model accurately represents the physical system; (2) decision-making mode when the model evaluates different control decision options. In this mode, simulation is used as in its traditional role. This study also discusses issues modeling, interfacing to the physical system, saving the system status for evaluating alternatives and recovery at decision points.

Harmonosky and Robohn [21] reviews the recent research on real-time scheduling and control of computer integrated manufacturing systems in the areas of operations research, artificial intelligence techniques and simulation methods. They discuss several implementation issues and make the following conclusions:

- The amount of interest in real-time scheduling and control systems for automated manufacturing environments strongly indicates a perceived need for these systems in industry.
- Real-time scheduling and control systems using simulation have great potential for application in the very near future.
- There are several problems in using simulation as a real-time decision making tool. One of them is the length of the simulation window (i.e. look-ahead window). The other issue is the frequency of the calling the simulation mechanism (frequency of simulation experiments to be conducted). Dealing with the simulation output, data acquisition and interface problems are the other matters that need to be studied.

In her later work, Harmonosky [20] analyses two key issues for using simulation as a real-time production control: (1) the simulation run length, and (2) the type of look-ahead horizon (deterministic versus stochastic). The former determines the response time (or execution time) of the simulation model. This concept is important for any real-time decision making tool since the amount of time it takes to make decisions will directly affect the degree to which the system is controllable in real-time. The latter regards the assumption made on the look-ahead horizon. Either a deterministic look-ahead window could be used, where no further system disruptions are considered, or a stochastic window could be used, which includes further system disruptions during look-ahead. The experiments focusing on these issues show that lower execution times are associated with high average processing time, work-in-process level as a performance measure, and a system that is closer to a flow shop than a job shop. Hence, the simulation can be suggested as a real-time control tool for systems which have these characteristics. While comparing the deterministic and stochastic simulation, machine breakdown event is used as a disruption. The experimental results indicate that deterministic simulation is preferred as the execution or response time is considered. But, when the final performance measure is measured, the stochastic simulation yields better results. For example, one case having higher arrival rate, longer repair times and more frequent interruptions appears to be a clear candidate for using a stochastic look-ahead.

4.2.2 Deterministic Studies with Look-ahead

According to our classification, deterministic studies with look-ahead are those that do not consider unexpected events except dynamic job arrivals and that assume that there is perfect information about the events that will occur in the next forecasting horizon. The most of the studies in this class uses simulation as a look-ahead tool. There are also studies using some optimization approaches for the decision making process.

Maimon [36] proposes a real-time operational control system of three levels of control: a scheduler level, a communication level, and a process sequencer level. The scheduler determines the instantaneous production rate for each part in the system in order to best utilize the system capacity. The communication level conveys the instructions to the controllers and transmits feedback to the scheduler. The process sequencer makes the actual part sequencing decisions and consists of a knowledge base, the system status, and production state. The process sequencer selects a specific set of actions based on the facts known about the system from the current shop floor status and the knowledge base. A case study of a flexible manufacturing cell is simulated.

Vepsalainen and Morton [60] propose a iterative approach for dynamic scheduling problem which is called lead time iteration (LTI). The proposed approach uses simulation as a scheduling generation tool. The approach is to repeat the simulation runs iteratively so as to find lead time estimates more consistent with the specific problem and load. At a rescheduling point defined by constant scheduling period length, a series of simulations is run for the jobs to be released in the next forecasting horizon. Beginning with an initial waiting time estimate for each operation, a simulation is run by using the ATC or COVERT priority rule which uses waiting time estimates. After each simulation, the waiting time estimates are updated using the realized waiting times. The iteration continues until a given stopping criterion is satisfied. Until the next scheduling period starts, the best lead time estimates are used during the implementation. The method is compared with the WSPT, S/RPT, FCFS and EDD as well as standard ATC and COVERT which is based on that the waiting time estimates are some multiple of the processing times. The experiments with fixed scheduling period and a longer forecasting horizon show that the performance of the single-pass ATC and COVERT can be significantly improved by using LTI.

Wu and Wysk [62], [63] propose a multi-pass scheduling algorithm that utilize simulation to make better scheduling decisions for an FMS. They assume that the factory control system (which corresponds to our short-range planning module) provide the perfect information about the events of the next scheduling period and objectives for which the alternatives are compared. In their study, alternatives are the priority dispatching rules that can be applied in the scheduling of jobs. The multi-pass scheduling system simulates each rule by using the current shop status information. By this way, one simulation run is conducted for a short time period, called "*simulation window*". The rule which yields the best performance measure is then selected and implemented during this period. At the end of the period, the procedure is repeated (at the end of the period, in fact, the state is the same as the ending condition of the one simulation run that is simulated with the selected rule, because there is no variation between the simulation and the real progress). The main idea behind this application is that combining different dispatching rules in a dynamic and multi-pass manner creates a better result than applying a single rule alone for the entire horizon in a static manner. The experimental results show that if the scheduling period (or simulation window) is accurately determined according to the environmental conditions and objectives then this logic is very useful. Therefore, they indicate that the length of the scheduling interval is a significant factor for the performance of multi-pass scheduling algorithm. According to Wu and Wysk [63], if the window is too short, the statistics collected will not give a reasonable measure of the system performance. But if the window is too long, the simulated system performance may be less sensitive to switching between dispatching rules at right time and the scheduling mechanism will only provide average and aggregate performance measures in each period, which may loose the advantages of the multi-pass scheduling. Wu and Wysk [62] combine the simulation mechanism with a knowledge based system. By this way, a manufacturing control system is developed that learns from its historical performance and makes own scheduling and control decisions by simulating alternating combinations of priority dispatching rules. In this case, the rules that will be evaluated are selected from a larger set of rules by the factory control system by considering the system conditions using knowledge base.

Ishii and Talavage [24] proposes another multi-pass scheduling algorithm for flexible manufacturing systems. They propose variable-length scheduling intervals and simulation windows, by observing the drawbacks of the study conducted by Wu and Wysk [63]. First, in their algorithm, the simulation window length is different from the length of the scheduling interval. In Wu and Wysk [63], the length of the simulation window is used to determine how long the prediction mechanism should look ahead, or how frequently the mechanism should be used to evaluate the various dispatching rules and it is constant (it is prespecified). But, as Ishii and Talavage state a constant length scheduling period cannot follow system state changes, hence it cannot always make performance better particularly in a dynamic system. Also, in constant scheduling period, there is a censored data problem. According to Ishii and Talavage, a number of parts remaining at the end of each simulation run affect different dispatching rules in an unequal way. The evaluation of dispatching rule based on censored data could cause a selection of an inadequate dispatching rule. This problem becomes more important when the scheduling interval is short as in this case. Hence, Ishii and Talavage define the length of the next scheduling interval according to the system transient state. This period is determined by calculating an index value representing the system congestion. As soon as the index is minimum in a representative simulation run with FCFS rule, this point determines the next scheduling interval. Since the value of the index totally dependent on the system state, the scheduling interval is not a constant, in fact, it is a random variable. By this way, they aim to create independent scheduling intervals as much as possible. For the simulation window, there are four alternatives tested: In the first one, as in Wu and Wysk [63], the simulation window is equal to the scheduling interval, but not a constant. In the other three methods the simulation window is changing from 2 times of the scheduling interval to the entire manufacturing horizon. By taking the simulation window longer as compared with the scheduling interval, it is aimed to avoid censored data problem. The results support these conjectures. The experiments show that using a scheduling interval defined based on the system transient state makes the performance of the multi-pass scheduling algorithm better than using a constant scheduling interval which has a very unstable performance as compared with the single-pass rules. Also if the scheduling intervals are not accurately determined, the performance of a multi-pass scheduling algorithm might be poorer than the single-pass algorithms. But if the intervals are well determined then switching the rules is better. When the simulation window is considered, the algorithm with the simulation window as equal to the entire manufacturing horizon performs better than the other strategies.

In their later work, Ishii and Talavage [25] concentrate on using the different rule on each machine in each period after observing the advantages of using different rule in each short time period. In this study, a mixed dispatching rule which can assign a different dispatching rule for each machine is proposed. A search algorithm which selects an appropriate mixed dispatching rule using predictions based on discrete event simulation beginning from the bottleneck machine is developed. The experiments conducted on a representative flexible manufacturing system show that the mixed dispatching rule performs better than the conventional single-pass single-rule approach.

Cho and Wysk [12] propose a intelligent workstation controller which is a part of a shop floor control system. In this study, the controller receives the information such as part type and quantity, part routing specifications, and process plans from the shop level controller and coordinates production activities. For this purpose they develop a neural network model that generates alternative part dispatching rules based on the current system status. When the alternative rules are determined, the multi-pass simulator evaluates each rule by running a simulation model of the real-life system by using the provided information for a pre-specified short time horizon (*simulation window*). The experiments show that the simulation window must be determined according to the performance measure. The proposed approach is also compared with the single-pass rules and it is found that the approach outperforms all the single-pass rules based on all the performance criteria.

Ovacik and Uzsoy [45] present several rolling horizon procedures to minimize maximum lateness on a single machine in the presence of sequence-dependent setup times. At any point in time when a scheduling decision is to be made, they solve a subproblem consisting of the jobs on hand and a subset of the jobs that will arrive in the near future. In this study, arrival times or ready times of the jobs that will be available in the next short time period called *forecast window* are assumed to be known a priori. These restricted-size subproblems are solved optimally by using branch-and-bound algorithm. A prespecified part of the schedule is applied for some time known as *planning horizon* until to the next decision point. By this way, the solution to the overall problem is approximated by segments of the solutions of these subproblems. The experimental study shows that if the forecast window and planning horizon parameters are appropriately selected the proposed rolling horizon procedures outperform the best available myopic dispatching heuristic (EDD) by an order of magnitude, and yield solutions that are on average 60% better than a dispatching rule combined with local search (EDD with pairwise interchange method). The parameters determining the forecast window and planning horizon demonstrate the tradeoff between solution time and quality explicitly. As the length of the forecast window becomes longer and the planning horizon length decreases, the rolling horizon procedures yield very impressive results at the expense of solution CPU time.

