ANALYSIS OF SCHEDULING PROBLEMS IN DYNAMIC AND STOCHASTIC FMS ENVIRONMENT: COMPARISON OF RESCHEDULING POLICIES

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
Ö. Batuhan Kızılışık
September, 2001

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.

Assoc. Prof. Ihsan Sabuncuoglu (Principle 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.

Assist. Prof. Bahar Kara

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.

Assoc. Prof. Erdal Erel

Approved for the Institute of Engineering and Sciences:

Prof. Mehmet Baray

Director of Institute of Engineering and Sciences

ABSTRACT

AN ANALYSIS OF SCHEDULING PROBLEMS IN DYNAMIC AND STOCHASTIC FMS ENVIRONMENT: COMPARISON OF RESCHEDULING POLICIES

Omer Batuhan Kizilisik
M.S. in Industrial Engineering
Supervisor: Assoc. Prof. Ihsan Sabuncuoglu
September, 2001

In this thesis, we study the reactive scheduling problems in a dynamic and stochastic flexible manufacturing environment. Specifically, we test different scheduling policies (how-to-schedule and when-to-schedule policies) under process time variations and machine breakdowns in a flexible manufacturing system. These policies are then compared with on-line scheduling schemes. The performance of the system is measured for the mean flowtime criterion. In this study, a beam search based algorithm is used. The algorithm allows us to generate partial or full schedules. The results indicate that on-line scheduling schemes are more robust than the off-line algorithm in dynamic and stochastic environments.

Keywords: Flexible Manufacturing Systems, Reactive Scheduling, Simulation.

ÖZET

ESNEK ÜRETİM SİSTEMLERİNDE ÇİZELGELEME PROBLEMİNİN DİNAMİK ORTAMDA ANALİZİ: TEPKİSEL ÇİZELGELEME METODLARININ KARŞILAŞTIRILMASI

Ömer Batuhan Kızılışık

Endüstri Mühendisliği Bölümü Yüksek Lisans

Tez Yöneticisi: Doç. İhsan Sabuncuoğlu

Eylül, 2001

Bu çalışmada rassal ve dinamik esnek üretim sistemlerinde ki tepkisel çizelgeleme problemi incelenmiştir. Farklı tepkisel çizelgeleme (ne zaman ve nasıl çizelgeleme) metotları rassal esnek üretim sistemlerinde sunulmuş ve test edilmiştir. Bu tepkisel çizelgeleme metotları daha sonra anında yönlendirme yaklaşımı ile karşılaştırılmıştır. Bu çalışmada süzülmüş ışın taramasına dayalı bir algoritma kullanılmıştır. Bu algoritmayı kullanarak kısmi çizelgeleme yöntemide araştırılmıştır. Yapılan bu çalışma sonucunda anında yönlendirme yaklaşımının önceden çizelgeleme yaklaşımına göre dinamik ve rassal ortamdan daha az etkilendiği bulunmuştur.

Anahtar Sözcükler: Esnek Üretim Sistemleri, Tepkisel Çizelgeleme, Benzetim.

Annem, Babam ve Kardeşime

ACKNOWLEDGMENTS

I would like to thank my advisor Assoc. Prof. Ihsan Sabuncuoglu for his supervision, encouragement and understanding throughout my long graduate education in Bilkent University.

I am also indepted to Bahar Kara and Erdal Erel for their valuable comments on this thesis.

I greatly appreciate Bilkent University administration for providing advanced computing facilities.

Table of Contents

1 INTRODUCTION **1**

1.1) Problem Definition 3

2 LITERATURE SURVEY **5**

2.1) Reactive Scheduling in Single Machine Environment 4

2.2) Reactive Scheduling in Job Shop Environment 10

2.3) Reactive Scheduling in FMS Environment 12

2.4) Observations and Open Research Points 13

3. SCHEDULING SYSTEM **16**

3.1) Background Information About Simulation and Scheduling 16

3.2) The Elements of Proposed Scheduling System 18

3.3) Schedule Generation Module 21

3.4) System Considerations and Experimental Conditions 22

3.5) Rescheduling Policies Considered in This Study 24

4. COMPARISON OF RESCHEDULING POLICIES **27**

4.1) Pilot Experiments 27

4.2) Comparison of Rescheduling Policies in a Dynamic Environment 32

4.2.1) How to Schedule policies 32

4.2.2) When to Schedule policies 37

4.3) Process Time Variation 41

4.4) Machine Breakdowns 46

CONCLUSION **51**

BIBLIOGRAPHY **53**

APPENDIX **56**

LIST of FIGURES

3.1 Simulation based scheduling system 19
4.1 Comparison of Meanflow times for two different initial job populations for high SF and RF 28
4.2 Comparison of limited and unlimited queue capacities for different scheduling policies 29
4.3 Comparison of flexibilities for different scheduling policies 30
4.4 Comparison of different scheduling frequencies for F-LOW 31
4.5 Comparison of how to schedule policies 33
4.6 Differences between full and partial schedules for different scheduling frequencies 35
4.7 Comparison of flexibilities for full and partial schedules 37
4.8 Comparison of when to schedule policies for partial and full schedules . . 39
4.9 Comparison of PERIODIC and ARRIVAL policies for different Partial Scedules 41
4.10 Mean flowtimes for the ARRIVAL, PERIODIC, RATIO policies for PV=0 and PV=0.4 44
4.11 Change of preformance with PV and change of performance with partial schedule 45
4.12 Mean flowtimes of scheduling policies for no breakdown and breakdown case 47
4.13 Change of performances with machine breakdown and with partial schedule 48

LIST of TABLES

2.1 Classification of the papers in reactive scheduling 15

3.1 Internal factors and their levels 23

3.2 When to reschedule policy 26

4.1 Outputs for limited and unlimited buffer capacities 57

4.2 Comparison of H_100 and H_50 for F-LOW and F-HIGH 60

4.3 Comparison of different scheduling policies for F-LOW 63

4.4 Comparison of when to schedule policies for H_50 65

4.5 Comparison of when to schedule policies for H_100 68

4.6 Performances of scheduling policies for different scheduling frequencies . 70

4.7 Comparison of ARRIVAL and PERIODIC policies for different Partial schedules 71

4.8 Comparison of How to Schedule policies for Partial schedule and PV . . . 72

4.9 Performance measures for machine breakdown for F-LOW and F-HIGH . 73

4.10 The results of Dispatch Policy 74

4.11 The performances of scheduling policies with and without PV 75

4.12 t-test for the Ratio Policy and the Dispatch Rule 76

4.13 t-test for the Ratio and the Periodic Policies 76

CHAPTER 1

INTRODUCTION

A flexible manufacturing system (FMS) is highly automated and capable of producing a variety of parts simultaneously. The flexibility of an FMS is mainly due to the capability of processing stations, which can perform several different types of operations, and its material handling system, which provides fast and flexible part transfer within the system. However, the benefits of FMS are not easy to realize and its several design and operational problems need to be solved in order to get a full benefit from these systems. One of the critical decision is scheduling. The scheduling decision in an FMS environment is considered to be a detailed minute-by-minute scheduling of machines, material handling system, and other support equipment (Sabuncuoglu and Hommertzheim, 1992). In this thesis, we will study the scheduling problem in a dynamic FMS environment.

In general, scheduling is a decision-making process, which concerns the allocation of limited resources to tasks over time. Since, scheduling serves as an overall plan on which many other shop activities are based, it plays an important role in manufacturing systems in order to have timely and costly effective production. In practice, a feasible schedule alone is rarely the only goal of scheduling, as there may be other objectives and preferences such as minimising mean flowtime, or tardiness. By properly planning and timing of shop floor activities, various system performance measures (due dates, utilisation. flowtime, etc.) can be optimised.

There are two key elements in any scheduling system: schedule generation and control. Schedule generation is viewed as the predictive mechanism that determines planned start and completion times of operations of the jobs. On the other hand, the control element has to do with updating schedules or reacting to unexpected random events. In other words, this system monitors the execution of the schedule and revises it to cope with unexpected events such as, machine breakdowns, arrival of hot jobs, etc. In practice, the performance of predetermined schedules degrades so quickly that an appropriate reaction should be made in order to return the systems back to the planned or desired performance.

As discussed in Sabuncuoglu and Toptal (2000), scheduling can be classified in many different ways (static vs dynamic, deterministic vs stochastic, on-line vs off-line, centralised vs hierarchical, etc.). One of the classifications can be made with respect to schedule generation mechanism (i.e., *on-line* and *off-line* schedule generation). In *off-line scheduling*, all available jobs are scheduled all at once for the entire planning horizon whereas in the *on-line scheduling*, schedule is made one at a time when it is needed according to the change in the system conditions. Thus, in *on-line scheduling*, the schedule is constructed over time (not all at once). Priority dispatching is a good example of on-line scheduling because decisions are made one at a time as the system state changes. Generally speaking, schedules are easily generated by using on-line dispatching rules. But the solution quality is sacrificed due to the myopic nature of these rules (Sabuncuoglu and Bayiz, 2000).

The majority of the published literature on the scheduling problem deals with the task of schedule generation. Although schedule control (or reactive) part is very important, especially in today's highly competitive manufacturing environments, it has not been adequately studied in the literature (Sabuncuoğlu and Bayız, 2000).

1.1. Problem Definition

In this thesis, we analyse the reactive scheduling problem in a flexible manufacturing (FMS) environment. Specifically, we investigate two important issues that have not been addressed thoroughly in the literature. These are as follows:

1. In most of the studies that are concerned with comparison of on-line and off line scheduling schemes, a deterministic and static manufacturing environment is used.

2. Different rescheduling policies are not compared with each other under dynamic and stochastic environment.

In this study, a simulation model is used to execute the schedules generated by different scheduling schemes in stochastic and dynamic manufacturing environments. The simulation model is linked with various scheduling algorithms to form a simulation based scheduling system. This system is composed of a simulation model, a controller and a scheduling module. The scheduling module contains on-line scheduling algorithm as well as the scheduling algorithm developed in this research.

We use a beam search based scheduling algorithm. This algorithm considers scheduling factors such as dynamic job arrivals, machine breakdowns, flexibility, and material handling capacity. The algorithm can develop schedules for varying scheduling periods. It can also generate partial schedules. This feature of the algorithm makes it possible to test various scheduling policies.

In the thesis, we study reactive scheduling problem in an FMS, which has several machines and material handling components and dynamic job arrivals. We develop different *when to schedule* and *how to schedule* reactive policies, which are defined in the next chapters. We test their performances under process time variation and machine breakdowns. We then compare their performances with on-line scheduling policy.

The rest of the thesis is organized as follows: In the next chapter, we provide a review of the related research on reactive scheduling. At the end of the chapter, the papers are classified according to their problem environments, schedule generation methods and reactive control implementations. In Chapter 3 the scheduling algorithm and the experimental conditions are described in detail and implementation issues are discussed. In Chapter 4, different rescheduling policies are tested under various conditions. Finally, concluding remarks are given in Chapter 5.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we will examine the relevant studies in the literature in a detailed manner. To provide an organized presentation, we will begin from the single machine environment. Then we will look at job shop studies. Finally, we will consider the studies in FMS environments.

2.1 Reactive Scheduling in Single Machine Environment

Rescheduling refers to the process of generating a new feasible schedule upon the occurance of a disruption (Svestka, Abumaziar, 1997). Many researches who address the rescheduling problem either reschedule resources every time an event that alters the system condition (continuous rescheduling) or reschedule the facility periodically (periodic rescheduling). Church and Uzsoy (1992) analyse the performance of such a hybrid rescheduling, which is referred as event-driven rescheduling policy in a single machine environment. Here, events that change the state of the facility are classified into those requiring immediate action (or exceptions) and those that can be ignored.

Thus, the scheduling is triggered in periods and when an exception occurs. At each rescheduling point, static schedules are generated for available jobs by using EDD rule. In this procedure, in addition to periodic rescheduling, arrival of a job with tight due date also causes a need for rescheduling of the system. Computational experiments with the maximum lateness (L_{max}) criterion show that the benefits of extra scheduling diminish rapidly. This means that a well designed event-driven policy can result in a good performance with less computational effort. The algorithm they used can be summarised as follows: for every job i arriving between period (i-1)T and iT, $s_i = d_i - r_i$ is computed. Here, T is the length of the period, d_i denotes the due date of job i, r_i is the ready time of job i, and s_i is called the slack between the due date and ready time of job i. If s_i is smaller than a constant value, called window length (w), then a new schedule is generated for all unprocessed jobs and implemented this until next rescheduling point. Also, at all points iT a new schedule is generated for unprocessed jobs using EDD rule.

Wu and Storer (1992) study the single machine rescheduling problem with a single unforeseen disruption. They propose several heuristics for rescheduling. The authors define the impact of schedule change in two ways: deviations of start time of operations between the new schedule and the original schedule, and deviations of sequence of the jobs between the new schedule and the original schedule. They minimise an objective function with two elements: makespan and impact of the schedule change. Namely, $Z(S) = r.M(S) + (1-r).D(S)$, where M(S) is the measure of performance (makespan) given the schedule S, D(S) is the measure of predictability (robustness) of the schedule S, and r is a number between (0,1). The schedule predicted before execution may be modified as time passes due to unpredictable changes in the shop floor. Thus D(S) measures the difference between the predicted schedule and the released one. They use the makespan criteria for M(S) and the deviation between the predicted schedule and the released schedule is used for D(S). The authors employ local search heuristics to optimise this bicriterion problem. The first set of heuristics are pairwise swapping methods, and the second set are based on local search using genetic algorithms. During local search, all heuristics start with a minimal makespan and a minimal deviation schedule. The minimal makespan schedule is generated using Carlier's Algorithm (Carlier, 1982). Two local search heuristics based on pairwise interchange of jobs in the sequence were developed, called straightforward approach and bicriterion steepest descent. Within each type of heuristics, both adjacent and all-pair interchanges were implemented resulting in a total of four methods. Also, the genetic algorithm is implemented using two different procedures, called α-ε grid search and r grid search procedures. Thus, totally six different methods are used to solve the problem. A set of experiments is conducted to test the efficency of these heuristics. They also compare the schedules of this heuristics with the optimal solution of the problem. The results indicate that: all-pair search methodology is the best, and this is followed by GA (Generic Algorithm). The adjacent pair search yield the worst performance.

In another study, Daniels and Kouvelis (1995) study "Robust Scheduling", which is the determination of a schedule whose performance is insensitive to the potential realisations of task parameters in a single machine environment using flowtime criterion. Specifically, they focused on identifying and applying schedule robustness in a single machine environment with a performance criterian of total flow. In their study, the authors assumed variable processing time parameters and no machine breakdown. What distinguishes robust scheduling from the predictable scheduling is that robust scheduling focuses on minimising the effects of disruptions on the performance measure, whereas predictable scheduling tries to ensure that the predictive and realised schedule do not differ drastically in terms of the completion times of jobs. They define two measures of schedule robustness which are called absolute robust schedule and relative robust schedule. Their model can be summarised as follows: Let λ represents a unique set of processing times of the job. Here, processing time uncertainty is described through a set of processing time scenarios Λ, where λ is an element of Λ. Also, let σ denotes a permutation of n jobs and σ_{λ}' denotes the optimal sequence given the processing times λ. Then, define $d(\sigma,\lambda)$ =

$P(\sigma,\lambda)$ - $P(\sigma_\lambda',\lambda)$ and $r(\sigma,\lambda) = P(\sigma,\lambda)/P(\sigma_\lambda',\lambda)$, where $P(\sigma_\lambda',\lambda)$ is the optimal schedule of problem instance λ. The objective here is, to obtain a schedule which satisfies $d' = \min d$, where $d = \max d(\sigma,\lambda)$ or $r' = \min r$, where $r = \max r(\sigma,\lambda)$. The first schedule is called absolute robust schedule and the second one is relative robust schedule. Shortly, they try to determine the schedule that minimises the worst-case absolute (or relative) deviation from optimality both of which are NP-hard. They also propose an algorithm for the robust scheduling problem. The result indicated that the robust schedules provide effective hedges and excellent flow time performance.

In another study, O'Donovan, Uzsoy, and McKay (1997) consider predictable scheduling in the single machine environment with the total tardiness criterion under stochastic machine failures. They present a predictive scheduling approach which inserts additional idle times into the schedule to absorb the impacts of breakdowns. They measure the predictability as the completion time deviations between the realised schedule and the predicted schedule. In the experiments they use two different procedures to generate predictive schedules, called ATC(1), and ATC(1)+OSMH. ATC heuristic is developed by Rachamadugu and Morton (1982). It is based on the idea that whenever a machine becomes free, the job with the highest priority index is scheduled first. In ATC(1)+OSMH heuristic a predictable schedule (Sp) is generated using ATC(1) assuming no breakdowns. Then expected breakdown duration is added to completion time of each job i using the equation $(p_i/\lambda)R$, where R is the expected breakdown duration, λ is the rate of breakdown occurance and p_i is the processing time of job i. Their rescheduling methods are ATC(1), ATC(2) and RHS. The only difference between ATC(1) and ATC(2) comes from the calculation of jobs priority indices. The results indicate that, in terms of tardiness scheduling/rescheduling policy ATC(1)/ATC(1) is the best. This schema generates the initial schedule using ATC(1) and then use the same schedule procedure for rescheduling also. In terms of predictability, ATC(1)+OSMH/ RHS is the best policy. In this policy, the initial schedule is generated using ATC(1)+OSMH, and the remaining jobs are simply right shifted at rescheduling points.