4.2.3 Deterministic Studies without Look-ahead

In this class, we review the studies without look-ahead where the scheduler has no information about the future, or it only considers scheduling of currently on-hand jobs. Again, these studies focus on the deterministic case where there is no unexpected event that will occur in the system. Although there are again some studies using discrete-event simulation as a decision making tool, the most of the studies concentrates on the algorithmic approaches that generate off-line schedules for current jobs at decision points.

As a next stage of the study conducted by Vepsalainen and Morton [60], Morton et al. [38] propose an iterative simulation approach to derive the resource prices along with the lead time estimates. In this study, while the lead time estimates are improved by repeated simulation, the resource prices are also explicitly calculated by analyzing the actual busy periods from the simulation outputs. The resource prices are first used in this study to maximize the NPV of the revenues and the costs. The static single machine and multi-machine experiments show the superiority of the proposed approach against the single-pass scheduling rules.

Lawrence and Morton [35] extend the LTI concept to the resource constrained multi-project scheduling with tardiness costs. The static project scheduling problems show that significant improvement on the weighted tardiness measure can be achieved by using LTI.

Kiran, Alptekin and Kaplan [33] proposes another type of multi-pass scheduling algorithm which is called Feedback Heuristic. In this study, current jobs on hand are considered for scheduling for the next period. A static job-based schedule is generated by using a priority dispatching rule. By using these priorities, a simulation iteration is conducted. By smoothing the job priority in the previous iteration and job's contribution to the performance measure in the same iteration, the priority for the next iteration is calculated. The next iteration uses the new priorities. From one pass to the other, the performance measure of the system is improved. The experiments conducted in static and dynamic flexible manufacturing environments show that iterative simulation mechanism may be very useful especially as compared with the single-pass rules and multi-pass COVERT rule. The difference between the earlier simulation studies and the study of Kiran et al. [33] is that they assume that either all the jobs are available at time zero (static shop), or jobs arrive in batches periodically and the scheduling algorithm is executed at batch arrival times. In this case, only the jobs on hand are considered for the scheduling (i.e., there is no look-ahead for the future).

In another study, Church and Uzsoy [14] analyze the periodic and event-driven rescheduling policies for dynamic single machine and parallel machine shops with maximum lateness as a performance measure. They call the scheduling procedure in which the scheduling decisions are made at the specified points in time which are usually equally-distant in time or observation as periodic scheduling. They define continuous scheduling as the scheduling procedure in which the scheduling decisions are always renewed from scratch at each occurrence of an unexpected event. Event-driven scheduling is defined as the scheduling procedure which includes periodic scheduling, but additionally classifies the unexpected events those that need rescheduling and those that can be ignored for some time without changing the current schedule.

In this study, they propose a event-driven rescheduling approach which classifies the events as those requiring immediate action (exceptions) and those that can be ignored for some time without significantly affecting the system performance. By this way, besides the regular rescheduling points defined by the periodic scheduling policy, the rescheduling points initiated by exception events are also added. From this point of view, the event-driven scheduling has properties of both continuous and periodic scheduling. In these approaches there is no look-ahead for future time; on-hand jobs are only considered for the scheduling. The experimental results show that periodic scheduling is very useful when the jobs arrive in batches periodically to the system (or they are released to the shop periodically without any cost), and the period of the scheduling coincides with the batch inter-arrival time. In the case of dynamic and continuous arrivals, an event driven scheduling procedure is designed which initiates scheduling action when a job that has a lower slack than a prespecified limit. Experimental results show that the benefit of extra scheduling diminishes rapidly, demonstrating that a well designed event-driven scheduling can produce excellent performance with substantially lower computational burden and lower system instability than a continuous rescheduling.

A similar approach is proposed by Raman and Talbot [49] for the tardiness problem. In this approach, the approach decomposes the dynamic problem into series of static problems. These static problems are solved by focusing on the bottleneck machine that is determined according to the load of the machines. Then a schedule is constructed for the entire system by iteratively observing the tardiness of the jobs and updating their operation due dates, and it is implemented dynamically on a rolling horizon basis. The schedule is implemented until the next job arrives when the process of generating and solving the static problem is repeated for the current jobs. The experiments show that the proposed approach is significantly better than the priority dispatching rules (The approach corresponds to the continuous rescheduling in the terminology proposed by Church and Uzsoy [14]).

In a recent study of dynamic job shop scheduling, Sun and Lin [57] propose a *backward scheduling approach* in which dynamic scheduling is carried out through solving a series of static backward scheduling problems. In this approach, the dynamic job shop scheduling is modeled as a discrete event control problem that is dealt with by sequentially implementing backward scheduling over a rolling time window. At each prespecified rescheduling point, a new schedule is generated by using the current status of the shop, and the emphasis is on the due date performance and the inventory cost. The experiments show that the performance of backward scheduling approach is better than the traditional forward scheduling. However, in this study, there is not any explicit discussion on how the rolling horizon device is used, especially on determination of rescheduling points.

4.2.4 Stochastic Studies without Look-ahead

Stochastic studies consider the unexpected events and interruptions explicitly in their methods. When we say that without look-ahead, we mean that the studies consider only current jobs in the system. By this way, they make their decisions by focusing current shop status. However, during the implementation of these decisions, unexpected events occur in the system. The literature in this class is reviewed in detail in Chapter 3. Therefore, it will not be repeated here.

4.2.5 Stochastic Studies with Look-ahead

Although, some of the approaches discussed in the previous sections can be applied in stochastic environments with some modifications, as we can see from the literature, there is not any effort except the one reviewed below which considers the unexpected events explicitly and uses look-ahead methods.

Tayanithi et al. [58] propose an integrated scheduling and control system that combines discrete-event simulation and knowledge base concepts to perform on-line analysis of interruptions in an flexible manufacturing system. The machine breakdowns and rush orders are explicitly analyzed in the model. When an interruption occurs in the system, the knowledge base controller performs an analysis to determine whether the interruption is significant or not. If it is not significant, the monitoring mode continues. Otherwise, some of control policies need to be evaluated. The design of knowledge base allows the supervisor to quickly evaluate several control policies, and, consequently, achieve effective control of the system during an interruption. When a decision cannot be obtained readily from the knowledge base, the alternative actions are feeded into the on-line simulation mechanism. The purpose of the on-line simulation is to evaluate the alternative control policies. This module consists of a simulation model of the system and is invoked by either the knowledge base controller or the supervisor. The simulation model uses the freshest data about the system and run each alternative action for a short horizon. Finally, the best alternative is selected based on the simulation results. There is a restricted case study involving this approach under machine breakdown, but it only shows the feasibility of the combined knowledge base and simulation approach.

In summary, the look-ahead property with some stochastic events have not been studied adequately in the literature. Hence, our study will be the first comprehensive study which investigates the effectiveness of the look-ahead property under stochastic events. In the next section, we describe our approach in detail.

4.3 Iterative Simulation-Based Scheduling

We consider the dynamic job shop scheduling problem in which some information about the future job arrivals is known with certainty for a certain period of time. For example, the "*plant controller*" creates a job release data while making the short-range planning. The job release data consists of

- job arrival times that will occur during the next forecasting horizon;
- job characteristics such as due dates, job weights, number of operations, best estimates of processing times, routing information (machines that will be visited by the job).

This is the input for the "scheduling level".

Our approach to handle this type of the scheduling problem is to use the discrete-event simulation as a real-time decision making tool. As shown in Figure 4.3, the plant controller lays out the job release data to the *Parameter Selector (PS)* and to the *Iterative Simulation Mechanism* (ISM). The plant controller also sends the objective(s) which the management wants to minimize or maximize. Also the current shop status is fed back to the ISM and Paramater

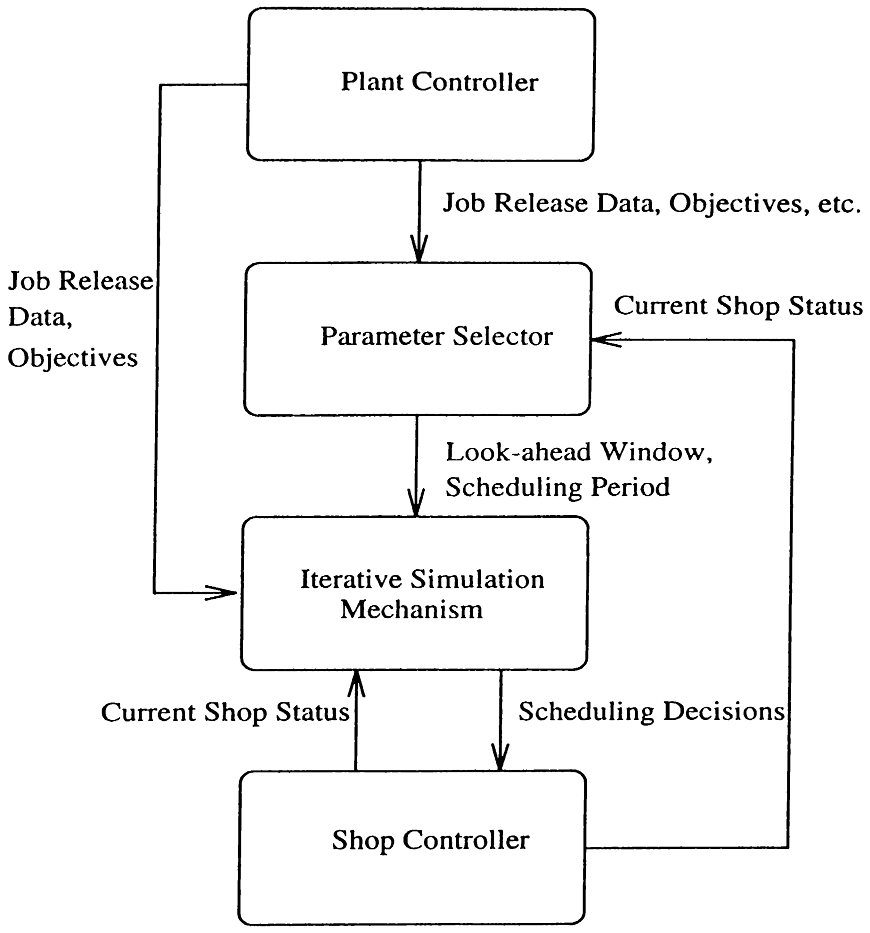
Plant Controller
Job Release Data, Objectives, etc.
Job Release Data, Objectives
Parameter Selector
Current Shop Status
Look-ahead Window, Scheduling Period
Iterative Simulation Mechanism
Current Shop Status
Scheduling Decisions
Shop Controller