Later, Uzsoy and Mehta (1999) study single machine problem subject to machine breakdowns with the objective of minimising deviations between the realised schedule and the predicted schedule while trying to keep the maximum lateness below a certain level. Their objective is to develop predictive schedules, which can absorb disruptions without affecting planned activities while maintaining high shop floor performance. Deviations between the realised schedule and the schedule predicted can be reduced by inserting additional idle times into the schedule predicted. While the predicted schedule is determined using Carlier Algorithm, Uzsoy and Mehta developed three reactive heuristics (called OSMH, LPT and LPTH). These heuristics minimise the deviations between the predictive schedule and the realised one, while maintaining the performance of the schedule in an acceptable level. Their approach can be formulated mathematically as follows"

Min $z = \sum DVi$,

given that

$f(S^r) < d$, where $S^r = g(S^p, E)$, $DVi = |Ci^r - Ci^p|$. Here,

E = environmental factors (ie machine breakdowns).

Ci^r = completion time of job i in realised schedule,

Ci^p = completion time of job i in predictive schedule.

f(S) = performance measure value of S.

d = a constant number.

The results indicate that the insertion of idle times in a controlled manner provides significant improvement in predictability at the expense of some degradation in realised schedule. Thus, predictable schedules are more robust to errors where machine breakdown occurs.

2.2 Reactive Scheduling in Job Shops

Wu and Storer (1994) study the job shop problem subject to machine breakdown. They defined robustness as follows:

$$Z(S) = r.E[M(S^r)] + (1-r).E[R(S)]$$

where $R(S) = M(S^r)-M(S^p)$. Here, $M(S^r)$ is the measure of performance in the realised schedule(S^r), and $M(S^p)$ is the measure of performance in predicted schedule(S^r). Thus, R(S) becomes the difference between these two schedules. Then they develop robust schedules using surrogate measures for expected delay and expected makespan after disruptions. The authors use the "Genetic Algorithm (GA)" whose objective function minimises the robustness measure. Then, they compare the results of the schedules released for different r values including machine breakdowns, with the schedule generated simply by setting objective function equal to deterministic makespan before disruptions. This study can be summarised as follows:

$$\text{Min } R(S) = r.E[f(S^r)] + (1-r).E[\delta(S)]$$

f(S) = performance measure value of S.

$\delta(S) = f(S^r) - f(S^p)$, using surrogate measures , the objective function becomes,

$$\text{Min } R(S) = r.SM + (1-r).DM,$$

where,

SM = surrogate measure of $E[f(S^r)]$

DM = surrogate measure of $E[\delta(S)] = (\sum Si)/N_f$

Si = | (latest start time of job i) - (earliest start time of job i corresponding to schedule S)|

$SM = f(S^p) - DM$.

N_f = the set of operations to be processed on fallible machines.

Their claim is that, there exist a trade off between the makespan and delay as the value for r changes. Moreover, considering both criteria is an attractive alternative for evaluating the suitability of schedules (Wu and Storer, 1994).

Apart from the study on predictable scheduling on the single machine, Mehta and Uzsoy (1998) also analyse the job shop scheduling problem subject to machine breakdowns with the objective of minimising deviations between the realised schedule and the predicted schedule, while trying to keep the maximum lateness below a certain level. They presented a predictable scheduling approach for a job shop with random machine breakdowns. The results indicated that predictable scheduling provides significant improvement in predictability at the expense of little degradation in realised schedule Lmax. In both studies, they conclude that, the heuristics OSMH and LPH(π=0,75) are the best one in terms of predictability without a little degregation in realised schedule. LPH(π) heuristics is a Linear Programming based heuristic which inserts additional idle times to predictive schedule. However, the amount of inserted additional time is constrained by controlling the realised schedule Lmax degradation using the value π. The authors used the surrogate measures of predictability and then optimise the objective function using these surrogate measures without yielding an unacceptable level of performance measure. Here, the objective is, to permit some decrease in the performance in order to increase the predictability of the schedule. They observe that similar results for both single machine and job shop are obtained. This means that studying single machine model provides insights that can be used to extend the approach to more complex, multi machine environments. Their results indicate that OSMH and LPH(0,75) are the best in terms of predictability.

Sabuncuoglu and Bayiz (2000) study the reactive scheduling problems and measure the effect of shop floor configurations (system size and load allocation) on the performance of scheduling methods (off-line and on-line scheduling methods) under the performance criteria makespan and mean tardiness. In the first part of their study, they compare the beam search based algorithm for the job shop problem with other well known algorithms including problems generated by Lawrance (1984), Adams (1988), and Applegate and Cook (1990). In their second part of the study, they study on the different reactive policies such as partial scheduling versus generating the full schedule, etc. Their computational experiments indicated that beam search is very promising heuristic for the job shop problems. Also, they conclude that partial scheduling with optimisation based scheduling algorithms can be a very practical tool in a highly dynamic and stochastic environment.

2.3 Reactive Scheduling in the FMS Environment

Akturk and Gorgulu (1997) consider the rescheduling of the modified flow shop in case of a machine breakdown. A modified flow shop falls between a job shop and a flow shop where parts can enter the system at one of the several machines, can progress through by a limited number of paths and can exit the system at one of the several machines. The scheduling strategy assumes that a preschedule has been constructed and followed until a single machine breakdown, which is not known priori, occurs. Then, they reschedule to match up with the preschedule at some point in the future. The rescheduling attempt begins with the determination of a match-up point on each machine so that time interval to be scheduled is determined. The authors approach to this problem heuristically by decomposing the rescheduling problem into three parts that are the scheduling of down machine, scheduling of the machines in the upward direction of the down machine, scheduling of the ones in the downward direction of down machine. If the resulting schedule is not feasible, then the match-up point is changed to enlarge the set of jobs that are rescheduled.

In another study, Sabuncuoglu and Karabuk (1999) investigate the scheduling rescheduling problem in an FMS environment. They begin by proposing a filtered beam search algorithm for the FMS environment. Then, the authors propose several reactive scheduling policies in response to machine breakdowns and processing time variations. Both off-line and on-line scheduling algorithms are compared under various experimental conditions. The performance of the system is measured for mean tardiness and makespan criteria. Their computational experiments indicate that the beam search based off-line algorithm performs better than on-line machine and AGV scheduling rules under all experimental conditions for the makespan, mean flow and mean tardiness criteria. Their experimental results also indicate that it is not always beneficial to reschedule the operations in response to every unexpected event. They conclude that the periodic response with an appropriate period length can be effective to cope with the interruptions.

2.4 Observations

As discussed in the previous section, scheduling systems consist of two key elements: schedule generation and reaction to events. We know from the previous experiences that the reactive part is very important for the successful implementation of scheduling systems. However, the published literature deals mostly with the schedule generation part. In the previous section we analysed the reactive scheduling literature. In order to provide an organised presentation of those studies on reactive studies, we use a classification scheme similar to Sabuncuoglu and Bayiz (2000). According to this classification schema, there are three main factors: *environment, schedule generation* and *implementation,* which define the characteristics of the problems. According to the *environment* factor, there are shop flor type, job arrival information and source of stochasticity attributes. Under *schedule generation*, we specify the method to generate schedules and the objective function of the problem. Finally, by the *implementation factor*, we define when and how the reactive scheduling policies are employed (see Table 2.1). From the literature review, we can make the following observations:

- As the number of reaction increases the system nervousness also increases (Sabuncuoglu and Bayiz, 2000).
- After a certain number of rescheduling, the improvement in the system performance is insignificant (Church and Uzsoy, 1992).
- The insertion of idle times in a controlled manner provides significant improvement in predictability at the expense of some degradation in realised schedule. Thus, predictable schedules are more robust to errors where machine breakdowns occur (Uzsoy and Mehta, 1999).

In the scheduling literature, most of the studies deal with the scheduling generation techniques. We do not know how the reactive scheduling performances are affected by the system stochasticity level, (machine breakdown, process time variation) in a dynamic environment. Also, we do not know whether all conclusions drawn from the static case are valid for the dynamic case. In this thesis, we consider scheduling problems in a dynamic flexible manufacturing environment. Specifically, we develop reactive scheduling policies and test their performances in dynamic and stochastic environment. We also compare their performance with on-line and off-line scheduling schemes.

Table 2.1. Classification of the papers in reactive scheduling

	ENVIRONMENT			SCHEDULE GENERATION		IMPLEMENTATION	
Author	**Shop Floor**	**Job Arrival**	**Stochacticity**	**Method**	**Objective Function**	**When**	**How**
Church&Uzsoy (1992)	Single Machine	Dynamic	No	EDD& EDS	Lmax	Periodic&Event Driven(urgent jobs)	Full New Schedule
Daniels&Kouvelis (1995)	Single Machine	Static	Proc. Time var. (no m/c breakdown	Branch and Bound	Robust Schedule	-	-
O'Donovan, Uzsoy & McKay(1997)	Single Machine	Dynamic	Machine Breakdown	Heuristics	Mean Tardiness	Event Driven (MB)	Full New Schedule
Sabuncuoglu & Bayiz (2000)	Job Shop	Static	Machine Breakdown	Filtered Beam Search	Makespan Mean Tardiness	Event Driven Periodic	Full, Partial Schedule
Sabuncuoglu& Karabuk (1999)	FMS	Static	Machine Breakdown Proc. Time Var.	Filtered Beam Search	Makespan& Mean Tardiness	Event Driven	Full, Partial Schedule
Uzsoy & Mehta (1998)	Job Shop	Dynamic	Machine Breakdown	Shifting Bottleneck & Heuristics	Lmax	Event Driven (MB)	Full New Schedule
Uzsoy & Mehta (1999)	Single Machine	Dynamic	Machine Breakdown	Carlier's Heuristics	Lmax	Event Driven (MB)	Full New Schedule
Wu &Storer (1992)	Single Machine	Static	Machine Breakdown	Carlier's Heuristic	Makespan& Expected Delay	Event Driven (MB)	Full New Schedule
Wu & Storer (1994)	Job Shop	Static	Machine Breakdown	Genetic Algorithm	Makespan& Expected Delay	Event Driven (MB)	Full New Schedule

CHAPTER 3

SCHEDULING SYSTEM

In this chapter, we discuss the scheduling system developed for FMS. The proposed system has two major components: 1) Schedule generation module and 2) simulation environment. The schedule generation module consists of both off-line and on-line schedule generation mechanisms. The off-line algorithm is based on the beam search methodology, which generates partial schedules as well as full schedules. The simulation module is developed to create a manufacturing environment so that, schedules can be evaluated in a simulated environment. In the following paragraphs, we will give the background information on simulation based scheduling systems. We will then describe the detailed structure of the proposed scheduling system.

3.1 BACKGROUND INFORMATION ABOUT SIMULATION AND SCHEDULING

From current FMS practice, simulation is seen as one of the most frequently used OR tool. The increased use of simulation is due to the growing need for solving complex problems in manufacturing. Especially, the ability of simulation models to capture necessary details of dynamic and complex systems makes simulation the most used OR tool.

Simulation applications can be classified into stand-alone applications and hybrid applications. In the stand-alone application case, which accounts for the majority of simulation applications, a simulation model is used as a test-bed for evaluating different design alternatives or operational policies without disturbing the actual system. In a typical situation, long and multiple runs are taken from the simulation model and its results are analysed by statistical methods. This can be called as an off-line use of simulation because there is no real time communication between the simulation model and the system elements. In general, the off-line use of simulation gives an overall picture about the system being simulated. In the second case, there are hybrid applications of simulation with other scientific tools such as expert systems (ES)/artificial intelligence (AI) and analytical techniques. These hybrid systems are usually developed for real time operation and control of the manufacturing systems. The simulation model discussed in this chapter has also several on-line capabilities. Hybrid model combines the powers of its constituting elements to solve much larger and complex problems with reduced computational efforts.

Scheduling problems become complicated by the dynamic and stochastic nature of manufacturing environment in which schedules must also be updated frequently over time. Traditional approaches (scheduling algorithms and math programming) may not be sufficient in dealing with these problems. In this chapter, such a hybrid approach in which both simulation and analytical model utilised, is described.

It can be observed that the majority of simulation applications to scheduling problems are in the form of testing on-line scheduling policies. Simulation of off-line scheduling methods has not received considerable attention from the literature. This is partly due to difficulty in applying simulation to the off-line generated schedules in a dynamic and stochastic manufacturing environment. Here, we describe a simulation model that implements both on-line and off-line scheduling methods. The proposed model also enables to compare a wide range of reactive scheduling policies under different environmental conditions.

3.2 THE ELEMENTS OF PROPOSED SCHEDULING SYSTEM

The simulation-based scheduling system consists of three major components: scheduler (scheduling module), simulation model, and controller (Figure 3.1). Scheduler is responsible for making all scheduling decisions. Given the system status and other relevant data (i.e. on-line, off-line) it generates a partial or complete schedule. The scheduling module is based on filtered beam-search technique that generates schedules by considering machines, AGVs, finite buffer capacities, sequence and routing flexibilites, etc. A deadlock resolution mechanism is also embedded in the algorithm.

Simulation model uses two sets of input data: system related data and values of environmental parameters. System related data consist of physical description of the manufacturing system (i.e. number of machines, number and speed of AGVs). Arrival rate of jobs, parameters of stochastic events (i.e. machine breakdown rate), part types, machine and part flexibilities constitute the environmental parameters. In the simulation model, machining subsystem, movement of AGVs and in-process storage capacity are represented in detail. The main task of the simulation model is to implement the scheduling decisions which are made by the scheduler. When an on-line scheduling policy is implemented, a resource triggers the controller upon completing a task which invokes the scheduler. The scheduler makes a decision by applying some scheduling rules and passes the final decision to the controller. Then the controller sends this schedule to the simulation model for execution.

The control module examines the state of the system at every discrete event that occurs in the simulation model and provides appropriate course of action to be executed by the simulation model. The control module has the following tasks: Keep up with the machine and AGV sequence in off-line mode, avoid and resolve deadlock

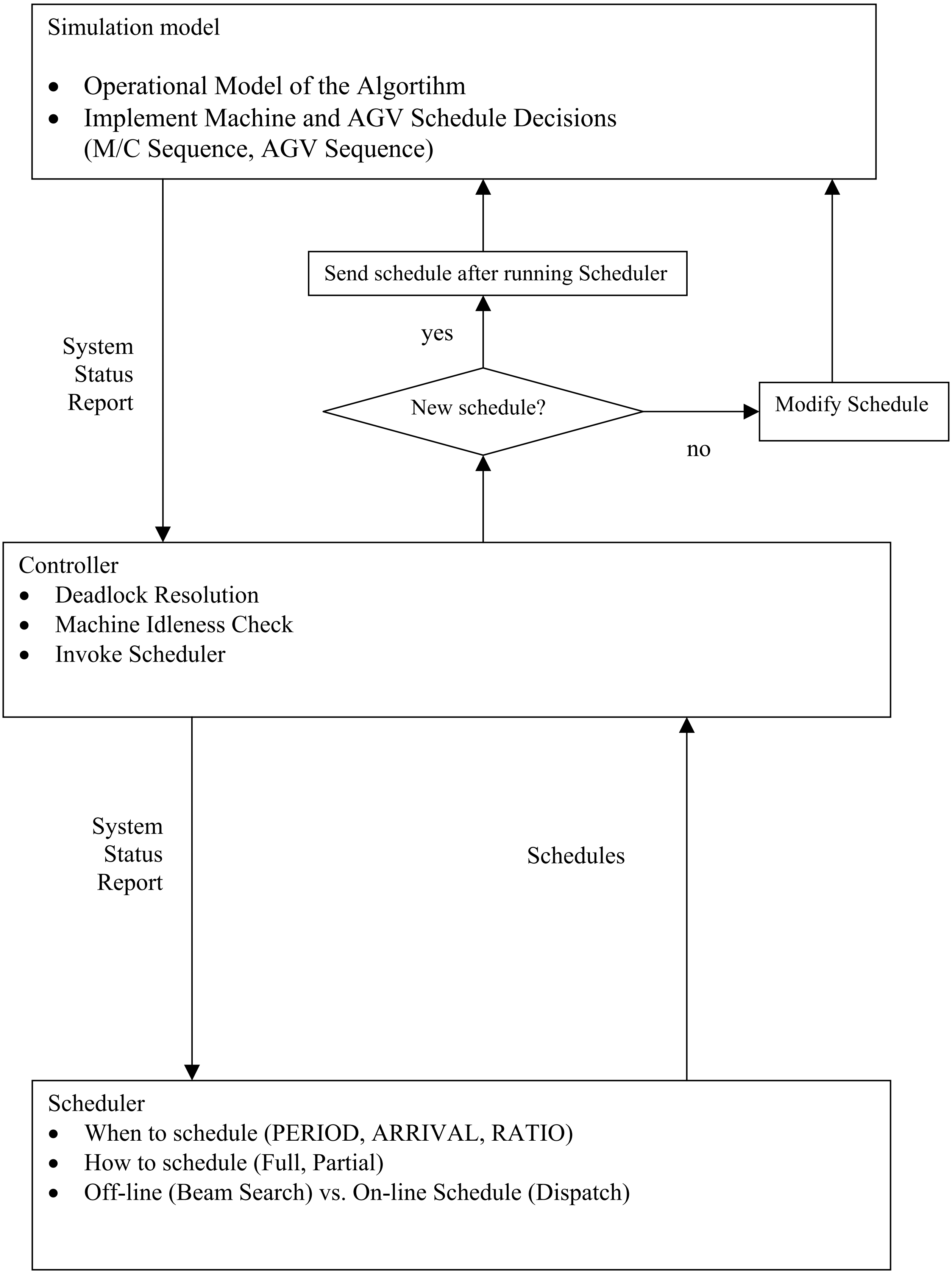


Figure 3.1. A simulation based scheduling system.

situations, implement scheduling policies.

The objective in simulating an off-line schedule is to observe its results in a stochastic and dynamic environment. However, it is not easy to follow the exact start and completion times imposed by the off-line schedule in a dynamic and stochastic environment. When this is not possible, machine processing sequences and AGV move sequences are tried to be followed as close as possible to the original schedule. In most of the manufacturing systems, in-storage capacity is limited. Hence, there is always a possibility for blocking (and locking) in the system due to finite capacities. This necessitates the use of effective control policies to avoid blocking of material movement in the system.

As the third task, the controller is responsible for implementation of scheduling policies by considering the environmental conditions over time. In order to accomplish this, the controller must either be supplied with the appropriate control policy or must simulate alternative policies and choose one according to the simulation results. The first case is encountered in off-line use of simulation, whereas the second stands for on-line use. In the second case, simulation is also used to evaluate different policies at decision points. This method has the advantage of being more adaptive to the dynamically changing manufacturing environment.

The proposed simulation based system is coded using a general purpose programming language (i.e., C language) and implemented in unix environment. From modeling point of view, simulation languages provide a higher level of abstraction to build a model. Although this helps in constructing the model easily and quickly, it also brings restriction. In most cases the control logic of the simulation model can not be implemented with the routines supplied by the simulation package. This is the most crucial part of a simulation model because simulation is mostly used to evaluate different control policies. From implementation point of view, general purpose languages produce faster and compact executable codes than simulation languages. To give a specific example, the simulation language SIMAN produces at least 1300Kbytes of executable code. On the other hand, our proposed system contains approximately 7000 lines of computer code (including all scheduling algorithms) and the size of the executable, when compiled using the C complier with the optimization flag of the compiler set, is 200 Kbytes.

3.3 SCHEDULE GENERATION MODULE

In the literature the solution approaches for the scheduling problems can be classified in two headings: exact solution methods and heuristic procedures. The exact solution approaches formulate the problem as an optimisation problem and then solve it using an exact algorithm. Unfortunately, inherent intractability of scheduling problems make heuristic procedures attractive alternatives. The scheduling algorithm proposed in this paper, is a heuristic based on the filtered beam search technique. This search method is an approximate branch and bound method in which the solution space is explored for the best solution by heuristics that examine a certain number of promising paths, permanently pruning the rest. Since a large part of the tree is pruned off to obtain a solution, its running time is polynomial in the size of the problems. (Sabuncuoglu and Karabuk, 1998; Sabuncuoglu and Bayiz, 1999).

The solution space is represented as a decision tree where each node corresponds to a scheduling decision to be made and each unique path from the root node to any particular node defines a partial solution associated with that node. Leaf node at the end of the tree specifies complete solutions. In the proposed method, the search tree is constructed in such a way that various system resources, their capacities and flexibilities are taken in to account at each layer.

In the filtered beam search, only a certain number of nodes (filterwidth) are sprouted, others are filtered out using a local evaluation function (can also be called *one-step priority evaluation function*). These remaining nodes are then evaluated by a global evaluation function (can also be called *total cost evaluation function*) and the ones found most promising are added to the partial solution. This procedure is repeated on a certain number of parallel paths (beamwidth). Hence, the number of solutions saved at any level of the tree is equal to the size of beamwidth. In contrast to global evaluation function, the local evaluation function typically has a more local view (Ow, Morton, 1988). Thus, the local evaluation function is quick but may discard good solutions. On the other hand, global evaluation function is more accurate but computationally more expensive. The values of filterwidth and beamwidth are usually determined empirically. In our study, we used the filterwidth of 5 and the beamwidth of 3. Other algorithmic details can be found in Sabuncuoglu and Karabuk (1998 and 1999).

3.4 SYSTEM CONSIDERATIONS AND EXPERIMENTAL CONDITIONS

A classical FMS is used in this study. The FMS environment we study consist of six machines each with buffer capacity, and one load/unload (L/U) station. Parts are transferred by three AGV's in the system. Parts enter and leave the system through the L/U station. This station is also used as a central buffer area when blocking occurs in the system. Five jobs are assumed to be ready at the L/U station at time zero. Also, jobs are arrived to the L/U station exponentially with mean 55. Each job has either 5 or 6 operations with equal probability and each operation is assigned to a different machine. Hence machine loads are kept nearly equal. Operation times are drawn from a 2-Erlang distribution with mean 55.

The performance of the proposed algorithm is measured under various operating conditions with the following experimental factors: 1) buffer capacity (Q), 2) sequence flexibility (SF), 3) routing flexibility (RF), 4) tardiness factor (TF), 5) process time variation (PV), 6) machine breakdown level (e). Among these factors buffer level, sequence flexibility, routing flexibility, and tardiness factor are called *internal factors*. The other factors (process time variation and machine breakdown level) are called *external factors*. For each of the above factors two levels (low and high) are considered in the experiments. The low and high levels for internal factors are given in Table 3.1. The queue capacity of the machines is set to 10 and 100, corresponding to finite and infinite values.

Table 3.1. Internal factors and their levels

Factor	Low	High
Queue Capacity (Q)	10	100
Routing Flexibility (RF)	1	2
Sequence Flexibility (SF)	0	1
Tardiness Factor (TF)	0.75-0.85	0.75-0.85

Routing flexibility (RF) is defined as the average number of machines on which a particular operation can be processed. The value is set to 1 and 2 for low and high levels of this factor, respectively. We assume that the first assigned machine is the ideal machine with the least processing time. The processing time on the alternative machine is computed by adding a random number to the processing time of the operation on the ideal machine. This random number comes from a uniform distribution with a mean of half the processing time of the operation on the ideal machine.

Sequence flexibility (SF) is an indicator of precedence relationships between operations of the job. Specifically, operations of a job are viewed as nodes on an acyclic graph. The density of precedence arcs on this graph determines the degree of sequence flexibility. Its equation is as follows:

SFM = 1.0 – (2*all precedence arcs)/(n*(n-1)),

where n is the number of operations. The SFM value ranges between 0.0 and 1.0. The closer the SFM to 1.0, the higher the sequence flexibility a job posseses. In our experiments, SFM is set to 0.0 and 1.0 for low and high sequence flexibilites.

Due dates are based on total work content (TWK) rule. According to this rule, due date of a job is determined by multiplying total work content of a job by a constant multiplier so that the desired TF value is achieved. In our study, they are assigned such that the tardiness factor (TF) is approximately fixed at 80%.

Performance of the algorithm is tested for mean flowtime criteria. The local evaluation function for the mean flowtime case is LWRK (least work remaining). Also, we use LWRK rule for the on-line scheduling. The proposed scheduling system (scheduling mechanism and simulation model) was initially developed by Sabuncuoglu and Karabuk (1998, 1999). Later it is modified to obtain a working version by this study.

3.5 RESCHEDULING POLICIES CONSIDERED IN THIS STUDY

We classify rescheduling policies in terms of two decisions: *when to schedule* and *how to schedule.* In the former case, we decide on scheduling points in time while in the later case we decide how to schedule the system at these time points. In other words, *when to schedule* determines the time between two consecutive scheduling points, while *how to schedule* determines a way of generating a feasible schedule. In this context, we call *full scheduling* when all of the jobs available are to be scheduled. We call *partial scheduling* if a subset of available jobs is to be scheduled.

In terms of *when to schedule,* we can identify three policies: fixed sequencing, periodic review, continuous review. In the *fixed sequencing* approach, a schedule is generated only once at the beginning of the scheduling period, and it is not later updated other than simple time-shifting operations in the Gannt chart. It is assumed that the system recovers from the negative effects of interruptions (breakdowns, new job arrivals, due date changes, etc.) in the system by itself (i.e., we assume that there is enough slack in the system that it can cope with the negative impacts of unexpected events).

According to the *periodic review* policy, the system is monitored periodically and rescheduling is invoked at the beginning of time points. As discussed in Sabuncuoglu and Karabuk (1999), the periodic policy can be implemented in two alternative ways: 1) Fixed time interval and 2) Variable time interval. According to the *fixed time* interval method, the review periods are equally spaced points in time (i.e., at the beginning of every shift, day, week, etc.). According to the *variable time* interval method, time between two scheduling points is not constant, but rather depends on the percentages of jobs processed or total processing time realised on all machines in the system (i.e., rescheduling is triggered when the cumulative processing time realised on machines exceeds a certain limit). Thus, *variable time* interval method is more responsive to the state of the system (and the current production rate) than the *fixed time* interval method.

In *continuous review*, the system is monitored continuously and rescheduling is triggered in response to changes in the system (new job arrivals and/or machine breakdowns). In the literature, this policy is also called *event-driven scheduling* policy (Ovacik and Uzsoy, 1992). This policy can be implemented to react to certain number of arrivals or machine breakdowns rather than responding to every arrival or breakdown.

These three policies are listed in Table 3.1. PERIOD corresponds to the periodic review policy. Very large values of PERIOD correspond to the fixed sequencing policy. RATIO implements the variable time increment method in such way that rescheduling is triggered when determined percentage of the scheduled jobs are processed in the system. ARRIVAL implements the continuous review policy.

Table 3.2. When to reschedule policy.

Name of the Method	Policy
PERIOD	-Periodic review with fixed time interval. - Fixed sequencing.
RATIO	-Periodic review with variable time interval.
ARRIVAL	- Continuous review.

CHAPTER 4

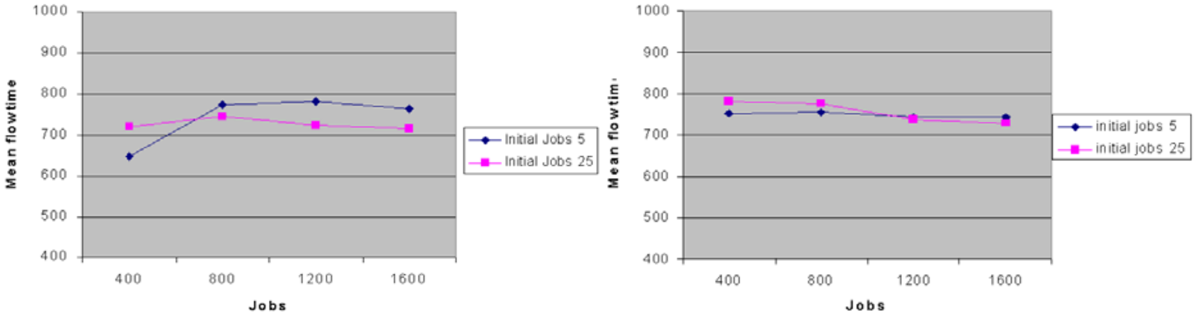
COMPARISON OF RESCHEDULING POLICIES

In this chapter, we present the results of simulation experiments conducted to compare three scheduling schemes: PERIOD, RATIO, and ARRIVAL. In Section 4.1, we give a brief summary of the pilot experiments to set values of some parameters. In Section 4.2, we compare three scheduling policies with dynamic job arrivals (i.e., in a dynamic environment). The other factors, machine breakdowns and processing time variations (i.e, stochastic environment) are considered in Section 4.3.

4.1 PILOT EXPERIMENTS

In our study, we are mainly interested to see the effects of external factors such as dynamic job arrivals, process time variation, and machine breakdowns on scheduling policies and schedule generation schemes (off-line and on-line). In order to keep the computational efforts at a reasonable level, we conduct some pilot experiments to set the values of some internal factors (i.e, queue capacity, sequence flexibility, and routing flexibility).

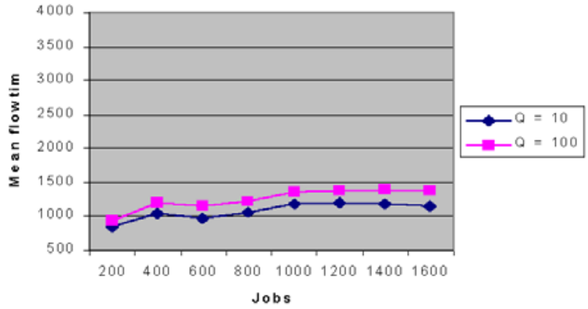
First, we simulate the system with two different initial job populations: 5 and 25. As can be seen in Figure 4.1, size of initial job population does not seem to affect the long term performance of the system because the difference between initial job populations is insignificant and the system reaches steady state at nearly the same times, even in the dispatch rule case (i.e., using LWRK rule -least work remaining). Thus we, begin the simulation with 5 jobs.



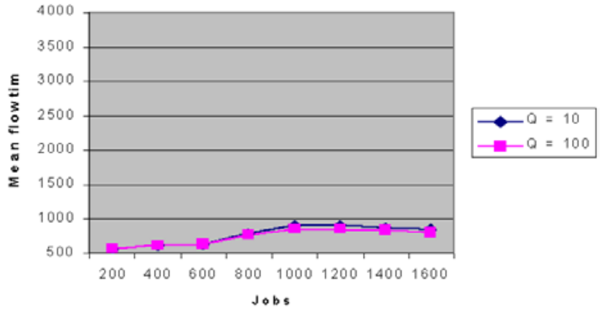
a) Reschedule with the period length 200 of Beam Search b) Dispatch rule

Figure 4.1. Comparison of mean flowtimes for two different initial job populations for high SF and RF.

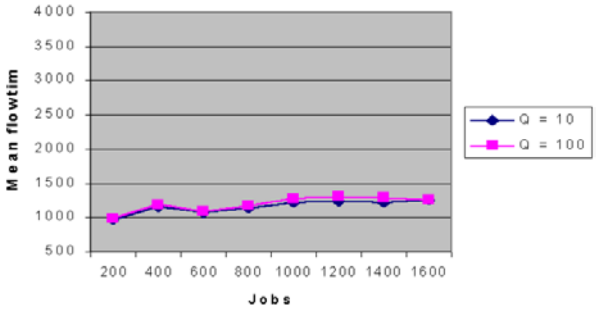
Second, we test different buffer capacities. In our pilot runs, we have observed that the system sometimes experiences a deadlock situation and spends a considerable amount of time to solve this problem when the buffer capacity is less than 10. Hence, we set the finite buffer capacity to 10 to avoid excessive amount of computation times. We set the buffer capacity to 100 to represent very large buffer capacity (i.e., the unlimited or infinite buffer capacity level). We test these two buffer sizes using scheduling policies ARRIVAL_1 (A_1), PERIOD_200 (P_200), RATIO_100 (R_100), and the LWRK dispatch rule (See the details of the results in Table 4.1, and Table 4.10 in Appendix). Here A_1 refers to the ARRIVAL policy with its parameter 1, schedule at every arrival. P_200 refers to the PERIODIC policy with period length of 200, and finally R_100 refers to the RATIO policy with its parameter 100, schedule the jobs when all the jobs scheduled previously are processed. As seen in Figure 4.2, the mean flowtime performance of the system is only slightly improved when we make the buffer capacity too large (i.e, unlimited buffer capacity). This is seen both in the off-line (beam search) algorithm and on-line dispatching rule cases. Hence, we decided to continue with the buffer size 10 in the rest of experiments.