Figure 4.3: Schematic view of the simulation-based iterative scheduling system

Selector by the "*shop controller*". Parameter Selector determines the look-ahead window (LW), scheduling period or planning horizon (SP), and other values of parameters that are used in the algorithm which the ISM uses during its decision making mode. This is done by examining the current shop status, job release data and objectives. ISM obtains the job release data and objectives from plant controller, current shop status from shop controller, and parameter values from PS, and activates the iterative simulation based scheduling algorithm by using these information. ISM initializes each iteration (or simulation run) with the same state (the current system status), and same dynamic events are generated by using the job release data in each iteration. The objective(s) set by the management serve as a performance measure in the simulation runs. The simulation run length (called as "simulation window") is determined by the look-ahead window parameter provided by PS. The scheduling period defines the frequency of normal ISM invokes (regular scheduling interval). According to the scheduling and control policy the ISM can be activated by the events that are unexpected at the beginning of the scheduling period such as machine breakdowns. In this case, besides the regular decision points defined by the scheduling period value set by the PS, there are *exceptional decision points* where the ISM is invoked to react to the unexpected changes. Hence, the time between two decision points will be equal to the first value set by the PS if there is no unexpected event during the real-life implementation of the decisions. Otherwise, it will be shorter than the value set by the PS before. When an exceptional scheduling is to be made, the simulation window (or look-ahead window) will be at most equal to the remaining time until the end of the current forecasting horizon, because there is no additional information since the last information arrival time.

In our current ISM implementation, we use two algorithms:

- Multi-pass Rule Selection
- Lead Time Iteration with ATC or BD

We describe these algorithms in detail in the following paragraphs:

4.3.1 Multi-pass Rule Selection Algorithm

Many researchers have studied the priority dispatching rules for more than three decades. There are a few hundreds of rules proposed in the literature. The major conclusion that can be drawn from these previous studies is that there is no single rule that yields the best performance in every environmental condition. Different scheduling rules perform better than the others under different operating conditions. This is also true when the objectives to be optimized is considered. All these results indicate that the performance of a rule is highly affected by the operating conditions of the system and the objective(s). Although a single dispatching rule will not perform the best for a long time period, some rules perform better than others under certain conditions. Therefore, changing a dispatching rule over successive short time periods based on the current system state, current performance measure, and the current information for the future expected events can make the performance better than using a single rule for a long time. As we know from the dynamic system simulation, even when the system is in steady state, there are changing conditions (high or low congestion levels, long or short queue lengths, loose or tight due dates, etc). This situation occurs more frequently in transient, or highly dynamic and stochastic systems. For this reason, switching the rules according to the current system state and current information might provide better results.

In the proposed approach, there is a set of rules which contains the candidate priority dispatching rules that can be applied in the shop for a given performance measure. The set of candidate rules can be determined by analyzing the historical information on the performances of the rules. At each decision point, a new series of simulation runs is performed by using one of the candidate priority dispatching rules in each iteration. At the end of each iteration, we record the performance measure yielded by applying the rule for this iteration. At the end of the iterations, the rule that produces the best performance measure for the simulation window is selected. This rule is applied in the real-life system until the next decision point which is defined by the new information arrival or occurrence of an unexpected event. Hence, in the proposed approach, time required to make a scheduling decision depends on the number of the rules and on the length of the look-ahead window.

In our experiments, the following rules are considered: ATC, BD, COVERT, MOD, and WSPT. These rules are used due to their better performances observed in Chapter 2.

4.3.2 Lead Time Iteration

As we see in Chapter 2, the performance of ATC and BD priority rules are better than the other rules. Note that both ATC and BD use waiting time estimates in calculation of the priority of a job. Previous experimental studies indicate that the performance of these rules is highly dependent on the waiting time estimation accuracy. Hence, if the waiting time estimation is made better, the priorities are determined more accurately and better performance values can result in. Although there are several methods to estimate the waiting times of a job for an operation, the traditional approach is to use a multiple times the operation processing time as a waiting time (This method is discussed in Chapter 2). To improve the performance of the BD rules, there are two possible ways among others to accurately estimate the waiting times by means of iterative simulation:

- Several alternative values for lead time constant is evaluated in each iteration, and the one which produces the best performance measure selected and used during the actual implementation of the rule.
- Lead time iteration (LTI) estimates waiting time of each individual operation iteratively. Waiting time estimates which produce the best performance are used during the implementation of the rule.

In this study, we focus on the second approach. The LTI starts with the initial waiting time estimates as a multiple times of the operation processing times, and smoothes the estimated and the actual waiting time estimations at the end of each iteration for the next iteration. The steps of the algorithm are as follows [39]:

- Step 1. Set iteration number n at 1. Make an initial estimate for waiting times (for example, three times of the processing times, $W_{ij}(1) = 3 \times p_{ij}$) for each job on hand and job that will arrive during the simulation window.
- Step 2. Perform a simulation with ATC or BD.
- Step 3. Record the performance measure obtained, and the actual waiting times ($Q_{ij}(n)$) for the iteration n.
- Step 4. If the termination condition is satisfied, go to Step 7.
- Step 5. Make the new estimates by smoothing the actual waiting times last recorded and the last estimates:

  $$W_{ij}(n+1) = \alpha W_{ij}(n) + (1-\alpha)Q_{ij}(n)$$
- Step 6. Go to step 2.
- Step 7. Report the performance measure and scheduling policy for the iteration that gives the best value of the objective. Use the waiting time estimates from this iteration in the actual implementation of the rule.

By smoothing process, it is aimed to prevent the waiting time estimates to change too fast. In this way, we want to close the waiting time estimations to the actual waiting times from one iteration to the next. A typical termination rule is to set a value for the maximum number of iterations. An alternative to this rule might be that "Terminate when there is no improvement in the performance for the last prespecified number of iterations".

In our current implementation of ISM, we apply the LTI method to improve the ATC performance. In the pilot experiments, we see that the end performance of the LTI algorithm is not too much affected by the initial waiting time estimates and smoothing parameter value. At each decision point, first we make an initial waiting time estimates as three times of operation processing times and the smoothing parameter is selected as 0.5. The pilot experiments also show that there is a significant improvement in the first a few iterations. We set a termination condition which stops the procedure either when the iteration number reaches to 30 or when there is no improvement in the performance measure for the last 10 iterations.

When scheduling decisions are made at decision points either by rule selection algorithm or lead time iteration, these decisions are implemented in the real-life system until the next decision point. If a certain rule is selected during the iterative simulation, then this rule is applied as a priority dispatching rule to select the job next to be processed on an available machine. If the decisions are made by LTI with ATC, then ATC is used as a priority rule with best waiting time estimates found in the iterations. During the iterative simulation process, the mechanism uses the best available information. But during the implementation of the decisions, the actual progress of the operations on the shop floor may be quite different. As mentioned earlier, we consider two types of events that will affect the actual progress in the system: (1) Machine breakdowns, and (2) Processing time variations. Hence in our study, we also analyze the interactions between the parameters of scheduling mechanism (i.e., scheduling period and look-ahead window) and these unexpected events.

4.4 Computational Study

In simulation experiments, an experimental design similar to ones in the previous chapter is conducted. The hypothetical reentrant classic job shop environment is with the following characteristics:

Jobs arrive continuously according to the Poisson process. The jobs have fixed number of operations selected from a discrete uniform distribution from 1 to 10. The operations are randomly processed through the machines. The shop contains 10 machines. Job weights are drawn from U[1,30]. Due dates are assigned randomly over a full range of flow allowances, with an average of 6.0 times the mean total job processing time.

The average utilization of the shop is determined by calibrating the arrival rate of the jobs. Arrival rate is adjusted to achieve approximately 70% utilization on the average at low level, and 90% utilization on the average at high level utilizations.

We consider two types of unexpected events in the study: (1) Processing time variation, and (2) Random machine breakdowns.

Best estimates of processing times are drawn from uniform distribution between 1 and 30. Actual processing times are determined from the best estimates as floows:

$$p'_{ij} = (1 + V \times U[-1.0, +1.0]) \times p_{ij}$$

where V defines the level of processing time variation, p_{ij} and p'_{ij} are the best estimate and actual processing times, respectively. In the current experimental study, V is set either at 0.0 (i.e. deterministic case) or at 0.60.

The machine efficiency is defined as in Chapter 3. We have three levels of efficiency for the machines. First level corresponds to no breakdown case, where efficiency is 100%. In the other two levels, the machines are all fallible with efficiencies 90% and 80%. By using the results of the Chapter 3, we define four levels of mean duration of breakdown in this set of experiments. The first level represents the no breakdown case with 0 mean duration of breakdown (or mean repair time). The next three levels are mean durations of breakdown of p_{avg}, $5p_{avg}$, and $10p_{avg}$, where p_{avg} is the mean operation processing time.

Again, the system is simulated by using SIMAN simulation language with some additional C subroutines linked in UNIX environment. We use 10 independent replications for output analysis. The simulation run length is 1,500 job completion. We initialize the system with 20 jobs to reach the desired system state faster.

Table 4.1: Experimental Factors and Levels

Factor	Number of Levels	Levels
Rule (R)	5	ATC, BD, COVERT, MOD, WSPT
Utilization (U)	2	70, 90
Processing Time Variation (V)	2	0, 0.6
Efficiency (E)	3	80, 90, 100
Mean Down Time (D)	4	0, 1, 5, 10
Look-ahead Window (F)	3	100, 250, 1500
Scheduling Period (P)	7	25, 50, 65, 100, 125, 250, 1500

By setting the simulation run length, we also determine our entire manufacturing horizon as 1,500 job completion. For this value, we determine two different forecasting horizon length: 250 job and 100 job arrivals. In our case, look-ahead window (or simulation window) is set equal to the forecasting horizon. The scheduling period that defines the regular decision points has three different levels according to the look-ahead window. At the first level, the scheduling period is equal to the forecasting horizon. In the other two cases, the scheduling period is shorter than the forecasting horizon. 1/4 of the forecasting horizon, 1/2 of the forecasting horizon. Hence, the scheduling period is equal to 65, 125, or 250 jobs for the case where the forecasting horizon is equal to 250, and it is equal to 25, 50, or 100 jobs for the case where forecasting horizon equals to 100. We use the case in which the forecasting horizon is equal to the entire manufacturing horizon as a benchmark. In this case, it is assumed that all the information for a very long time period is available for the ISM (The experimental factors and their levels are summarized in Table 4.1).

4.5 Experimental Results

The results of experiments are analyzed in three sections. The next section presents the results of single-pass versions of the rules. Then we discuss the results of iterative simulation-based scheduling system with multi-pass rule selection algorithm in Section 4.5.2 and with lead time iteration algorithm in Section 4.5.3.

Before going into discussions of the results, we note two points: (1) In our simulation experiments, we use common random numbers as a variance reduction technique as recommended by Law and Kelton [34]. By using common random numbers across the replications, we inherently construct the randomized blocking as discussed by Heikes, Montgomery and Rardin [23]. For this reason, we should separate the effects of blocking in our analyses. We show the source of variation due to blocking (or replications) as B in the analyses of the experimental results. (2) We have to be careful while detecting the effects of the down times and length of scheduling periods. For 100% efficiency level, we have no mean duration breakdown (it is shown as 0), and for other levels we have three mean durations of breakdown. In addition, the scheduling periods as 25, 50 and 100 jobs are defined according to 100-job look-ahead window, while 65, 125, and 250-job scheduling periods are defined with respect to 250-job look-ahead window. These types of factors are called *nested factors* in experimental design. In our case, scheduling period is nested in look-ahead window (shown as P(F)), and mean duration of breakdown is nested in efficiency (D(E)). The analysis of the results is performed by using SAS statistics software.

4.5.1 Results of Single-pass Experiments

The analysis of variance (ANOVA) for the weighted tardiness performance measure is presented in Table 4.2. The column named as "*Pr gt F*" shows significance level of the source. If the significance level of the analysis is taken as 0.05, the effects of the sources which yield probability smaller than 0.05 are statistically significant. According to the F-test, the main effects of all the factors are found to be significant. Also, all 2-way interactions of rules, utilization, processing time variation, efficiency and mean duration of breakdown nested in efficiency are statistically significant. The three-way interaction of rules, utilization and efficiency is effective on the weighted tardiness criterion.

Table 4.2: Analysis of Variance for Weighted Tardiness (WT) (Single-pass Rules)

Source	DF	Sum of Squares	F Value	Pr gt F
Model	148	4305218750.23	141.46	0.0001
Error	1251	257243013.65		
Source	**DF**	**Anova SS**	**F Value**	**Pr gt F**
B	9	78459689.45	42.40	0.0001
R	4	176377062.06	214.44	0.0001
U	1	729224603.36	3546.30	0.0001
R*U	4	66292561.20	80.60	0.0001
V	1	6162597.31	29.97	0.0001
R*V	4	217221.97	0.26	0.9011
U*V	1	500191.15	2.43	0.1191
R*U*V	4	130987.23	0.16	0.9588
E	2	1841908686.72	4478.70	0.0001
R*E	8	88237797.50	53.64	0.0001
U*E	2	275704820.80	670.39	0.0001
R*U*E	8	33849156.53	20.58	0.0001
V*E	2	858029.21	2.09	0.1246
R*V*E	8	322360.01	0.20	0.9915
U*V*E	2	31222.69	0.08	0.9269
R*U*V*E	8	262709.14	0.16	0.9958
D(E)	4	916468601.08	1114.22	0.0001
R*D(E)	16	18004682.52	5.47	0.0001
U*D(E)	4	66818835.40	81.24	0.0001
R*U*D(E)	16	2886616.38	0.88	0.5959
V*D(E)	4	282567.87	0.34	0.8486
R*V*D(E)	16	924265.08	0.28	0.9977
U*V*D(E)	4	229676.86	0.28	0.8915
R*U*V*D(E)	16	1063808.71	0.32	0.9947

Table 4.3: Duncan's Multiple Range Test for Weighted Tardiness (Single-pass Rules)

Factor: Rules (R)

Duncan Grouping	Mean	N	R
A	2810.84	280	MOD
B	1999.84	280	WSPT
C	1911.66	280	ATC
C	1907.01	280	BD
C	1895.07	280	COVERT

Factor: Utilization (U)

Duncan Grouping	Mean	N	Utilization
A	2826.60	700	90
B	1383.16	700	70

Factor: Processing Time Variation (V)

Duncan Grouping	Mean	N	Variation
A	2171.23	700	0.6
B	2038.53	700	0

Factor: Efficiency (E)

Duncan Grouping	Mean	N	Efficiency
A	3383.75	600	80
B	1373.59	600	90
C	462.13	200	100

We also apply Duncan's multiple range test for the main effects of the factors. Table 4.3 summarizes the Duncan's test results (In the table, significance level is 0.05, the levels with the same letter are not statistically different, and N is the number of observations in the corresponding level). Although the rules significantly affect the performance in overall, the performances of ATC, BD and COVERT are not statistically distinguishable. MOD is the worst among the rules producing 47% higher weighted tardiness than the best rules. Increase in utilization of the system adversely affect the system performance as in the case of processing time variation. Lower efficiency levels significantly increases the weighted tardiness.

Since the 2-way interactions of the rule factor and other factors are significant, these interactions are further depicted in the figures. Figure 4.4 shows the effects of utilization on the performances of the rules. At both levels of utilization, the performances of ATC, BD, and COVERT are very close to each other. MOD deteriorates sharply as utilization increases as compared with the other rules. Effects of processing time variation are shown in Figure 4.5. In this case, the deterioration of the rules are almost in the same rate while processing time variation increases. The effects of efficiency and down times are depicted in Figures 4.6–4.8. From these figures, the MOD rule is found to be very sensitive to the efficiency and duration of down times. While, all rules perform equally in terms of weighted tardiness in no breakdown case, MOD leaves the group as the efficiency gets lower. Also, the performance of WSPT is very close to the performances of ATC, BD and COVERT at 80% efficiency, whereas it is worse than ATC, BD and COVERT at 90% efficiency.

4.5.2 Results of Multi-pass Rule Selection Algorithm

The ANOVA for weighted tardiness for multi-pass rule selection algorithm is presented in Table 4.4. The main effects of all factors except the scheduling period nested in look-ahead window are statistically significant. All the 2-way interactions of utilization, efficiency, duration of down time nested in efficiency and look-ahead window are effective on the performance. 3-way interaction of utilization, efficiency and look-ahead window and 3-way interaction of utilization, down time and look-ahead window are significant. The length of scheduling period is effective only with down time and efficiency.

Table 4.5 summarizes the Duncan's multiple range test results for the multi-pass algorithm. The results support the findings of ANOVA for utilization, processing time variation, and efficiency. For look-ahead window, Duncan grouping shows that look-ahead of 100 jobs is significantly the worst among tested look-aheads. The look-ahead windows with 250 and 1500 jobs produce lowest measures without statistical difference. Even though the scheduling period does not have significant effect on the system performance, their effects are listed in Table 4.6. From these results, we observe that look-ahead window set at 250 jobs with scheduling periods of either 65 jobs or 250 jobs produce

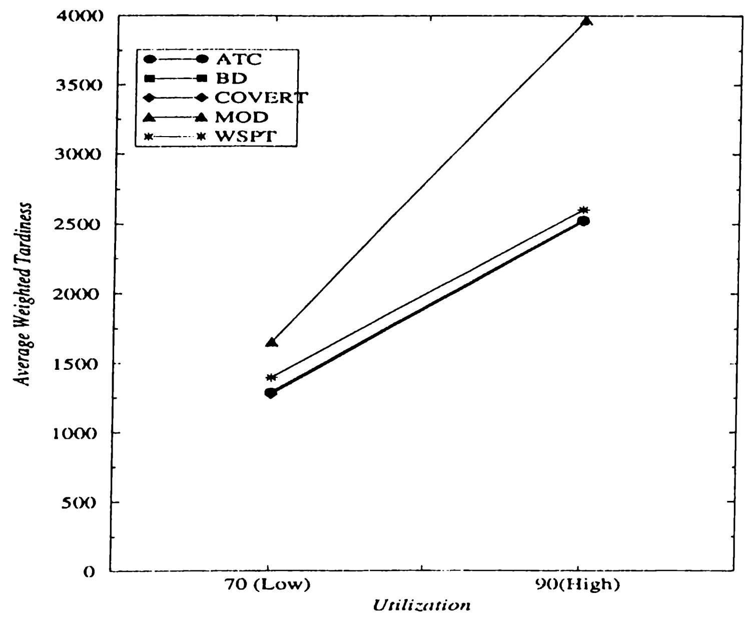

Figure 4.4: Average Weighted Tardiness versus Utilization (Single-pass)