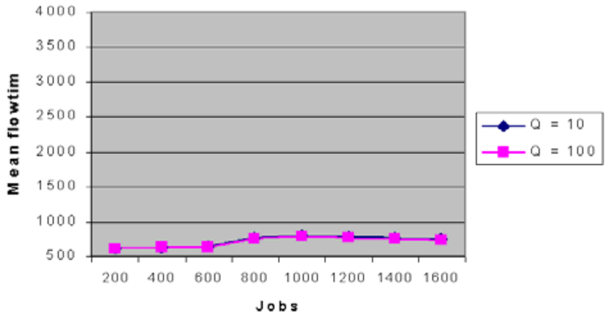
a) A_1 for low SF and RF



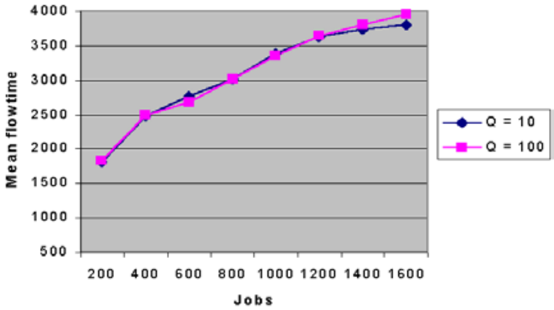
b) A_1 for high SF and RF



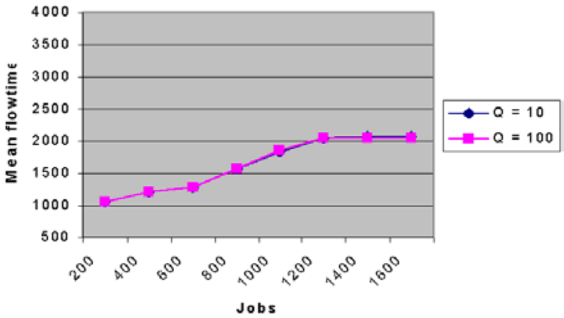
c) P_200 for low SF and RF



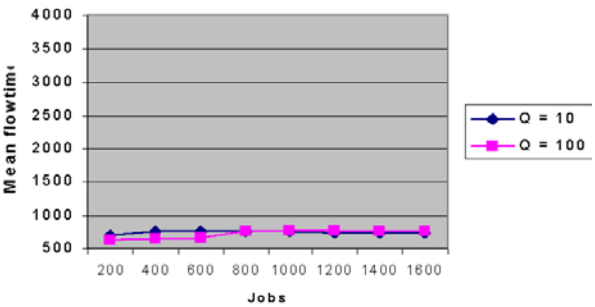
d) P_200 for high SF and RF



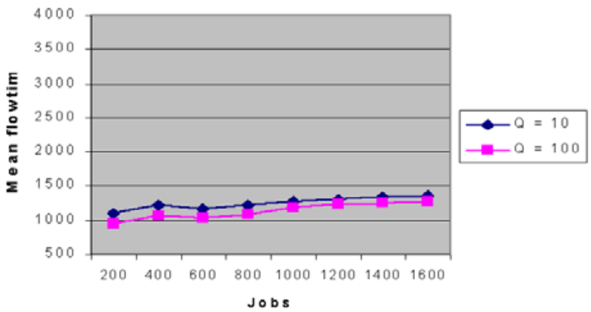
e.) R_100 for low SF and RF



f.) R_100 for high SF and RF.



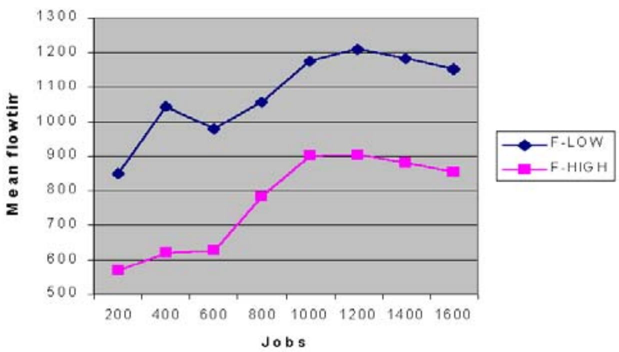
g) Dispatch for low SF and RF



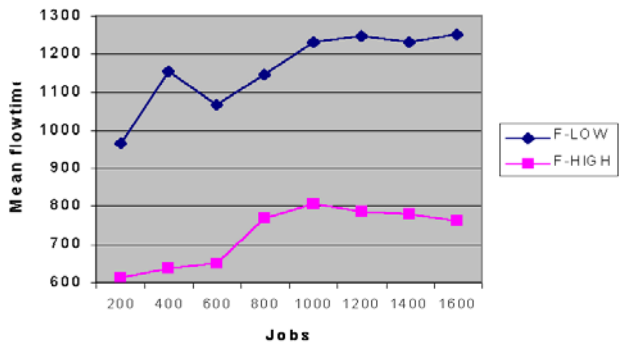
h) Dispatch for high SF and RF

Figure 4.2. Comparison of limited and unlimited queue capacities for different scheduling policies.

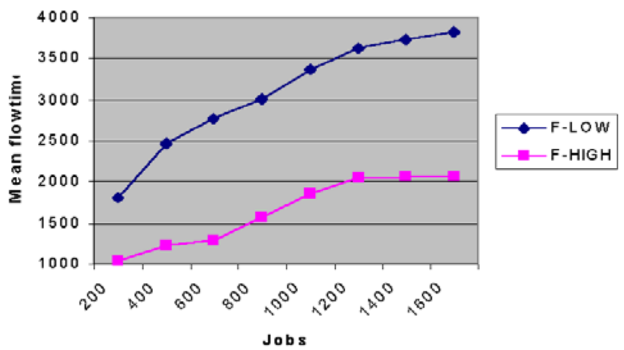
We also examine the effects of sequence flexibility and routing flexibility. Initially, we considered four cases: 1) high sequence flexibility and high routing flexibility, 2) high sequence flexibility and low routing flexibility, 3) low sequence flexibility and high routing flexibility, and 4) low sequence flexibility and low routing flexibility. To save computational time, we take the pilot runs at only the two levels: F-HIGH (routing flexibility is high and sequence flexibility is high) and F-LOW (routing flexibility is low and sequence flexibility is low). As can be seen in Figure 4.3, the performance of the system is substantially affected by the level of flexibilities (this can also be seen in Table 4.2 in Appendix). For that reason, we continue with using both the low and high levels of flexibilities (i.e., F-LOW, and F-HIGH) in the rest of experiments.



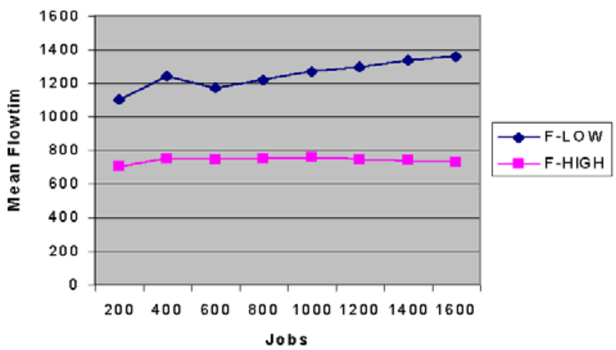
a) A_1



b) P_200



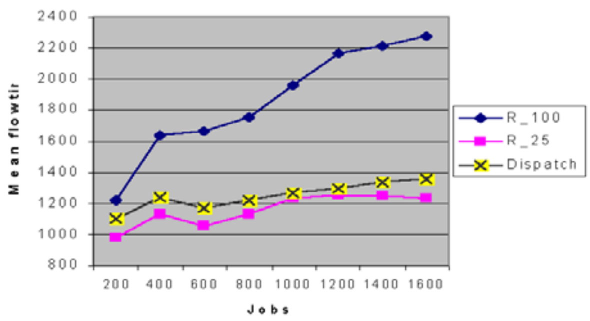
c) R_100



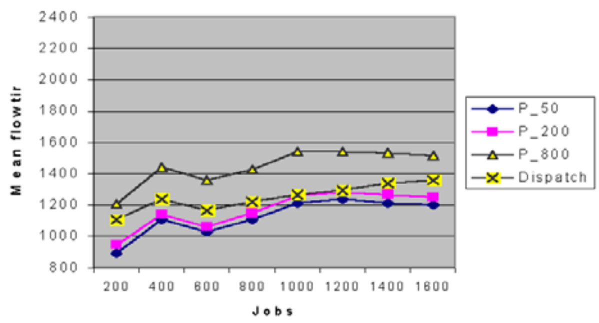
d) Dispatch

Figure 4.3. Comparison of flexibilites for different scheduling policies.

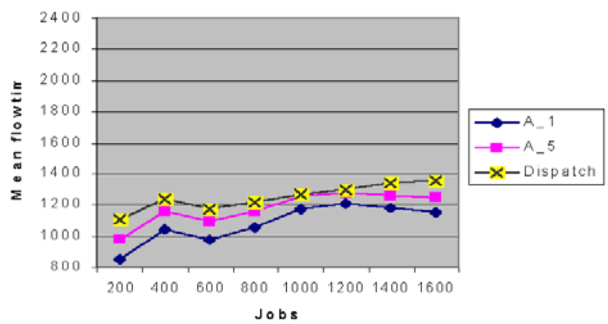
During the pilot runs, we have also noted that the performance of the system is improved as the scheduling frequency increases (See Table 4.3 in Appendix). For example, RATIO_25, which has a higher scheduling frequency (more frequent update) than RATIO_100 yields better performance. Also, ARRIVAL_1 has a better performance than ARRIVAL_5. Similarly, PERIOD_200 yields a better performance than PERIOD_800 (Figure 4.4). This finding, in a dynamic environment, is consistent with the results of the previous studies obtained in a static environment (Sabuncuoglu and Karabuk, 1999; Sabuncuoglu and Bayiz, 2000).



a) R ATIO policy



b) PERIODIC policy



c) ARRIVAL policy

Figure 4.4. Comparison of different scheduling frequencies for F-LOW.

As a result, in the rest of our study, we set queue capacity to 10, tardiness factor between (0.75 - 0.85). Also, we combine the two flexibility levels and define only F-HIGH and F-LOW.

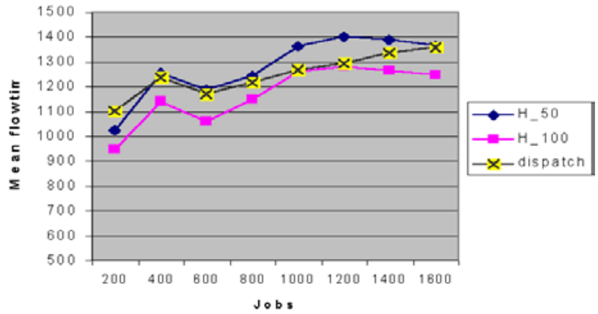
4.2 COMPARISON OF SCHEDULING POLICIES IN A DYNAMIC ENVIRONMENT

In this section, we analyse the scheduling system in a dynamic environment (i.e., dynamic job arrivals). As mentioned earlier, we consider two types of scheduling decisions: *how to schedule* and *when to schedule*. We use three policies (ARRIVAL, PERIODIC, and RATIO) for *when to schedule*, and the two policies (full scheduling and partial scheduling) for *how to schedule.* We will start with the *how to schedule policies.*

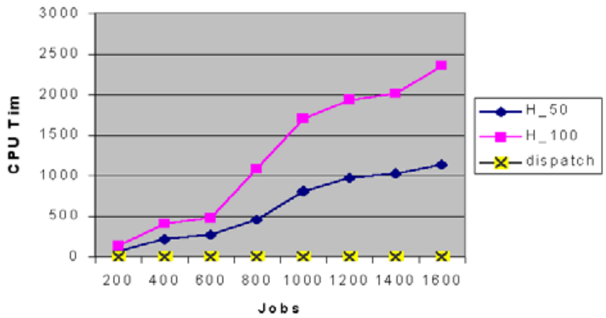
4.2.1. HOW TO SCHEDULE POLICIES

The feature of our beam search algorithm allows us to obtain partial schedules since it generates the schedules in the forward direction. In general, the length of a partial schedule can be defined either by in terms of clock time or percentage of the total jobs or operations to be scheduled. In this study, we use the latter approach (i.e., a certain percentage of the jobs is scheduled at each scheduling point). Thus, we identify two cases: 1) full scheduling (corresponds to 100%), and 2) partial scheduling (corresponds to 50%).

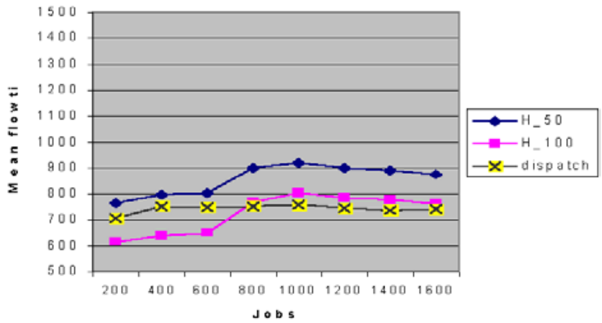
The results of the simulation experiments are given in Table 4.2 in Appendix. In Figure 4.5, the effect of partial scheduling is displayed for the mean flowtime criterion. As expected, full scheduling (100% job scheduled) yields better performance than partial scheduling (50% job rescheduled). This is observed for each *when to schedule* policy. But notice that, it requires considerably higher computational time compared to partial scheduling.



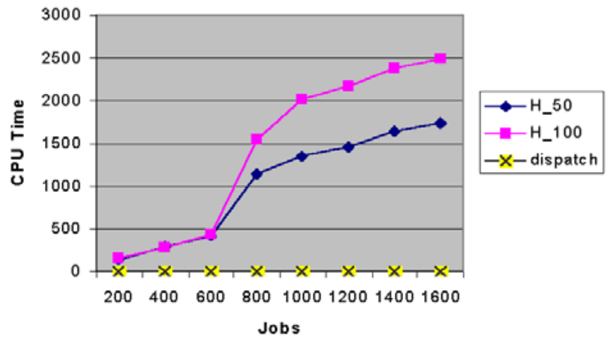
a) P_200, F-LOW.



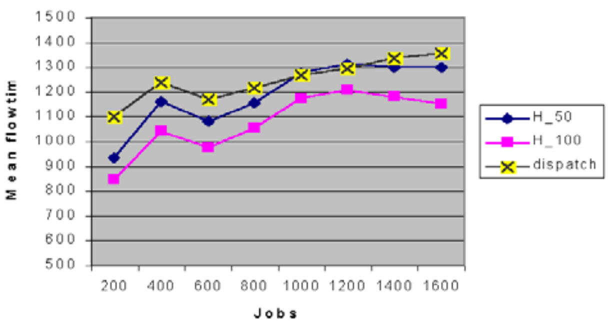
b) P_200, F-LOW



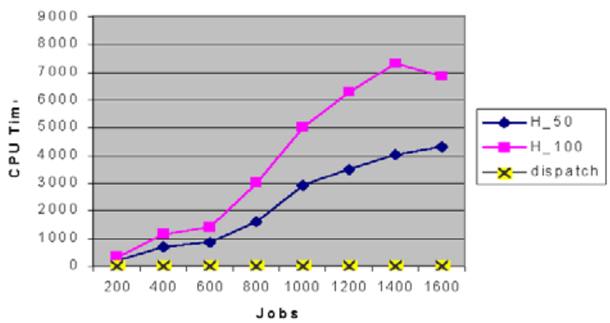
c) P_200, F-HIGH



d) P_200, F-HIGH.

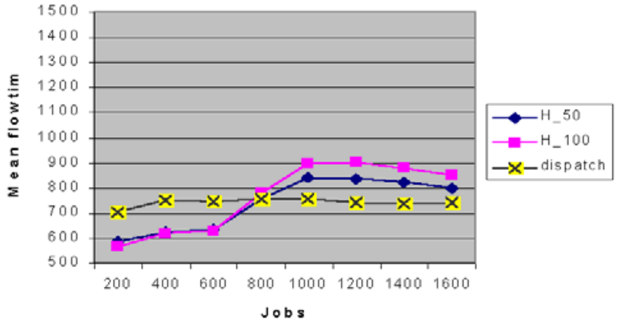


e) A_1, F-LOW

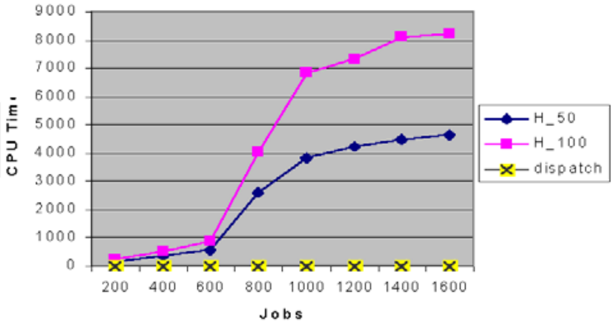


f) A_1, F-HIGH.

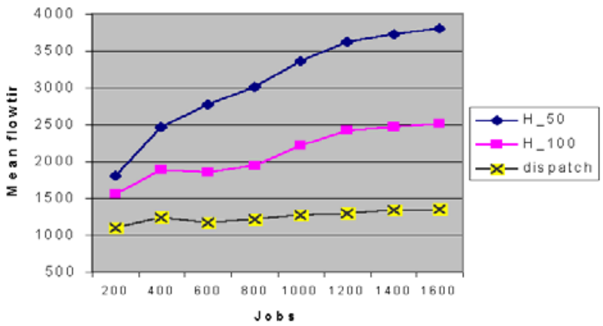
Figure 4.5. Comparison of how to schedule policies.



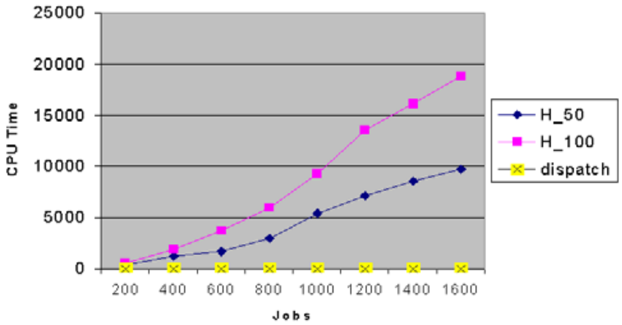
g) A_1, F-HIGH.