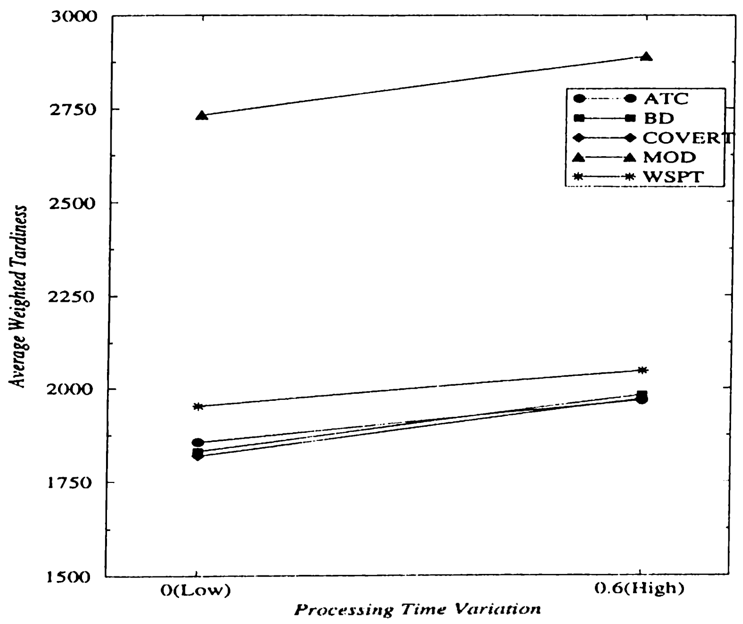

Figure 4.5: Average Weighted Tardiness versus Processing Time Variation (Single-pass)

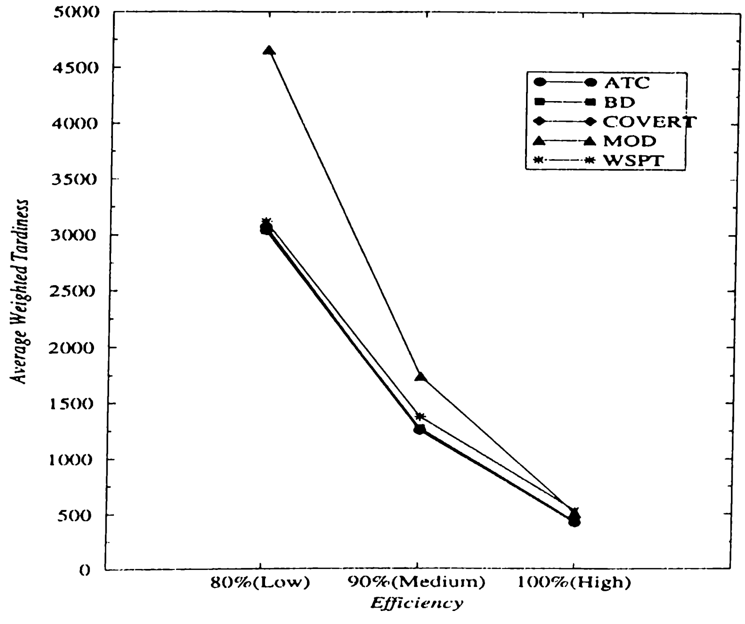

Figure 4.6: Average Weighted Tardiness versus Efficiency (Single-pass)

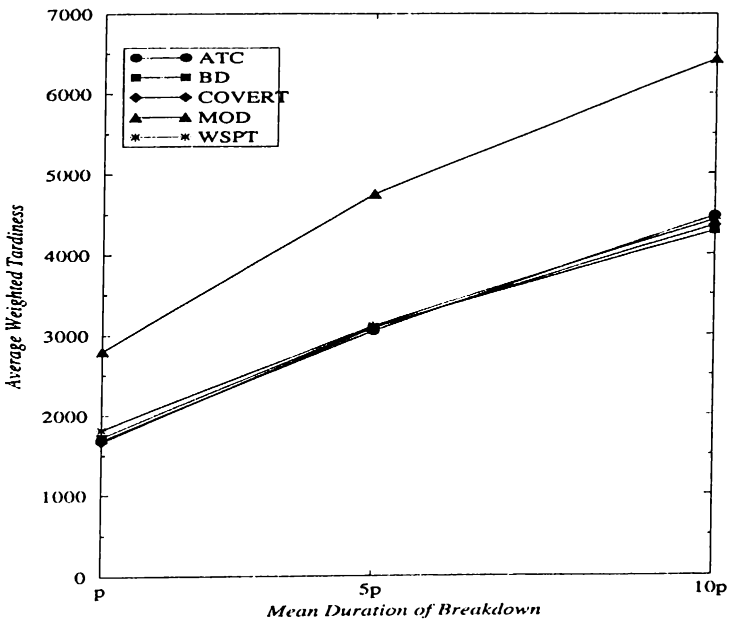

Figure 4.7: Average Weighted Tardiness versus Mean Duration of Breakdown (Single-pass, Efficiency=80%)

Table 4.4: Analysis of Variance for Weighted Tardiness (WT) (Multi-pass Rule Selection Algorithm)

Source	DF	Sum of Squares	F Value	Pr gt F
Model	204	5336807534.54	93.04	0.0001
Error	1755	493474510.01		
Source	**DF**	**Anova SS**	**F Value**	**Pr gt F**
B	9	128727548.77	50.87	0.0001
U	1	857480788.75	3049.56	0.0001
V	1	9672458.36	34.40	0.0001
U*V	1	995615.25	3.54	0.0600
E	2	2353820883.82	4185.58	0.0001
U*E	2	370356964.26	658.57	0.0001
V*E	2	1462751.85	2.60	0.0745
U*V*E	2	178918.97	0.32	0.7275
D(E)	4	1362144691.42	1211.09	0.0001
U*D(E)	4	146419209.21	130.18	0.0001
V*D(E)	4	1236815.98	1.10	0.3551
U*V*D(E)	4	1863158.63	1.66	0.1575
F	2	7611557.59	13.53	0.0001
U*F	2	7214977.51	12.83	0.0001
V*F	2	98048.79	0.17	0.8400
U*V*F	2	16597.43	0.03	0.9709
E*F	4	11283954.31	10.03	0.0001
U*E*F	4	9492677.70	8.44	0.0001
V*E*F	4	153526.66	0.14	0.9688
U*V*E*F	4	42890.62	0.04	0.9972
D*F(E)	8	12603550.76	5.60	0.0001
U*D*F(E)	8	10231745.42	4.55	0.0001
V*D*F(E)	8	1458950.07	0.65	0.7371
U*V*D*F(E)	8	1014110.52	0.45	0.8906
P(F)	4	382086.21	0.34	0.8513
U*P(F)	4	363088.62	0.32	0.8628
V*P(F)	4	1105326.33	0.98	0.4157
U*V*P(F)	4	1151821.94	1.02	0.3934
E*P(F)	8	250406.36	0.11	0.9988
U*E*P(F)	8	529866.13	0.24	0.9843
V*E*P(F)	8	1950854.51	0.87	0.5435
U*V*E*P(F)	8	1587157.07	0.71	0.6869
D*P(E*F)	16	9353208.53	2.08	0.0072
U*D*P(E*F)	16	9186562.31	2.04	0.0086
V*D*P(E*F)	16	7835016.21	1.74	0.0338
U*V*D*P(E*F)	16	7529747.64	1.67	0.0452

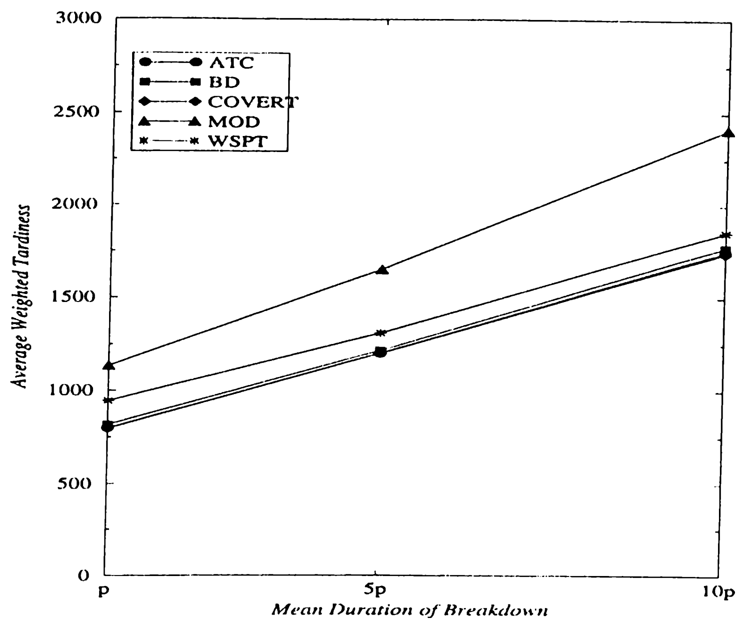

Figure 4.8: Average Weighted Tardiness versus Mean Duration of Breakdown (Single-pass, Efficiency=90%)

slightly better performance.

Table 4.7 shows the percentages of the selection of the rules according to the look-ahead window and scheduling period as well as the average CPU time for one replication of simulation. The results show that the selection percentages are highly dependent on the lengths of the look-ahead window and scheduling period. Although, ATC and COVERT are dominantly selected for long look-ahead windows, the other rules are alternatively selected in the short look-aheads. This shows that several rules can be preferred in the short term, although their long-term performances are dominated by others. The table depicts the effects of look-ahead window and scheduling period on the simulation time. Especially scheduling period is highly effective on the CPU time of the one simulation replication, since it directly determines the frequency of the multi-pass algorithm for one simulation replication of 1500 job completion.

The effects of utilization on the performances of selected look-ahead windows are depicted in Figure 4.9. The figure shows that look-ahead window with 100 jobs is mostly affected by the utilization. The effects of efficiency and mean

Table 4.5: Duncan's Multiple Range Test for Weighted Tardiness (Multi-pass Rule Selection Algorithm)

Factor: Utilization (U)

Duncan Grouping	Mean	N	Utilization
A	2620.44	980	90
B	1297.58	980	70

Factor: Processing Time Variation (V)

Duncan Grouping	Mean	N	Variation
A	2029.26	980	0.6
B	1888.76	980	0

Factor: Efficiency (E)

Duncan Grouping	Mean	N	Efficiency
A	3184.70	840	80
B	1247.80	840	90
C	415.60	280	100

Factor: Look-ahead Window (F)

Duncan Grouping	Mean	N	Look-ahead
A	2030.86	840	100
B	1912.96	280	1500
B	1902.51	840	250

duration of breakdown are shown in Figures 4.10-4.12. In higher efficiency levels or in short breakdowns, the look-ahead windows produce very similar performances. This effect is even stronger when the efficiency is high. However, when the efficiency decreases or mean duration of breakdown is longer, longer look-ahead windows significantly yield lower performances.

4.5.3 Results of Lead Time Iteration Algorithm

The ANOVA for weighted tardiness for lead time iteration algorithm is presented in Table 4.8. The F-test shows that the lead time iteration is robust in the sense that the majority of the possible effects are not significant on its performance. Only the main effects of utilization, variation, efficiency and duration of breakdown are significant. The lengths of the look-ahead window

Table 4.6: Effects of Look-ahead Window and Scheduling Period on Weighted Tardiness (Multi-pass Rule Selection Algorithm)

Scheduling Period	Look-ahead Window	Average
25	100	2012.26629
50	100	2051.58400
100	100	2028.73282
65	250	1894.89089
125	250	1922.09896
250	250	1890.54839
1500	1500	1912.96000

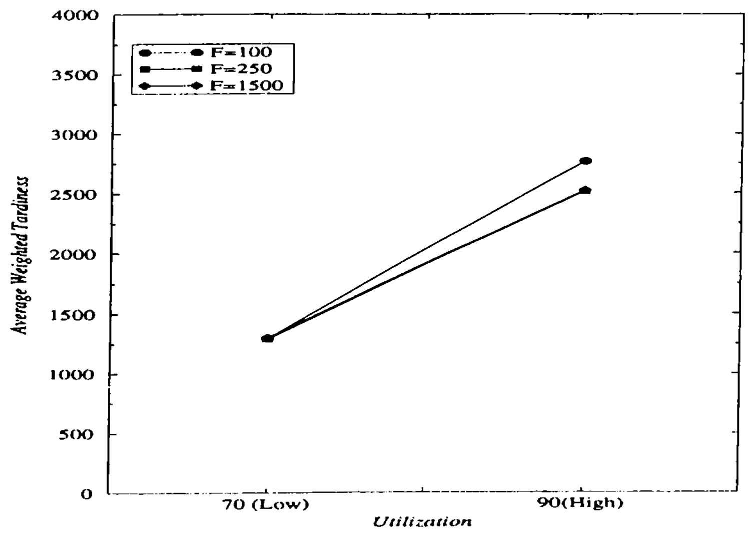

Figure 4.9: Average Weighted Tardiness versus Utilization (Multi-pass Rule Selection Algorithm)

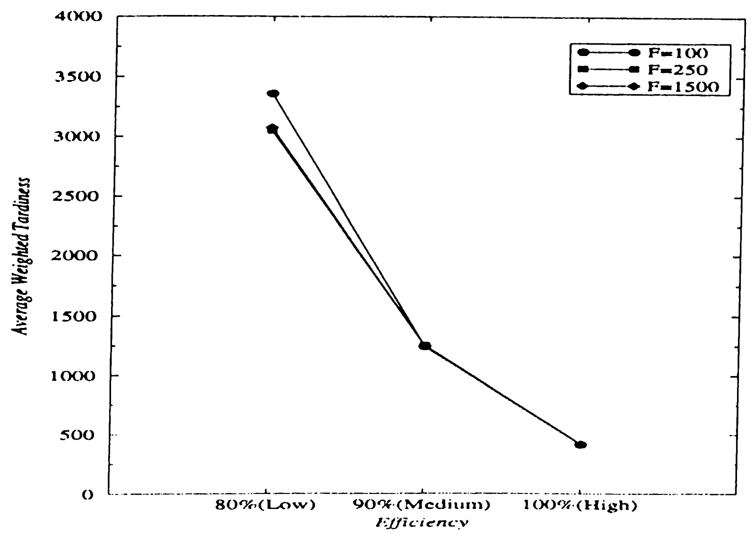

Figure 4.10: Average Weighted Tardiness versus Efficiency (Multi-pass Rule Selection Algorithm)

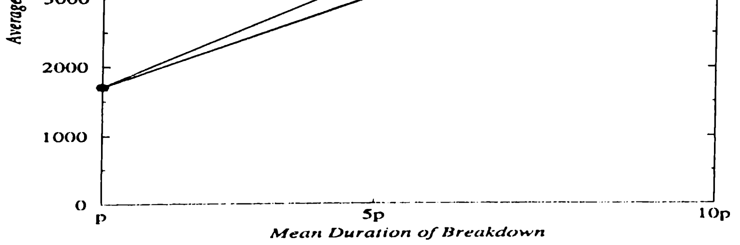

Figure 4.11: Average Weighted Tardiness versus Mean Duration of Breakdown (Multi-pass Rule Selection Algorithm, Efficiency=80%)

Table 4.7: Selection percentages of the rules at each decision point and the average CPU time per replication

L-ahead Win.	Sched. Per.	ATC	BD	COVERT	MOD	WSPT	CPU
100	25	32.49	16.47	34.59	10.00	5.95	1520.08
100	50	33.73	15.68	34.65	10.51	5.43	798.11
100	100	32.25	16.50	35.86	8.59	6.79	444.32
250	65	37.65	12.76	43.31	5.82	0.45	1256.92
250	125	38.23	12.64	43.52	4.98	0.61	731.55
250	250	38.57	11.57	43.57	5.71	0.57	416.12
1500	1500	54.00	2.00	40.00	4.00	0.00	297.76

and scheduling period are not effective for the lead time iteration algorithm. Among the two-way interactions, only utilization and efficiency interaction, and utilization and mean duration of breakdown interactions are significant. Duncan's range test supports these findings as shown in Table 4.9.

When we compare the single-pass performances with those of multi-pass rule selection algorithm, we observe that multi-pass algorithm provides 7.5% improvement over the rules on the average. When the utilization is high or the processing time variation is low, multi-pass rule selection algorithm is more advantageous. The lead time iteration algorithm produce lower weighted tardiness with an average improvement of 9.6% over the rules. In higher efficiency levels, the multi-pass rule selection yields more improved performances than in the lower efficiencies.

If we compare the single-pass performances with those of lead time iteration, we observe that the lead time iteration algorithm produces lower weighted tardiness with an average improvement of 9.6% over the rules. While it produces lower weighted tardiness than the multi-pass rule selection algorithm in high utilization level, the multi-pass rule selection algorithm is more effective in low levels of utilization. At all levels of processing time variation and efficiency, the lead time iteration is more effective as compared with the multi-pass rule selection algorithm and single-pass rules.

Table 4.8: Analysis of Variance for Weighted Tardiness (WT) (Lead Time Iteration Algorithm)

Source	DF	Sum of Squares	F Value	Pr gt F
Model	204	4628127753.47	126.82	0.0001
Error	1755	313947140.33		
Source	DF	Anova SS	F Value	Pr gt F
B	9	115808543.20	71.93	0.0001
U	1	750328450.97	4194.42	0.0001
V	1	8583146.32	47.98	0.0001
U*V	1	557577.82	3.12	0.0777
E	2	2168815600.28	6061.96	0.0001
U*E	2	284637329.57	795.58	0.0001
V*E	2	641829.88	1.79	0.1666
U*V*E	2	19358.52	0.05	0.9473
D(E)	4	1201889440.60	1679.67	0.0001
U*D(E)	4	91285489.58	127.57	0.0001
V*D(E)	4	143597.84	0.20	0.9380
U*V*D(E)	4	324093.93	0.45	0.7700
F	2	205876.14	0.58	0.5626
U*F	2	156292.16	0.44	0.6461
V*F	2	79285.56	0.22	0.8013
U*V*F	2	95432.61	0.27	0.7659
E*F	4	108445.48	0.15	0.9623
U*E*F	4	238551.48	0.33	0.8556
V*E*F	4	233195.59	0.33	0.8607
U*V*E*F	4	91062.26	0.13	0.9726
D*F(E)	8	297038.63	0.21	0.9897
U*D*F(E)	8	193969.17	0.14	0.9976
V*D*F(E)	8	359362.95	0.25	0.9807
U*V*D*F(E)	8	152200.06	0.11	0.9990
P(F)	4	152083.15	0.21	0.9316
U*P(F)	4	148489.04	0.21	0.9343
V*P(F)	4	7866.53	0.01	0.9998
U*V*P(F)	4	42043.34	0.06	0.9936
E*P(F)	8	177688.44	0.12	0.9983
U*E*P(F)	8	244903.55	0.17	0.9947
V*E*P(F)	8	159002.95	0.11	0.9989
U*V*E*P(F)	8	11471.68	0.01	1.0000
D*P(E*F)	16	458033.14	0.16	0.9999
U*D*P(E*F)	16	830654.05	0.29	0.9972
V*D*P(E*F)	16	391543.55	0.14	1.0000
U*V*D*P(E*F)	16	258803.44	0.09	1.0000

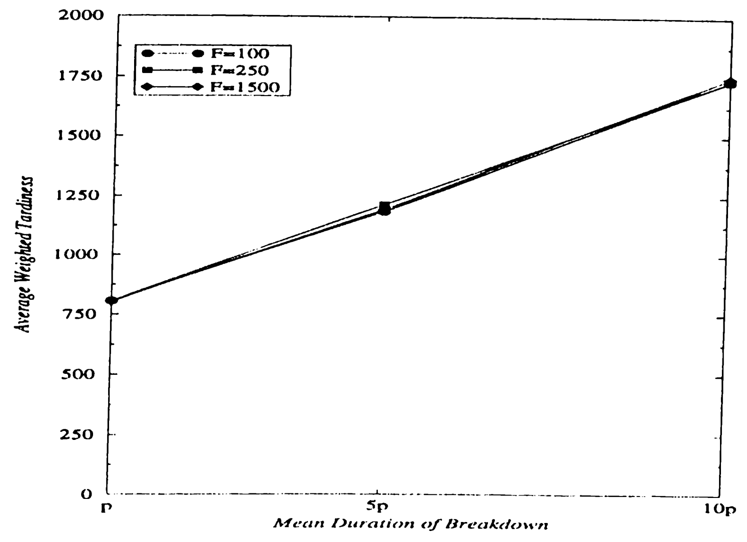

Figure 4.12: Average Weighted Tardiness versus Mean Duration of Breakdown (Multi-pass Rule Selection Algorithm, Efficiency=90%)

4.6 Conclusions

In this chapter, we have defined and used a new scheduling environment where the scheduler has some information about some events that will occur in the near future. For this type of environment, we have proposed a simulation-based scheduling mechanism. We have tested its effectiveness using multi-pass rule selection algorithm and lead time iteration method with an extended experimental study by creating both deterministic and stochastic environments. In addition, we have investigated the effects of unforeseen events such as machine breakdowns and processing time variation.