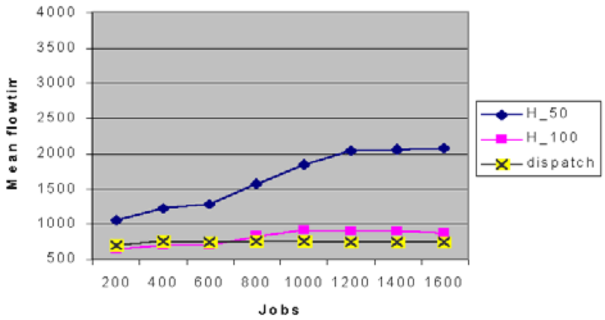
h) A_1, F-HIGH.



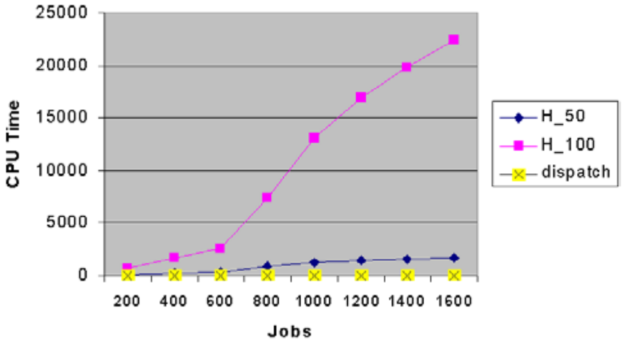
i) R_100, F-LOW.



j) R_100, F-LOW.



k) R_100, F-HIGH.

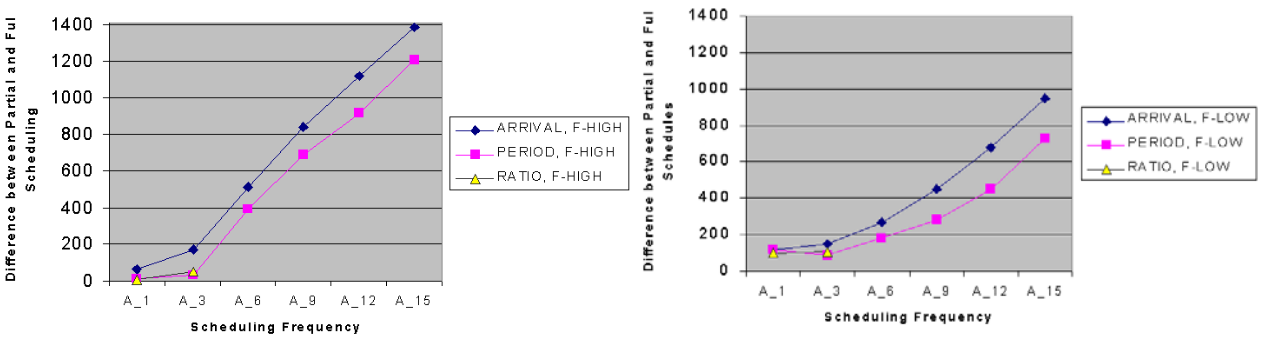


l) R_100, F-HIGH.

Figure 4.5. Comparison of how to schedule policies (Cont'd.).

We further investigate the difference between partial scheduling and full scheduling at various scheduling frequencies (See Table 4.6 in Appendix). Scheduling frequencies are adjusted accordingly for the ARRIVAL policy. In other words, the parameters of the PERIODIC and RATIO policies are adjusted according to the ARRIVAL policy. For that reason, A_x is displayed in the horizontal axis in Figure 4.6.

In general, we observe that the ARRIVAL policy is more affected from scheduling frequency than the RATIO and PERIODIC policies, since it displays the higher envelope in the curves. In our experiments, we could not compare the RATIO policy for the scheduling frequency more than A_3. Because even if we process %100 of the jobs scheduled at the previous scheduling point (i.e., R_100), the scheduling interval can not be greater 180. This means that we can not apply the RATIO policy, which is equivalent to the PERIOD policy 200 or above. For that reason, under the current experimental settings, RATIO is implemented up to the scheduling frequency corresponds to A_3.



a) F-HIGH

b) F-LOW

Figure 4.6. Differences between full and partial schedules for different scheduling frequencies.

In our experiments, we also study the affect of scheduling frequencies on the system performance at two different flexibility levels for the ARRIVAL and PERIODIC policies (Figure 4.7). The RATIO policy is not included in the figures due to the reason started before. The results of the simulation experiments indicate that, as the scheduling frequency decreases, the differences between F-HIGH and F-LOW decrease for partial scheduling (Figure 4.7.c, 4.7.d), whereas it is nearly constant for the full scheduling scheme (Figure 4.7.a, 4.7.b). We also observe that for scheduling frequencies lower than A_12, both F-HIGH and F-LOW show nearly the same performances (Figure 4.7.b, 4.7.d). In our opinion, this is due to the fact that, the search space is much smaller in partial scheduling (as compared to full scheduling) and hence, the algorithm can not get enough opportunities to utilise the flexibility.

We also compare simple dispatch rules with the beam search algorithm for different frequency levels for both full and partial scheduling (Figure 4.7). As can be seen in the Figures 4.7.a and b, the beam search algorithm (with full schedule) yields better performance than the dispatch rule, when the scheduling frequency is more than A_9 (i.e., A_1, and A_6) and the flexibility is low. However, when the partial scheduling is implemented, the algorithm performs better than the simple dispatch rule for the scheduling frequency more than A_3 and the flexibility is low. For the high flexibility case, dispatch rule always performs better than the beam search algorithm when the algorithm is implemented with partial schedule. For the full schedule case, the ARRIVAL policy performs better than the dispatch rule for only A_1 and A_3 scheduling frequencies (Figure 4.7.a). But the PERIODIC scheduling policy always yields worse performance than the dispatch rule.

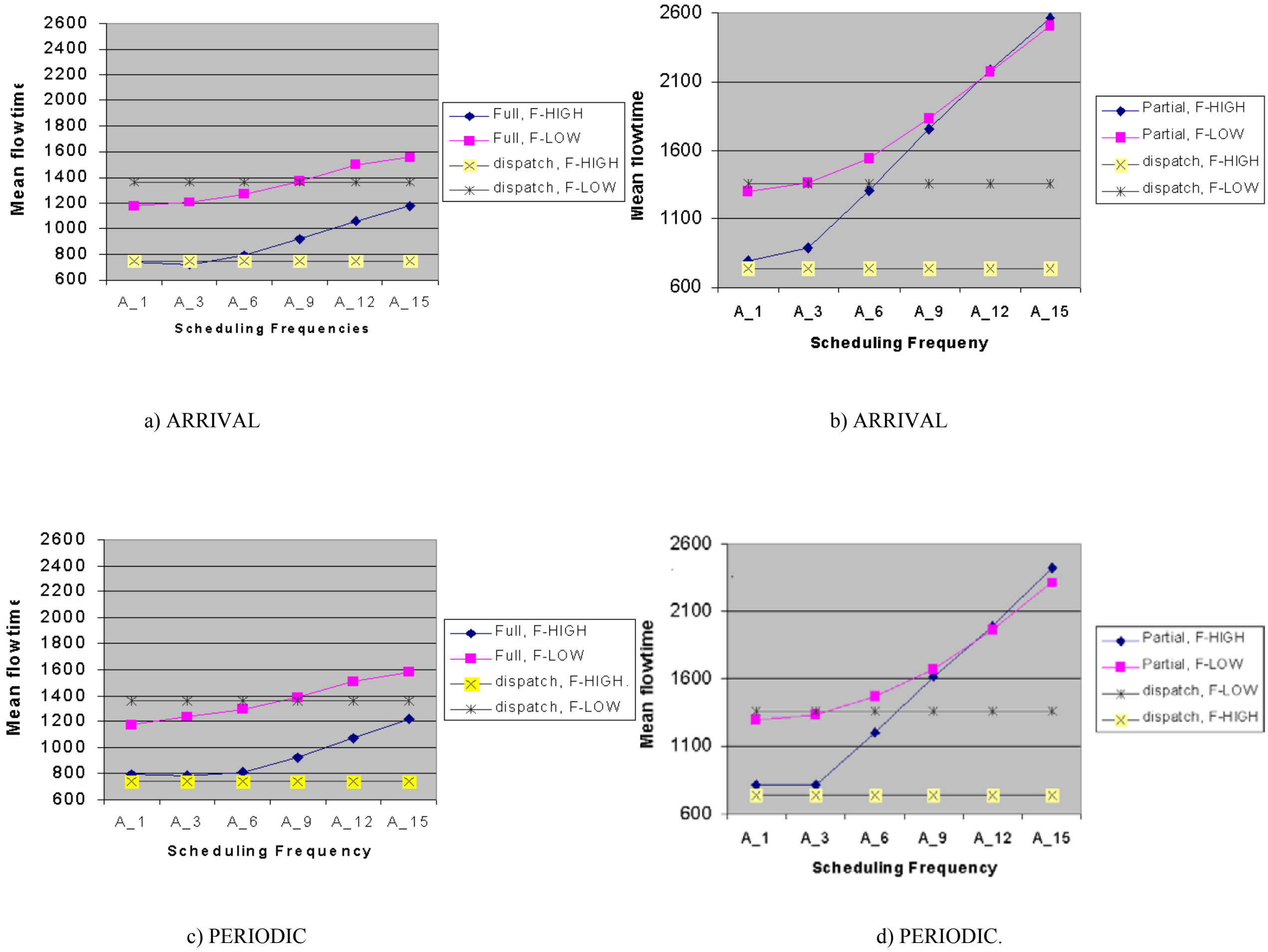


Figure 4.7. Comparison of flexibilities for full and partial schedules.

4.2.2. WHEN TO SCHEDULE POLICIES

As mentioned earlier the , *when to schedule* decision determines on rescheduling points in time. (i.e., the time between two reschedule points). According to this policy the jobs are scheduled either at fixed time intervals or variable time intervals. Recall that PERIOD is the fixed time interval method whereas ARRIVAL and RATIO are variable time interval methods.

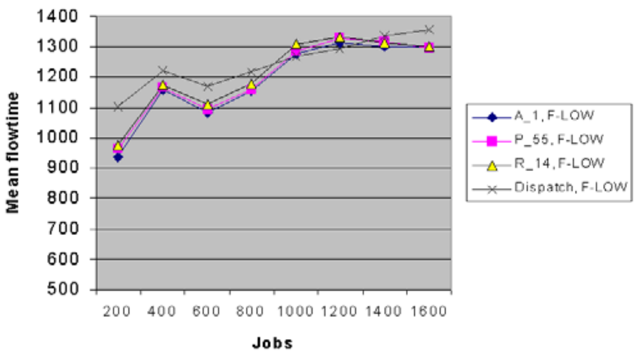
These three methods are compared at various scheduling frequencies. Again the scheduling system is simulated for 1600 jobs. In order to compare the policies on equal basis, we adjust their parameters in such a way that each *when to schedule* policy has approximately the same scheduling frequencies (i.e., number of scheduling points are approximately equal). Specifically, the parameter of ARRIVAL is first set and then the parameters of the RATIO and PERIOD policies are adjusted accordingly. The same type of adjustment is also made for partial scheduling (H_50). The details of the results are given in Tables 4.3 and 4.4 in Appendix.

As seen in Figures 4.8, the RATIO policy is generally better than ARRIVAL and PERIOD. We also observe that the differences between scheduling policies are minimum when the flexibility is low (i.e., F-LOW) and the frequency of scheduling is very high (i.e., A_1 case). This is due to the fact that scheduling policies can not find enough opportunities to improve the system performance when flexibility is LOW. Also, when the scheduling decisions are made so frequently (i.e., A_1 case), the scheduling policies can not show themselves. Because, in the absence of breakdowns and process time variation, the policy respond to nearly every arrival or departure. Hence, they do not display different performances. Note that this observation is valid both in full and partial scheduling.

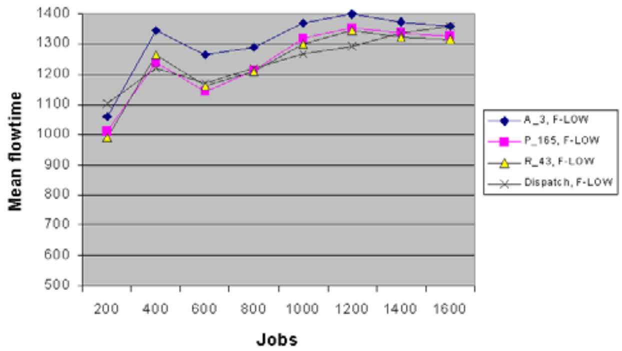
The better performance of the RATIO policy can be attributed to the fact that it is somehow related to the production rate (output process of the system). Note that this policy relies on the production capability as well as demand (or arrival) information.

The performance of the PERIODIC and ARRIVAL policies are quite mixed. In general, ARRIVAL policy is better when used with full scheduling whereas PERIODIC is better when the partial scheduling is implemented.

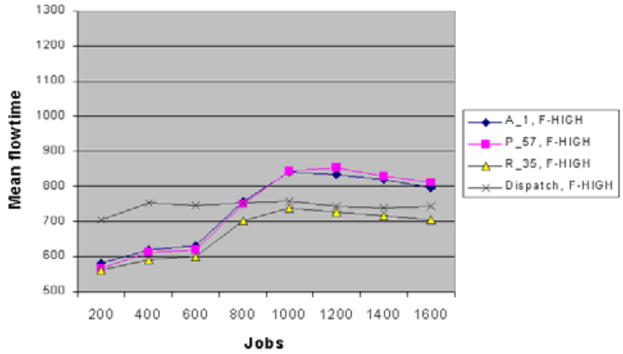
In order to understand this mixed behaviour, we further run the simulation experiments for these two policies at various values of partial scheduling levels. The results are summarised in Figure 4.9 and Table 4.7. As can also be seen in Figure 4.9,