The results show that iterative simulation-based scheduling mechanism is very useful to improve the system performance. Among the algorithms tested, the lead time iteration algorithm is slightly better than multi-pass rule selection algorithm which produces better performance than the best single-pass rules.

It was observed that the unforeseen events adversely affect the system performance. Both processing time variations and machine breakdowns cause deterioration on the performance of both of the algorithms. This deterioration

Table 4.9: Duncan's Multiple Range Test for Weighted Tardiness (Lead Time Iteration Algorithm)

Factor: Utilization (U)

Duncan Grouping	Mean	N	Utilization
A	2538.99	980	90
B	1301.54	980	70

Factor: Processing Time Variation (V)

Duncan Grouping	Mean	N	Variation
A	1986.44	980	0.6
B	1854.09	980	0

Factor: Efficiency (E)

Duncan Grouping	Mean	N	Efficiency
A	3091.71	840	80
B	1254.09	840	90
C	404.50	280	100

Factor: Look-ahead Window (F)

Duncan Grouping	Mean	N	Look-ahead
A	1944.24	280	1500
A	1919.55	840	100
A	1912.99	840	250

is almost in the same rate for the algorithms.

The length of the look-ahead window is effective on the performance of multi-pass rule selection algorithm. However, the scheduling period is not too much effective. It is important to select the proper time period for the look-ahead window. If an appropriate look-ahead window selected, then the scheduling period is not so important. For the lead time iteration method, both the look-ahead window and scheduling period are not statistically significant. The levels of utilization and unexpected events determine the performance. This shows the robustness of the lead time iteration method. With improved waiting time estimations, the system performance is not too much affected by the revision of scheduling decisions with shorter scheduling periods. Since the lengths of the look-ahead and scheduling period directly affect the time to make the scheduling decisions, we can select the longer look-aheads with longer scheduling periods to decrease the computational burden.

In our implementation of iterative simulation-based scheduling mechanism, we make the scheduling decisions in two types according to the scheduling algorithm. In multi-pass rule selection, we select the rule to implement during the next scheduling period, whereas we seek to make better and better waiting time estimates in the lead time iteration method. Namely, we make global scheduling decisions at the beginning of scheduling period, and we implement the scheduling decisions dynamically over time. The remaining part of the scheduling decisions such as determining the start times of certain operations are determined in a dynamic manner during the implementation of the global decisions. We do not actually generate a static off-line schedule as opposed to the traditional approaches. If there is no unexpected event in the system, then the global decisions determine the full schedule over the scheduling period. However, if unforeseen events occur in the system, then the dynamic and state-dependent nature of the algorithms makes the remaining decisions as the events occur in the system. By this way, the algorithms which are dynamic and state-dependent in nature inherently develop their reactions to the events until the next decision point. At the next decision point, a new set of global decisions are made according to the new conditions of the system.

The results show that when we use such an algorithm which is dynamic and robust to certain changes, there is no need to revise the global scheduling decisions more frequently even if there are significant interruptions or variations in the system during the real-life implementation of these global decisions. This situation is especially true when the lead time iteration method is considered. If the algorithm implemented in the iterative simulation mechanism were not dynamic and robust, then revision of the scheduling decisions at certain time points would be a critical issue. In our case, we utilize the information for the next forecasting horizon explicitly during the decision making process. We assume that the arrival times and other characteristics of the jobs are known with certainty at the beginning of scheduling period. The only unforeseen events are machine breakdowns and processing time variations. However, the results of the experiments have indicated that these two events do not necessiate frequent revision of the scheduling decisions. Namely, stochastic environment which is characterized by processing time variations and machine breakdowns may not really magnify the effects of the timing of the scheduling decisions. The results also indicated that the information about the arrival times and other characteristics of the jobs are more critical. For this reason, if we had order cancelations, rush job arrivals, or operational changes in jobs, more frequent scheduling revisions would result better performances.

Also, if an off-line schedule was generated at decision points (all of the decisions are made at one time), then the proper setting the frequency of scheduling would be even more effective. In this case, the more frequent scheduling with shorter scheduling periods would yield better performances. The results of traditional scheduling/rescheduling study is that more frequent scheduling makes the performance better at the expense of system nervousness and solution time. When the scheduling is performed by considering only on-hand jobs in a static manner, and revisions of the schedules are made either in occurrence of dynamic events (job arrival, machine breakdown, etc.) or frequently without waiting the stochastic events, the system performance gets better. If we stick to the prespecified static schedule in spite of these events, then the overall performance even gets worse since the current schedule does not reflect the current shop status anymore.

Chapter 5

Conclusions

In this thesis we studied job shop scheduling problem under dynamic and stochastic environment. We first focused on the dynamic and deterministic job shop scheduling with priority dispatching rules. Then we developed and tested on-line reactive scheduling policies in a dynamic and stochastic environment under random machine breakdowns. Finally, we proposed a simulation-based scheduling mechanism and investigated the effects of stochastic events on the performance of the system. Detailed discussion of results of each study have been presented in the conclusion sections of chapters. Here, we summarize these conclusions in a comprehensive manner, and outline further research directions.

At the first stage of the thesis, we studied several priority dispatching rules for the dynamic job shop problems. Our special emphasis was on the bottleneck dynamics heuristic. We tested several versions of BD for the first time. We also compared the performances of bottleneck dynamics and its myopic counterpart (ATC) with the performances of 16 different priority dispatching rules. The results showed that the complex rules designed for weighted tardiness objectives such as COVERT, ATC and BD were better than the the other rules. However, BD was not significantly better than the COVERT and ATC, despite it used more global information about the job and the system itself. In simulation experiments, it was noted that the inserted idleness improved the performances of ATC and BD significantly if the system was not heavily loaded.

At the second stage of the thesis, we studied the reactive scheduling problem in stochastic environments where random machine breakdowns occur in the system. Here, we proposed several reactive scheduling policies consisting of rerouting mechanisms to response the unexpected machine breakdowns and tested their performances under various operating conditions. The results showed that the proper selection of a reactive policy against an unexpected event depends on the type and level of the event and the system characteristics. Utilization level of the system, duration and frequency of the unexpected events are some of these factors that affect the best reactive policy.

The last stage introduced a more realistic manufacturing environment where scheduling problem is an intermediate level of a global planning problem. In such an environment we proposed an iterative simulation-based scheduling mechanism that utilizes two different improvement procedures (multi-pass rule selection algorithm and lead time iteration algorithm). This mechanism uses discrete-event simulation as a decision making and evaluation tool. By using this scheduling mechanism, some part of the scheduling decisions are made at decision points while the remaining decisions are left to be determined according to the dynamic changes in the system. We analyzed the interactions between the forecasting horizon, scheduling period, look-ahead window, and the unexpected events such as machine breakdowns and processing time variations. The experimental results indicated that the improvement procedures improve the performances of the priority dispatching rules significantly at the expense of some computational time. Also, the determination of the look-ahead window is an important factor for the multi-pass rule selection algorithm, while lead time iteration algorithm is relatively robust to the lengths of forecasting horizon and scheduling periods.

From these conclusions, we suggest the future research topics as follows:

- Although existing priority dispatching rules are tested in our study, there is a need to further study the dynamic job shop scheduling problem and propose new rules for different objectives. BD was an attempt to achieve this goal, but it was not sufficient to get better performances. As the experimental results have indicated the performance of BD in terms of percent of tardy jobs is very good, but its conditional tardiness performance is not very good. The improvement methods on the conditional tardiness without sacrificing percent of tardy job performance might yield very good tardiness performances. Inserted idleness in a more extended way can be helpful.

- Iterative improvement techniques for priority dispatching approach is very effective as compared with the single-pass rules. However, to utilize this method, either we must have static environment, we have perfect information about the future or we implement the improvement method as an off-line scheduling method in a dynamic environment. Two iterative methods in the second case are investigated in our study. The investigation of the methods in other cases are remained for the further research.
- The reactive scheduling and control in stochastic environments has to be further studied. The event-driven rescheduling, partial rescheduling, and performance-driven rescheduling are needed to be investigated in detail in the future. The classification of the events and some characteristics of the system are important factors to select the best reactive policy. For this reason, it is essential to determine which reactive scheduling and control scheme is utilized at which level and type of events and in which environment.
- Discrete-event simulation can be used as an evaluation tool to compare the different scheduling alternatives. One implementation is proposed in this study. When the computer power and information systems capability of the today's systems are considered, the gain from the simulation in such a way might be more encouraging. In this track, it is needed to efficiently combine the physical manufacturing systems with the simulation mechanisms and further investigate the simulation-based scheduling/planning systems. The application area of the simulation might be

rule selection, determination of bottleneck machine, or even generation of a static off-line schedule.

These are some of the areas open for the future research.

Bibliography

[1] J. Adams, E. Balas, and D. Zawack. The shifting bottleneck procedure for job shop scheduling. *Management Science*, 34(3):391-401, 1988.

[2] M. S. Akturk and E. Gorgulu. Reactive scheduling under a machine breakdown. Technical Report IEOR-9316, Bilkent University, Department of Industrial Engineering, 1993.

[3] E. J. Anderson and J. C. Nyirenda. Two new rules to minimize tardiness in a job shop. *International Journal of Production Research*, 28(12):2277–2292, 1990.

[4] D. Applegate and W. Cook. A computational study of job shop scheduling. *ORSA Journal on Computing*, 3:149–156, 1991.

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

[6] K. R. Baker. *Elements of Sequencing and Scheduling.* Dortmouth College, 1994.