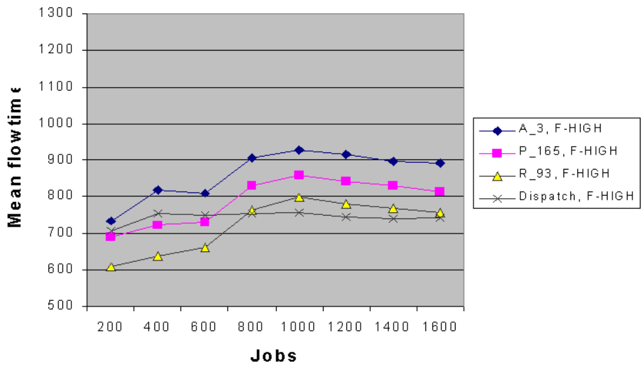
a) Partial Schedule at low flexibility



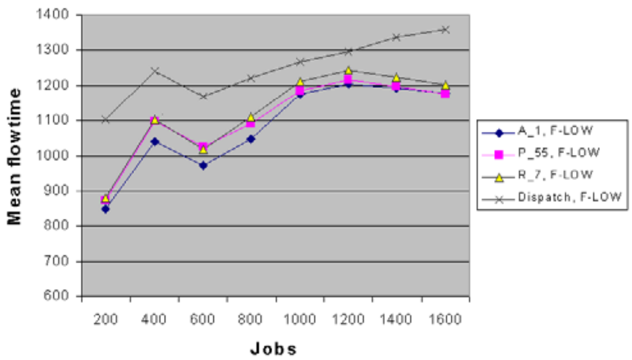
b) Partial Schedule at low flexibility



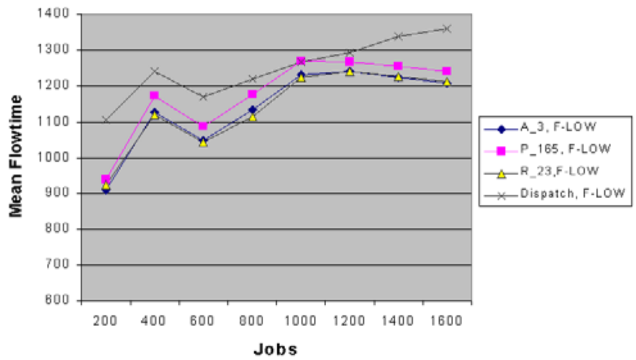
c) Partial Schedule at high flexibility



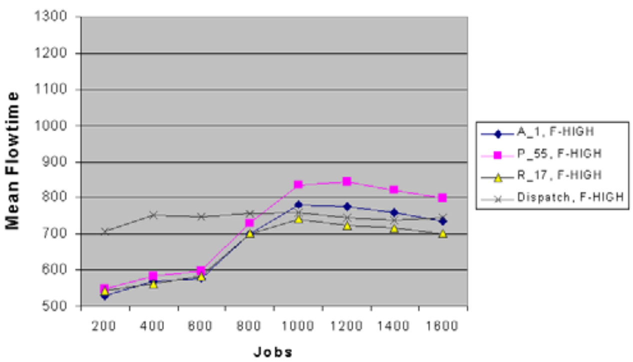
d) Partial Schedule at high flexibility



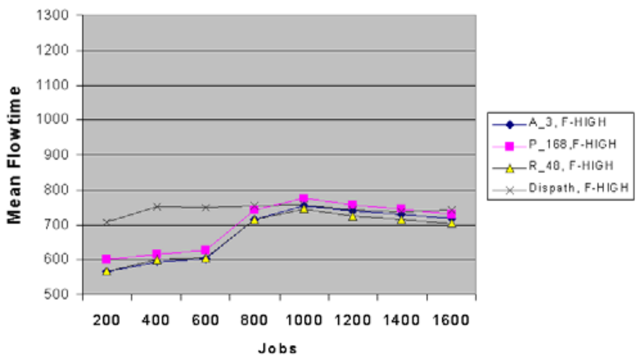
e) Full Schedule at low flexibility



f) Full Schedule at low flexibility



g) Full Schedule at high flexibility



h) Full Schedule at high flexibility

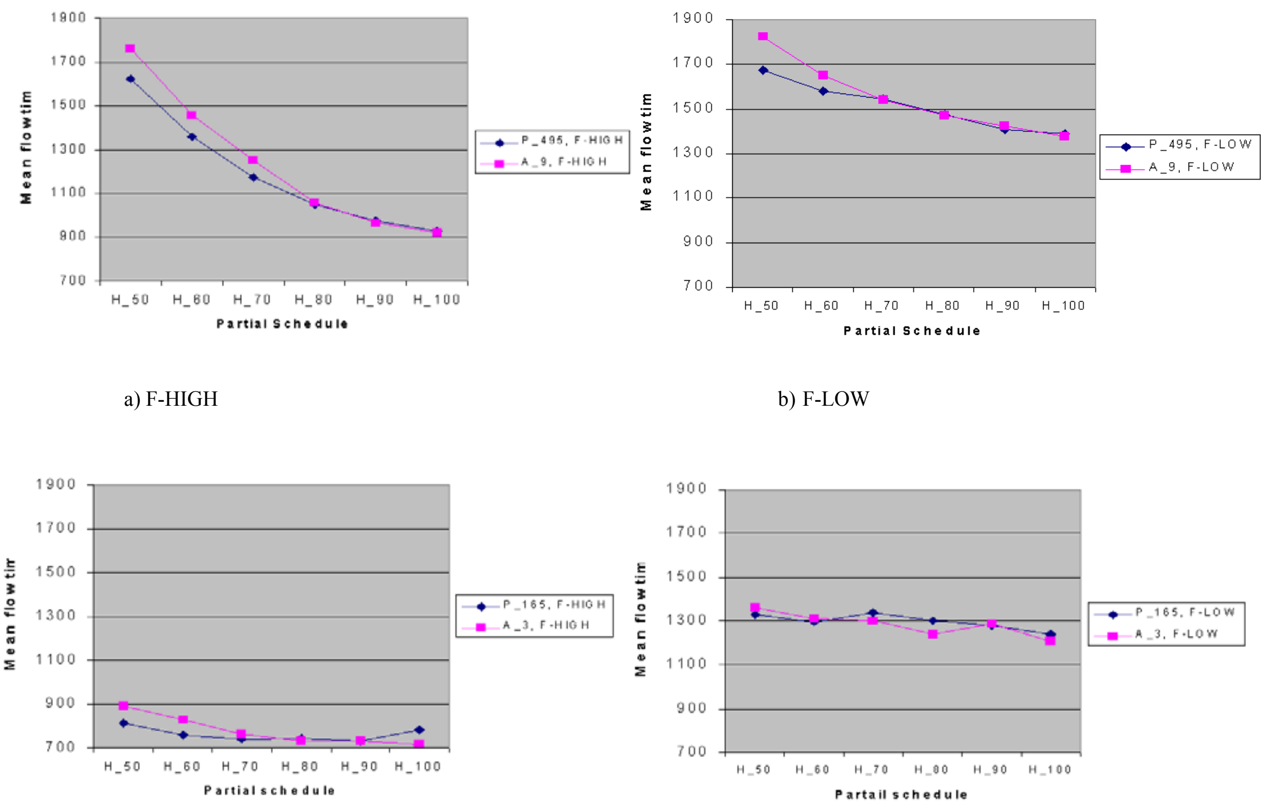
Figure 4.8. Comparison of when to schedule policies for partial and full schedules.

the PERIODIC policy is in fact better than the ARRIVAL policy as the partial scheduling level is low whereas, the ARRIVAL policy becomes better when the partial scheduling level increases and gets closer to full scheduling. We also observe that ARRIVAL policy is more sensitive to partial scheduling as compared to the PERIODIC policy. Notice that crossover point moves (shifts) to the left when the scheduling frequency is high. Because the difference between PERIODIC and ARRIVAL decrease when scheduling frequency increases. Moreover, rescheduling interval is variable in the ARRIVAL policy (a long rescheduling interval can be followed by a shorter interval or a longer interval) as compared to fixed scheduling interval in PERIODIC policy.

When the scheduling interval is too long the system can process all the jobs scheduled by the low partial schedule, and waits idle. As compared to fixed scheduling interval of the PERIODIC policy, when the scheduling interval is too long in the ARRIVAL policy, we insert some unnecessary idleness in the system since the machines can process all the jobs scheduled according to partial scheduling for the ARRIVAL policy. For that reason the ARRIVAL policy display inferior performance when the partial scheduling level is low.

In short, we can conclude that the RATIO policy, which relies on the output process, is better than the ARRIVAL policy (which relies on the input process) and the PERIODIC policy (which does rely neither on input nor output process). Our results also indicate that the ARRIVAL policy performs better than the PERIODIC policy with full scheduling.

We also test whether the difference of the performances between the scheduling policies is significant or not, for the A_3 case (Figure 4.8.f and h). We first compare the on-line scheduling scheme with the off-line scheduling scheme (specifically with the RATIO policy with the dispatch rule). The results of the paired t-test reveal that it significant for the high flexibility case in favour of the off-line scheduling algorithm (Table 4.12). In low flexibility case we could not identify a significant difference between the policies due to the result of one replication, which can extremely different then other replications. This in turn creates a high variance in the confidence interval estimation. We also tested the RATIO policy with the PERIODIC policy. The same observations are made for the comparison of PERIOD and RATIO policies (Table 4.13).



a) F-HIGH b) F-LOW

c) F-HIGH d) F-LOW

Figure 4.9. Comparison of PERIODIC and ARRIVAL schedules for different Partial Schedules.

4.3) PROCESSING TIME VARIATION

In a typical real manufacturing environment actual processing times of operations may be different than the estimated processing times used in the scheduling process due to changing machining conditions and other factors. This uncertainty of course can degrade the performance of scheduling decisions as well as the performance of the entire system. In this section, we will investigate the impact of processing time variations (PV) on the scheduling decisions and the system performance.

The estimated processing times used in the scheduling algorithm are still drawn from a 2-Erlang distribution. Actual processing times differ from the estimates by a certain amount when schedule is implemented via the simulation model. Specifically, actual times are generated from a truncated normal distribution with mean equal to the estimated processing time. During simulation experiments, the coefficient of variation (CV) is 0.4.

We run the simulation model for the three *when to schedule* (ARRIVAL, PERIODIC and RATIO) policies and two *how to schedule* policies (partial scheduling and full scheduling). Figure 4.10 (and Table 4.8 in the Appendix) presents the results for the scheduling frequencies corresponding to A_3. The performances of policies for without process time variation are quite mixed, so we display the results in Table 4.11. The following observations are made from the results:

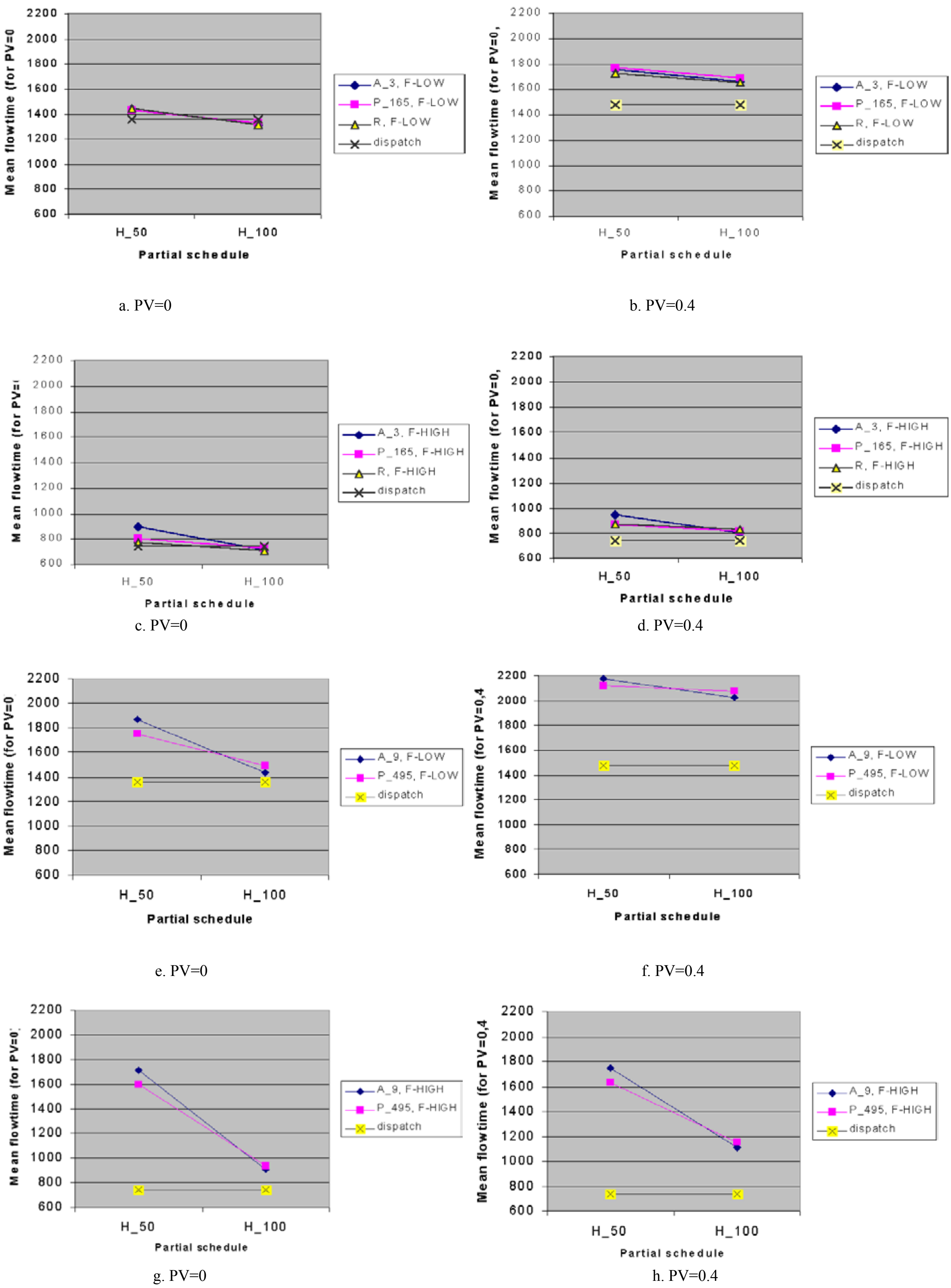
First, in the without process time variation case the performances of off-line policies with full scheduling are better than simple dispatch rule for the scheduling frequency A_3 (Figure 4.10.a and c). However, dispatching rule performs better than the PERIODIC, ARRIVAL, and RATIO policies for partial scheduling. Note also that, the simple dispatching rule performs better than the off-line algorithm for A_9 case. This means that the rules which are commonly used in practice are quite effective in dynamic and stochastic environments.

Second, as can be seen in Figure 4.10, the performance of scheduling methods and the dispatch rule detoriates as PV increases. However, off-line algorithm is more sensitive to process time variation than the simple dispatch rule. As seen in Figure 4.10.b, d, f, and h, the performance of the beam search algorithm detoriates more than on-line algorithm as PV increases. For both partial and full scheduling the LWRK rule performs better than the three policies at the scheduling frequencies A_3 and A_9. Note also that, difference becomes larger when we decrease the scheduling frequency from A_3 to A_9.

Third, we observe that the performance of the system is better with the full scheduling scheme compared to the partial scheduling scheme.

Fourth, the PERIODIC policy performs better than the ARRIVAL policy in process time variation case with partial scheduling. However, ARRIVAL policy performs better than the PERIODIC policy in full schedule case. The RATIO policy seems to be the best among the three policies. The same behavior was also observed in the without processing time case.

Fifth, the difference between partial and full scheduling decreases as PV increases (Figure 4.11.e, f, g and, h). This is because of the fact that, full scheduling is more affected by PV compared to partial scheduling (Figure 4.11.a, b, c and, d). Note that this observation is more apparent for low flexibility.



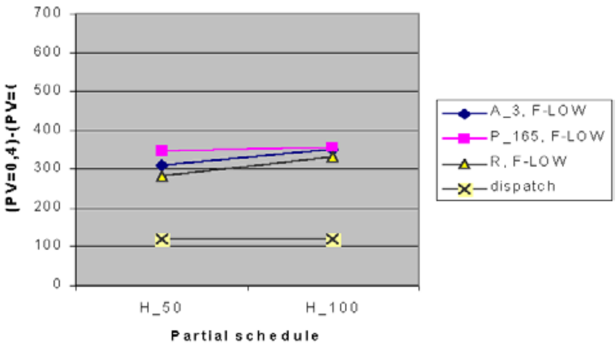
a. PV=0 b. PV=0.4

c. PV=0 d. PV=0.4

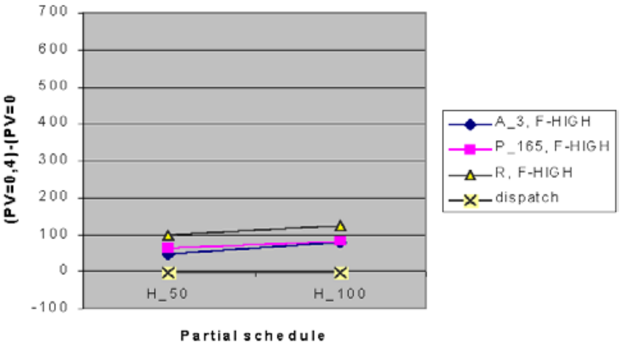
e. PV=0 f. PV=0.4

g. PV=0 h. PV=0.4

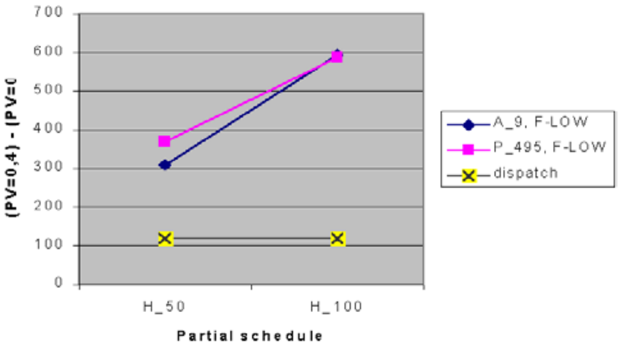
Figure 4.10. Mean flowtimes for the ARRIVAL, PERIODIC, RATIO policies for PV=0 and PV=0.4.



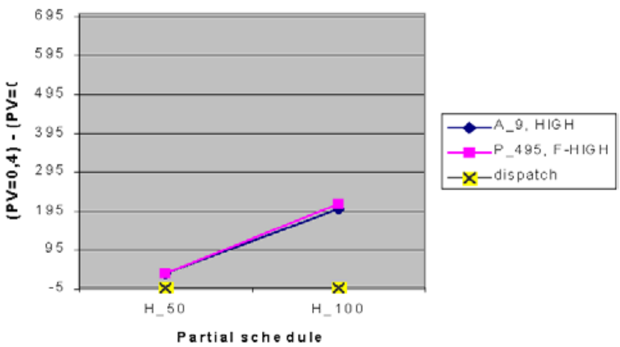
a) (PV=0.4) - (PV=0)



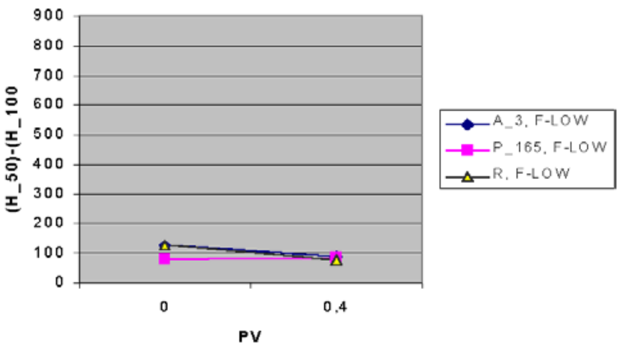
b) (PV=0.4) - (PV=0)



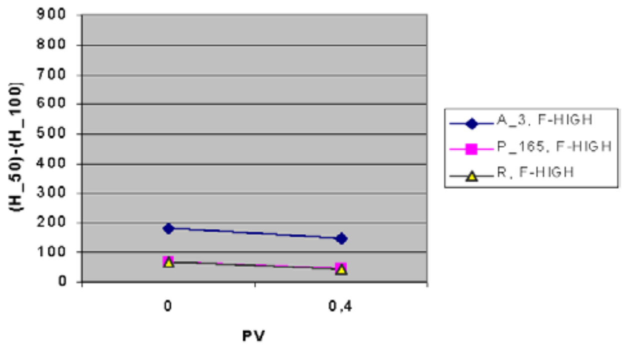
c) (PV=0.4) - (PV=0)



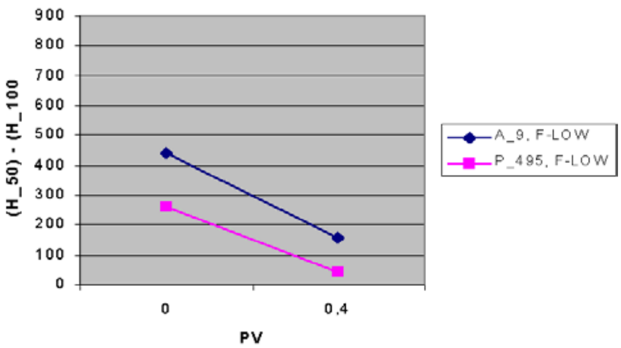
d) (PV=0.4) - (PV=0)



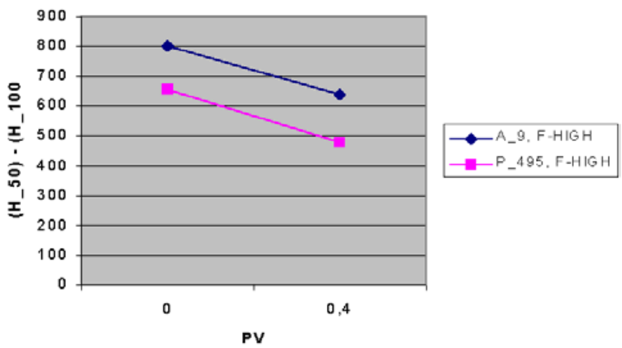
e) (H_50) - (H_100)



f) (H_50) - (H_100)



g) (H_50) - (H_100)



h) (H_50) - (H_100)

Figure 4.11. Change of performance with PV and change of performance with partial schedule.