[7] K. R. Baker and J. W. M. Bertrand. A dynamic priority rule for sequencing against due dates. *Journal of Operations Management*, 3(1):37–42, 1982.

[8] K. R. Baker and J. J. Kanet. Job shop scheduling with modified due dates. *Journal of Operations Management*, 4(1):11–22, 1983.

[9] J. C. Bean, J. R. Birge, J. Mittenthal, and C. E. Noon. Match-up scheduling with multiple resources, release dates and disruptions. *Operations Research*, 39(3):470–483, 1991.

[10] G. Bengu. A simulation-based scheduler for flexible flowlines. *International Journal of Production Research*, 32(2):321–344, 1994.

[11] E. S. Byeon, S. David Wu, and R. H. Storer. Decomposition heuristics for the job-shop scheduling problems. Technical Report 93T-008, Department of Industrial Engineering, Lehigh University, 1993.

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

[13] D. P. Christy and J. J. Kanet. Manufacturing systems with forbidden early shipment: implications for choice of scheduling rules. *International Journal of Production Research*, 28(1):91–100, 1990.

[14] L. K. Church and R. Uzsoy. Analysis of periodic and event-driven rescheduling policies in dynamic shops. *International Journal of Computer Integrated Manufacturing*, 5(3):153–163, 1992.

[15] W. J. Davis and A. T. Jones. A real-time production scheduler for a stochastic manufacturing environmment. *International Journal of Computer Integrated Manufacturing*, 1(2):101–112, 1988.

[16] A. Dutta. Reacting to scheduling exceptions in FMS environments. *IIE Transactions*, 22(4):300–314, 1990.

[17] D. A. Elvers. Job shop dispatching using various due date setting criteria. *Production and Inventory Management*, 14(4):62–69, 1973.

[18] M. S. Fox and S. F. Smith. ISIS - a knowledge-based system for factory scheduling. *Expert Systems*, 1(1):25–49, 1984.

[19] C. M. Harmonosky. Implementation issues using simulation for real-time scheduling, control, and monitoring. In *Proceedings of Winter Simulation Conference*, pages 595–598, 1990.

[20] C. M. Harmonosky. Analysis of two key issues for using simulation for real-time production control. In *Proceedings of 2nd Industrial Engineering Research Conference*, pages 41–45, 1993.

[21] C. M. Harmonosky and S. F. Robohn. Real-time scheduling in a computer integrated manufacturing: a review of recent research. *International Journal of Computer Integrated Manufacturing*, 4(6):331–340, 1991.

[22] Y. He, M. L. Smith, and R. A. Dudek. Effects of inaccuracy of processing time estimation on effectiveness of dispatching rule. In *Proceedings of 3rd Industrial Engineering Research Conference*, pages 308–313, 1994.

[23] R. G. Heikes, D. C. Montgomery, and R. L. Rardin. Using common random numbers in simulation experiments - an approach to statistical analysis. *Simulation*, 27(3):81–85, 1976.

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

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

[26] S. Jain and W. J. Foley. Real-time control of manufacturing systems with redundancy. *Computer and Engineering*, 2, 1987.

[27] J. J. Kanet. On anomalies in dynamic ratio type scheduling rules: A clarifying analysis. *Management Science*, 28(11):1337–1341, 1982.

[28] J. J. Kanet and J. C. Hayya. Priority dispatching with operation due dates in a job shop. *Journal of Operations Management*, 2(3):155–163, 1982.

[29] J. J. Kanet and Z. Zhou. A decision theory approach to priority dispatching for job shop scheduling. *Production and Operations Management*, 2(1):2–14, 1993.

[30] S. Karabuk. An analysis of FMS scheduling problem: A beam search based algorithm and comparison of scheduling schemes. Master's thesis, Bilkent University, Department of Industrial Engineering, 1994.

[31] M. H. Kim and Y. Kim. Simulation-based real-time scheduling in a flexible manufacturing system. *Journal of Manufacturing Systems*, 13(2):85–93, 1994.

[32] S. C. Kim and P. M. Bobrowski. Impact of sequence-dependent setup time on job shop scheduling performance. *International Journal of Production Research*, 32(7):1503–1520, 1994.

[33] A. S. Kiran, S. Alptekin, and A. C. Kaplan. Tardiness heuristic for scheduling flexible manufacturing systems. *Production Planning and Control*, 2(3):228–241, 1991.

[34] A. M. Law and W. D. Kelton. *Simulation Modeling and Analysis.* McGraw Hill, New York, 1991.

[35] S. R. Lawrence and T. E. Morton. Resource-constrained multi-project scheduling with tardy costs: Comparing myopic, bottleneck, and resource pricing heuristics. *European Journal of Operational Research*, 64:168–187, 1993.

[36] O. Z. Maimon. Real-time operational control of flexible manufacturing systems. *Journal of Manufacturing Systems*, 6(2):125–136, 1987.

[37] H. Matsuura, H. Tsubone, and M. Kanezashi. Sequencing, dispatching, and switching in a dynamic manufacturing environment. *International Journal of Production Research*, 31(7):1671–1688, 1993.

[38] T. E. Morton, S. R. Lawrence, S. Rajagopolon, and S. Kekre. SCHED-STAR: A price based shop scheduling module. *Journal of Manufacturing and Operations Management*, 1(3):131–181, 1988.

[39] T. E. Morton and D. Pentico. *Heuristic scheduling systems with applications to production and project management.* John Wiley and Sons, New York, 1993.

[40] T. E. Morton and R. M. V. Rachamadugu. Myopic heuristics for the single machine weighted tardiness problem. Technical Report 30-82-83, Graduate School of Industrial Administration, Carnegie-Mellon University, 1982.

[41] T. E. Morton and P. Ramnath. Guided forward tabu/beam search for scheduling very large dynamic job shops, I. Technical Report 1992-47, Graduate School of Industrial Administration, Carnegie-Mellon University, 1992.

[42] T. E. Morton and M. R. Singh. Implicit costs and prices for resources with busy periods. *Journal of Manufacturing and Operations Management*, 1(2):305–322, 1988.

[43] A. P. Muhlemann, A. G. Lockett, and C. K. Farn. Job shop scheduling heuristic and frequency of scheduling. *International Journal of Production Research*, 20(2):227–241, 1982.

[44] S. Y. Nof and F. H. Grant. Adaptive/predictive scheduling: review and a general framework. *Production Planning and Control*, 2(4):298–312, 1991.

[45] I. M. Ovacik and R. Uzsoy. Rolling horizon algorithms for a single-machine dynamic scheduling problem with sequence-dependent setup times. *International Journal of Production Research*, 32(6):1243–1263, 1994.

[46] P. S. Ow. Focused scheduling in proportionate flow shops. *Management Science*, 31(7):852–869, 1985.

[47] P. S. Ow and T. E. Morton. Filtered beam search in scheduling. *International Journal of Production Research*, 26(1):35–62, 1989.

[48] P. S. Ow and T. E. Morton. The single machine early/tardy problem. *Management Science*, 35(2):177–191, 1989.

[49] N. Raman and F. B. Talbot. The job shop tardiness problem: A decomposition approach. *European Journal of Operations Research*, 69:187–199, 1993.

[50] N. Raman, F. B. Talbot, and R. V. Rachamadugu. Due date based scheduling in a general flexible manufacturing system. *Journal of Operations Management*, 8:115–132, 1989.

[51] R. Ramasesh. Dynamic job shop scheduling: A survey of simulation research. *OMEGA*, 18(1):43–57, 1990.

[52] T. R. Rohleder and G. D. Scudder. Scheduling rule selection for the forbidden early shipment environment: a comparison of economic objectives. *International Journal of Production Research*, 30(1):129–140, 1992.

[53] R. S. Russell, E. M. Dar-el, and B. W. Taylor III. A comparative analysis of the COVERT job sequencing rule using various shop performance measures. *International Journal of Production Research*, 25(10):1523–1539, 1987.

[54] G. D. Scudder, T. R. Hoffmann, and T. R. Rohleder. Scheduling with forbidden early shipments: alternative performance criteria and conditions. *International Journal of Production Research*, 31(10):2287–2305, 1993.

[55] C. R. Shultz. An expediting heuristic for the shortest processing time dispatching rule. *International Journal of Production Research*, 27(1):31–41, 1989.

[56] S. F. Smith, P. S. Ow, J. Potvin, and D. C. Matthys. An integrated framework for generating and revising factory schedules. *Journal of Operational Research Society*, 41(6):539–552, 1990.

[57] D. Sun and L. Lin. A dynamic job shop scheduling framework: a backward approach. *International Journal of Production Research*, 32(4):967–985, 1994.

[58] P. Tayanithi, S. Manivannan, and J. Banks. A knowledge-based simulation architecture to analyze interruptions in a flexible manufacturing system. *Manufacturing Systems*, 11(3):195–214, 1993.

[59] A. P. J. Vepsalainen and T. E. Morton. Priority rules for job shops with weighted tardiness costs. *Management Science*, 33(8):1035–1047, 1987.

[60] A. P. J. Vepsalainen and T. E. Morton. Improving local priority rules with global lead-time estimates: A simulation study. *Journal of Manufacturing and Operations Management*, 1(2):102–118, 1988.

[61] J. K. Weeks. A simulation study of predictable due dates. *Management Science*, 25(4):363–373, 1979.

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

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

[64] S. David Wu, E. S. Byeon, and R. H. Storer. A graph-theoretic decomposition of job shop scheduling problems to achieve scheduling robustness. Technical Report 93T-009, Department of Industrial Engineering, Lehigh University, 1993.

[65] M. Yamamoto. An approximation solution of machine scheduling problems by decomposition method. *International Journal of Production Research*, 15(6):599–608, 1977.

[66] M. Yamamoto and S. Y. Nof. Scheduling and rescheduling in the manufacturing operating system environment. *International Journal of Production Research*, 23(4):705–722, 1985.

[67] M. Ye and G. B. Williams. Technical note on an expediting heuristic for the shortest processing time dispatching rule. *International Journal of Production Research*, 29(1):209–213, 1991.