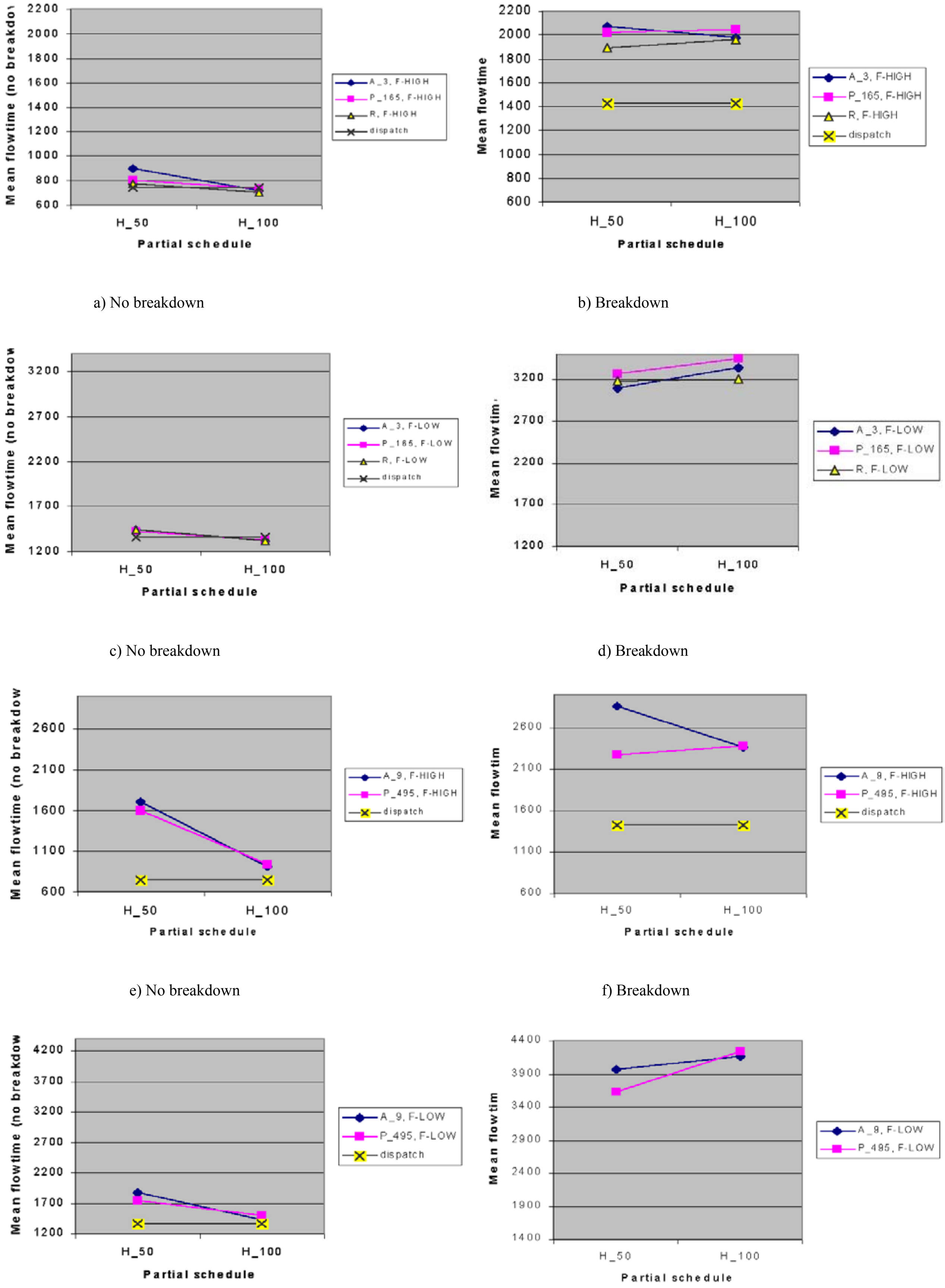
4.4. MACHINE BREAKDOWNS

In this section, we examine the impact of machine breakdowns on the scheduling policies. Machine breakdowns are modelled by the busy time approach (Law and Kelton, 1991). With this approach a random uptime is generated from a busy time distribution. The machine is considered as up, until its total accumulated busy (processing) time reaches the end of this uptime. Then it fails for a random down time, after which an uptime will again be generated. In our experiments the mean for busy time is 180, while the mean for the down time is 20. The busy and down times are drawn from a gamma distribution with shape parameter of 0.7. Thus, the systems overall availability level is 90%, which gives the long run ratio of a machine busy time to busy plus down time.

The results are displayed in Figures 4.12 and Figure 4.13. As expected, machine breakdown negatively affects the performances of scheduling policies (Figure 4.12). Mean flowtime performance of the system deteoriates regardless of the level of scheduling frequency, full vs. partial scheduling, and flexibility levels. We also observe that the negative impact of machine breakdown is larger with full scheduling than the partial scheduling. This may be due to the fact that more number of operations in the schedule are affected by these breakdown events (Figure 4.13. a, b, c, and d).

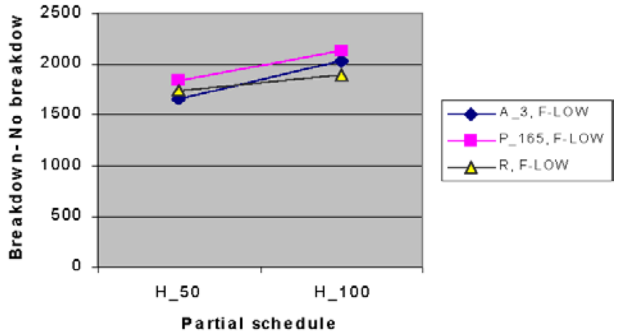
Second, CPU time during the experiments increases approximately 10 times. This is due to the fact that, when a machine breakdown occurs the algorithm spends more time to find new machines for the affected jobs waiting in the queue in front of that machine.

Third, as compared to PV (with parameter 0.4), machine breakdown (with 90% efficiency) has more negative effect on the system performance (Figure 4.10 and 4.12). This finding is consistent with the result found in static environment (Sabuncuoglu and Bayiz, 2000).

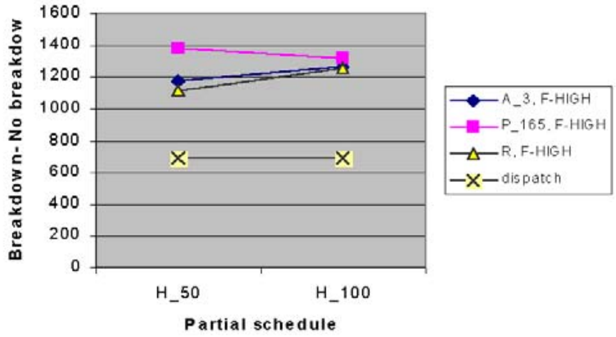


a) No breakdown
b) Breakdown
c) No breakdown
d) Breakdown
e) No breakdown
f) Breakdown
g) No breakdown
h) Breakdown

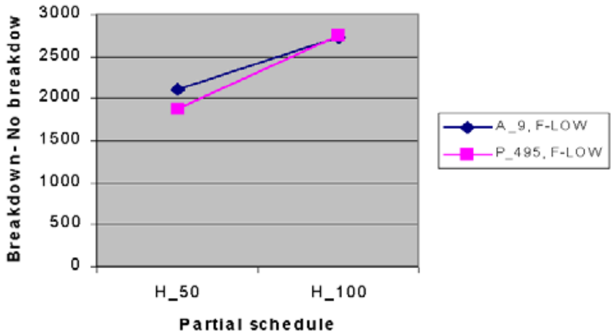
Figure 4.12. Mean flowtimes of scheduling policies for no breakdown and breakdown cases.



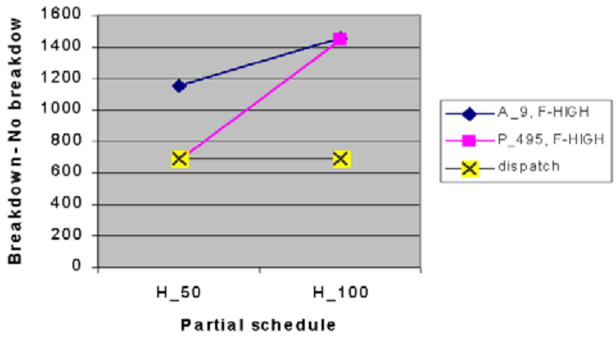
a) Breakdown - No breakdown



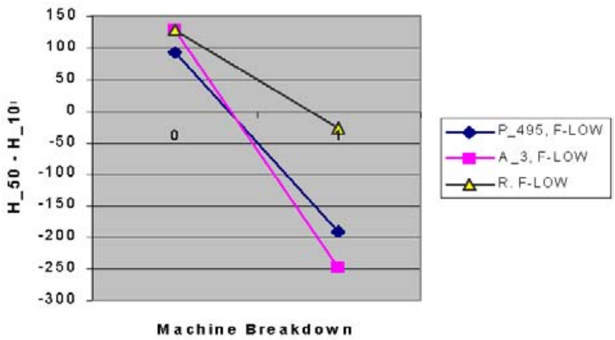
b) Breakdown - No breakdown



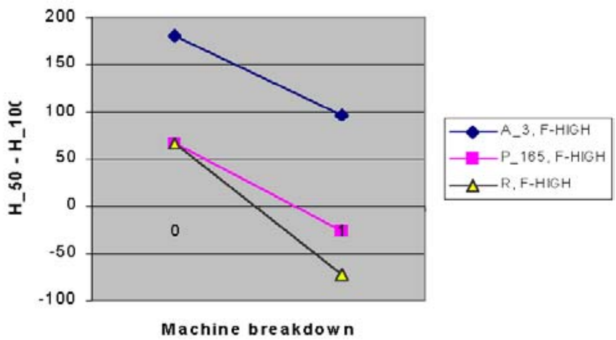
c) Breakdown - No breakdown



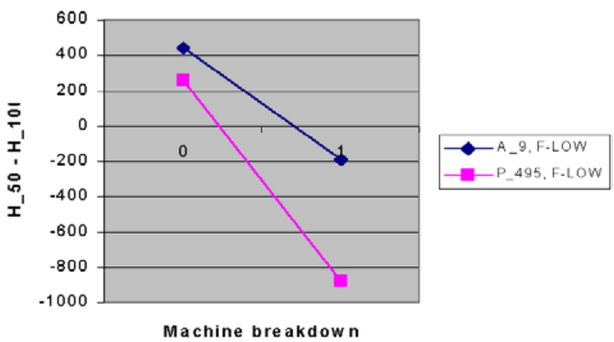
d) Breakdown - No breakdown



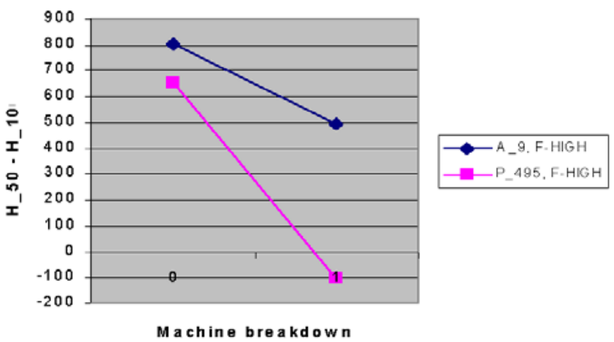
e) (H_50) - (H_100)



f) (H_50) - (H_100)



g) (H_50) - (H_100)



h) (H_50) - (H_100)

Figure 4.13. Change of performances with machine breakdown and with partial schedule.

Fourth, difference between scheduling policies increases with machine breakdowns (compared to no machine breakdown case) when partial schedule is implemented.

Fifth, the performances of the scheduling policies are significantly better with full scheduling than partial scheduling in the no breakdown case. However, as we have breakdowns, we observe an opposite behaviour of the scheduling policies. Specifically, performance of the system gets better with partial scheduling (Figure 4.12). Only exception is observed with the scheduling policy ARRIVAL when the flexibility is high (i.e., F-HIGH). Note that this observation is made for both the A_3, and A_9 frequency levels. This counter intuitive result (deterioration in the performance of the system with machine breakdowns) can be attributed to the fact that the benefit of using full scheduling totally diminishes when there are machine breakdowns (entire scheduling may be totally useless with machine breakdowns).

Sixth, the PERIODIC policy performs better than the ARRIVAL policy in breakdown case with partial scheduling. However, ARRIVAL policy performs better than the PERIODIC policy in full schedule case. The only exception is A_3 and F-LOW. The RATIO policy seems to be the best among the three policies. The same behavior was also observed in no breakdown case.

Seventh, the off-line algorithm is more sensitive to machine breakdowns than the on-line dispatching rule. Specifically, the performance of the off-line schedule detoriates more than the on-line schedules with breakdowns. For the full scheduling case (in high flexibility) the algorithm with scheduling frequency A_3 performs better than the dispatch when there is no machine breakdown (Figure 4.12.a). However, in the breakdown case the dispatch rule performs better than all three scheduling methods (Figure 4.12.b). For the scheduling frequency A_9, although the dispatching performs better than the off-line scheduling for no machine breakdown case, the difference becomes larger in machine breakdown case (Figure 4.12.e and f). Notice that the results for the dispatching rule is not given for the low flexibility cases. We could not be able to run the simulation model for the low flexibility case. In this case, the system is saturated due to exponentially growing job populations. This means that, when there is no flexibility in the system, the dispatch policy can not cope with the adverse effect of machine breakdowns.

Eventually, the differences between full and partial scheduling are nearly same for all three policies when there is no machine breakdown. However, this difference is getting significantly larger when there is breakdown (Figure 4.13.e, f, g and h).

CHAPTER 5

CONCLUSION

In this paper three issues are addressed. In the first part, we briefly reviewed the reactive scheduling literature and classified the existing studies. In the second part, we modified the scheduling system proposed earlier (Sabuncuoglu and Karabuk, 1998) to have it working in a dynamic and stochastic environment. In the final part we compared different rescheduling policies.

In the existing studies, the long run performances of the reactive policies are not measured. Most of the studies analyzed the performance of the scheduling methods in static and deterministic environments. In this paper, we studied long run performances of scheduling methods in dynamic environment. The following conclusions are drawn from our study:

First, as the frequency of rescheduling increases the performance of the system becomes better. Thus, system performance is directly proportional with reschedule frequency.

Second, although scheduling frequency has significant affect on system performance, type of response is also important. Our results indicate that variable time reschedule policy is better than fixed time rescheduling policy.

Third, we tested partial rescheduling with full rescheduling. Our experiments indicate that performance of the FMS system is affected with the level of partial rescheduling. Full rescheduling performs better than partial rescheduling. On the other hand, CPU times decreases. This conclusion is also consistent with the result drawn by Sabuncuoglu and Bayiz (2000).

Fourth, on-line scheduling (dispatch) rules are more robust to process time variation and machine breakdowns then the off-line scheduling algorithm. The dispatching rule not only performs better in stochastic environment but also spends less CPU time compared to the off-line algorithm. As a result, it is much more beneficial to use on-line scheduling schemes in dynamic and stochastic environments.

For the future research, different efficiency level for the machine breakdowns can be tested in the simulation experiments. Moreover, the effects of different duration of mean machine up and down times for the same efficiency level can also be analyzed. Also, different rescheduling methods can be investigated (i.e., adaptive rescheduling policy) in stochastic and dynamic environment. The beam search algorithm can be tested using different local and global evaluation functions. Finally, the simulation experiments can be extended to cover other combinations of experimental factors (i.e., different machine load levels, AGV speeds, tardiness factor).

BIBLIOGRAPHY

[1] Akturk, S. and Gorgulu, E., "Match-Up Scheduling Under a Machine Breakdown", *European Journal of Operational Research*, (Vol. 12, 1997), pp 81-97.

[2] Carlier Jacques, "The One-Machine Sequencing Problem", *European Journal of Operational Research*, (Vol.11,1982), pp 42-47.

[3] Church, L. and Uzsoy, R., "Analysis of Periodic and Event Driven Rescheduling Policies in Dynamic Shops", *International Journal of Computer Integrated Manufacturing*, (Vol. 5, 1992), pp153-163.

[4] Daniels, R. and Kouvelis, P., "Robust Scheduling to Hedge Against Processing Time Uncertainty in Single Stage Production", *Management Science*, (Vol. 41, 1995), pp.363-376.

[5] Kutanoglu, E. and Sabuncuoglu, I., "Job Shop Scheduling Under Dynamic and Stochastic Manufacturing Environment", *Master Thesis*, 1995, Bilkent University, Department of Industrial Engineering.

[6] Law, A.M. and Kelton, W.D., *Simulation Modelling and Analysis*, 1991, McGraw-Hill.

[7] Mehta, S. and Uzsoy, R., "Predictable Scheduling of a Single Machine Subject to Breakdowns", *International Journal Computer Integrated Manufacturing*, (Vol. 12, 1999), pp15-38.

[8] Mehta, S. and Uzsoy, R., "Predictable Scheduling of a Job Shop Subject to Breakdowns", *IEEE Transactions on Robotics and Automation*, Vol. 14, (1998), pp 365-378.

[9] O'Donovan, R. ,Uzsoy, R. and McKay, K., "Predictable Rescheduling of a Single Machine with Machine Breakdowns and Sensitive Jobs", *Research Memorandum*, (97-8, 1997), Purdue University.

[10] Ovacik, I.M. and Uzsoy, R., "Analysis of Periodic and Event-Driven Rescheduling Policies in Dynamic Shops", *International Journal of Computer Integrated Manufacturing*, (Vol 5, 1992), pp. 153-163.

[11] Ow, P.S. and Morton T., "Filtered Beam Search In Scheduling", *International Journal of Production Research,* (Vol 26, 1988), pp.35-62.

[12] Sabuncuoglu, I. and Bayiz, M., "Analysis of Reactive Scheduling Problems in a Job Shop Environment", *European Journal of Operational Research*, (126, 2000), pp.567-586.

[13] Sabuncuoglu, I. and Bayiz, M., "Job Shop Scheduling with Beam Search", *European Journal of Operational Research*, (Vol. 118, 1999), pp.390-412.

[14] Sabuncuoglu, I., and Hommertzheim, L., "Dynamic Dispatching Algorithm for Machines and Automated Guided Vehicles in a Flexible Manufacturing System", *International Journal of Production Research,* (Vol. 30, 1992), pp.1059-1079.

[15] Sabuncuoglu, I. and Karabuk, S., “A Beam Search Algorithm and Evaluation of Scheduling Approach for Flexible Manufacturing Systems”, *IIE Transactions*, (Vol. 30, 1998), pp.179-191.

[16] Sabuncuoglu, I. and Karabuk, S., “Rescheduling Frequency in an FMS with Uncertain Processing Times and Unreliable Machines”, *Journal of Manufacturing Systems*, Volume 18, No. 4, (1999).

[17] Sabuncuoglu, I. and Toptal, A. , “Distributed Scheduling: A Review of Concepts and Recent Applications”, *Master Thesis*, 1999, Bilkent University, Department of Industrial Engineering.

[18] Svestka, A.J. and Abumaziar, R.J., “Rescheduling Job Shops Under Random Disruptions”, *International Journal of Production Research,* (Vol. 35, 1997), pp. 2065-2082.

[19] Wu, L. and Storer, R., “Robustness Scheduling for Job Shop”, *IIE Transactions*, (Vol. 26, 1994), pp32-41.

[20] Wu, R. and Storer, R., “One Machine Rescheduling Heuristics with Efficiency and Stability as Criteria”, *Computers Operations Research*, (Vol. 20, 1992), pp.1-14.

APPENDIX

Table 4.1. Outputs for limited and unlimited buffer capacities.

For Q=10

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	A_1	Low	200	11543	848	160	328.7	79	85
H_100	A_1	Low	400	23648	1043	311	1163.42	79	82
H_100	A_1	Low	600	35122	977	260	1414.43	78	82
H_100	A_1	Low	800	45482	1056	325	3015.05	80	85
H_100	A_1	Low	1000	55865	1175	426	4993.65	82	86
H_100	A_1	Low	1200	66621	1208	455	6301.68	82	85
H_100	A_1	Low	1400	77812	1180	428	7309.37	82	86
H_100	A_1	Low	1600	89156	1152	402	6876.58	82	86
H_100	A_1	Low	1800	101409	1157	404	7521.87	81	85

For Q=100

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	A_1	Low	100	6494	753	96	74.97	75	81
H_100	A_1	Low	200	11904	946	241	414.4	79	84
H_100	A_1	Low	400	23840	1201	457	15646.65	79	83
H_100	A_1	Low	600	35235	1152	422	2146.7	78	83
H_100	A_1	Low	800	45555	1220	481	3890.07	80	85
H_100	A_1	Low	1000	56202	1350	595	7544.27	82	86
H_100	A_1	Low	1200	66746	1382	620	8139.87	82	86
H_100	A_1	Low	1400	78110	1384	618	9834.17	82	86
H_100	A_1	Low	1600	89455	1368	601	10122	82	86
H_100	A_1	Low	1800	101466	1387	616	11498.8	81	85

For Q=10

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	A_1	High	100	6101	447	1	25.47	79	67
H_100	A_1	High	200	11330	568	51	232.93	85	73
H_100	A_1	High	400	22802	619	67	542.4	82	73
H_100	A_1	High	600	34981	627	73	912.88	82	72
H_100	A_1	High	800	45537	780	174	4044.53	85	77
H_100	A_1	High	1000	55367	899	258	6840.98	87	80
H_100	A_1	High	1200	66296	905	259	17318.12	87	79
H_100	A_1	High	1400	77400	879	238	8123.2	88	79
H_100	A_1	High	1600	88743	851	215	8241.27	87	79
H_100	A_1	High	1800	100933	825	199	8364.12	86	79

Q = 100

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	A_1	High	100	6101	447	1	27.97	79	67
H_100	A_1	High	200	11330	568	51	227.38	85	73
H_100	A_1	High	400	22788	620	67	591.87	82	73
H_100	A_1	High	600	34974	629	72	984.67	82	71
H_100	A_1	High	800	45353	771	164	4030.13	85	76
H_100	A_1	High	1000	55266	848	210	5749.55	87	79
H_100	A_1	High	1200	66276	848	208	6484.93	87	79
H_100	A_1	High	1400	77265	832	194	6776.08	87	79
H_100	A_1	High	1600	88692	809	175	7001.75	87	79
H_100	A_1	High	1800	100924	786	163	7280.52	86	77

Table 4.1. Outputs for limited and unlimited buffer capacities (Cont'd).

Q=10

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	P_200	Low	200	11824	966	226	139.38	78	84
H_100	P_200	Low	400	23763	1156	385	440.37	79	81
H_100	P_200	Low	600	35076	1066	318	559.38	78	81
H_100	P_200	Low	800	45417	1146	389	955.13	80	83
H_100	P_200	Low	1000	55781	1230	459	1447.85	82	84
H_100	P_200	Low	1200	66704	1246	470	1792.3	82	84
H_100	P_200	Low	1400	77975	1231	451	2017.72	82	84
H_100	P_200	Low	1600	89548	1251	435	2163.67	82	84

Q=100

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	P_200	Low	200	11956	984	244	155.2	78	84
H_100	P_200	Low	400	23752	1187	418	482.48	79	82
H_100	P_200	Low	600	35192	1091	341	577.77	77	82
H_100	P_200	Low	800	45420	1173	414	1081.58	80	84
H_100	P_200	Low	1000	55700	1282	510	1761.07	82	85
H_100	P_200	Low	1200	66781	1300	523	2105.7	82	82
H_100	P_200	Low	1400	77824	1286	505	2388.7	82	85
H_100	P_200	Low	1600	89350	1260	479	2561.28	82	84

Q=10

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	P_200	High	200	11353	613	44	147.58	85	65
H_100	P_200	High	400	22982	637	45	271.13	82	64
H_100	P_200	High	600	34981	650	49	430.88	82	63
H_100	P_200	High	800	45142	769	125	1547.42	86	67
H_100	P_200	High	1000	54893	804	139	2008.55	88	70
H_100	P_200	High	1200	66362	786	125	2164.88	87	68
H_100	P_200	High	1400	77289	778	117	2369.65	88	68
H_100	P_200	High	1600	88679	762	105	2486.33	88	68

Q=100

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	P_200	High	200	11353	613	43	148.25	85	65
H_100	P_200	High	400	22980	636	45	321.42	82	63
H_100	P_200	High	600	35038	644	47	511.27	82	63
H_100	P_200	High	800	45132	762	122	1675.77	86	67
H_100	P_200	High	1000	54791	796	136	2127.55	88	69
H_100	P_200	High	1200	66415	772	119	2218.78	87	67
H_100	P_200	High	1400	77272	761	109	2399	87	68
H_100	P_200	High	1600	88698	745	98	2508.83	87	67

Table 4.1. Outputs for limited and unlimited buffer capacities (Cont'd).

Q=10

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	R_100	Low	200	13406	1814	1031	570.77	68	70
H_100	R_100	Low	400	25863	2476	1671	1904.98	71	71
H_100	R_100	Low	600	37843	2776	1964	3713.88	72	72
H_100	R_100	Low	800	48451	3014	2212	5990.17	75	74
H_100	R_100	Low	1000	59074	3377	2566	9328.05	77	75
H_100	R_100	Low	1200	70107	3630	2819	13538.93	77	76
H_100	R_100	Low	1400	81110	3735	2922	16132.22	78	77
H_100	R_100	Low	1600	92200	3813	2997	18856.43	79	77

Q=100

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	R_100	Low	200	13365	1830	1049	541.1	68	70
H_100	R_100	Low	400	25961	2487	1675	1801.62	71	70
H_100	R_100	Low	600	37610	2673	1867	3359.93	72	71
H_100	R_100	Low	800	48618	3006	2197	7597.8	74	73
H_100	R_100	Low	1000	59343	3349	2539	11730.62	76	75
H_100	R_100	Low	1200	70464	3639	2829	14548.67	77	75
H_100	R_100	Low	1400	81496	3801	2987	17887.85	78	76
H_100	R_100	Low	1600	93403	3944	3123	22614.93	78	76

Q=10

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	R_100	High	200	12079	1050	327	714.17	78	55
H_100	R_100	High	400	23819	1219	462	1694.1	79	57
H_100	R_100	High	600	36142	1278	502	2586.82	79	56
H_100	R_100	High	800	46882	1565	778	7376.13	82	58
H_100	R_100	High	1000	57513	1848	1057	13136.7	84	59
H_100	R_100	High	1200	67883	2047	1245	16894	85	60
H_100	R_100	High	1400	78688	2065	1260	19838.73	85	60
H_100	R_100	High	1600	89974	2067	1260	22475.12	86	61

Q=100

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	R_100	High	200	12079	1050	327	648.4	78	55
H_100	R_100	High	400	23873	1223	464	1532.3	78	56
H_100	R_100	High	600	36138	1287	512	2546.92	79	56
H_100	R_100	High	800	46836	1565	779	6320.38	82	59
H_100	R_100	High	1000	57244	1861	1065	11101.83	83	60
H_100	R_100	High	1200	67988	2057	1254	15739.18	85	61
H_100	R_100	High	1400	78695	2047	1245	18695.37	85	61
H_100	R_100	High	1600	90057	2049	1239	20555.52	86	61

Table 4.2. Comparison of H_100 and H_50 for F-LOW and F-HIGH

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_50	P_200	Low	200	11911	1024	276	73.33	77	82
H_50	P_200	Low	400	23783	1257	474	213.62	79	81
H_50	P_200	Low	600	35241	1187	418	265.97	78	82
H_50	P_200	Low	800	45504	1243	471	454.58	80	84
H_50	P_200	Low	1000	56259	1362	578	816.85	82	85
H_50	P_200	Low	1200	66830	1402	611	966.43	82	84
H_50	P_200	Low	1400	78000	1387	592	1026.37	82	85
H_50	P_200	Low	1600	89498	1369	574	1137.95	82	85
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**MC. Utilization**	**AGV Utilization**
H_100	P_200	Low	200	11758	949	203	134.4	78	84
H_100	P_200	Low	400	23752	1139	367	404.82	79	82
H_100	P_200	Low	600	35133	1061	305	479.07	78	82
H_100	P_200	Low	800	45576	1150	385	1086.58	80	84
H_100	P_200	Low	1000	55873	1260	484	1702.75	82	85
H_100	P_200	Low	1200	66702	1281	500	1928.62	82	84
H_100	P_200	Low	1400	77986	1267	483	2013.6	82	84
H_100	P_200	Low	1600	89422	1247	462	2357.92	82	84
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**MC. Utilization**	**AGV Utilization**
H_50	P_200	High	200	11420	766	75	136.62	83	57
H_50	P_200	High	400	23228	797	92	287.02	81	55
H_50	P_200	High	600	35100	806	94	412.65	82	55
H_50	P_200	High	800	45147	898	165	1136.35	85	60
H_50	P_200	High	1000	54992	919	179	1353.9	87	63
H_50	P_200	High	1200	66640	899	160	1452.08	87	61
H_50	P_200	High	1400	77400	892	152	1635.13	87	62
H_50	P_200	High	1600	88762	876	139	1740.67	87	62
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**MC. Utilization**	**AGV Utilization**
H_100	P_200	High	200	11353	613	44	147.58	85	65
H_100	P_200	High	400	22982	637	45	271.13	82	64
H_100	P_200	High	600	34981	650	49	430.88	82	63
H_100	P_200	High	800	45142	769	125	1547.42	86	67
H_100	P_200	High	1000	54893	804	139	2008.55	88	70
H_100	P_200	High	1200	66362	786	125	2164.88	87	68
H_100	P_200	High	1400	77289	778	117	2369.65	88	68
H_100	P_200	High	1600	88679	762	105	2486.33	88	68
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**MC. Utilization**	**AGV Utilization**
H_50	A_1	Low	200	11784	936	219	193.63	79	84
H_50	A_1	Low	400	23809	1160	401	701.32	79	83
H_50	A_1	Low	600	35101	1084	346	844.38	78	82
H_50	A_1	Low	800	45475	1156	406	1608.27	80	84
H_50	A_1	Low	1000	55863	1277	513	2901.48	82	85
H_50	A_1	Low	1200	66741	1313	543	3495.1	82	86
H_50	A_1	Low	1400	78206	1301	530	4009.63	82	86
H_50	A_1	Low	1600	89635	1299	525	4328.33	82	85

Table 4.2. Comparison of H_100 and H_50 for F-LOW and F-HIGH (Cont'd)

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_100	A_1	Low	200	11543	848	160	328.7	79	85
H_100	A_1	Low	400	23648	1043	311	1163.42	79	82
H_100	A_1	Low	600	35122	977	260	1414.43	78	82
H_100	A_1	Low	800	45482	1056	325	3015.05	80	85
H_100	A_1	Low	1000	55865	1175	426	4993.65	82	86
H_100	A_1	Low	1200	66621	1208	455	6301.68	82	85
H_100	A_1	Low	1400	77812	1180	428	7309.37	82	86
H_100	A_1	Low	1600	89156	1152	402	6876.58	82	86
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**MC. Utilization**	**AGV Utilization**
H_50	A_1	High	200	11330	583	36	174.04	84	67
H_50	A_1	High	400	22878	621	46	362.75	82	67
H_50	A_1	High	600	34923	631	49	558.43	82	67
H_50	A_1	High	800	45379	758	134	2589.38	85	72
H_50	A_1	High	1000	55208	840	182	3846.82	87	76
H_50	A_1	High	1200	66322	836	177	4238.5	87	75
H_50	A_1	High	1400	77179	820	162	4478.58	87	76
H_50	A_1	High	1600	88763	797	146	4626.22	87	75
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**MC. Utilization**	**AGV Utilization**
H_100	A_1	High	200	11330	568	51	232.93	85	73
H_100	A_1	High	400	22802	619	67	542.4	82	73
H_100	A_1	High	600	34981	627	73	912.88	82	72
H_100	A_1	High	800	45537	780	174	4044.53	85	77
H_100	A_1	High	1000	55367	899	258	6840.98	87	80
H_100	A_1	High	1200	66296	905	259	7318.12	87	79
H_100	A_1	High	1400	77400	879	238	8123.2	88	79
H_100	A_1	High	1600	88743	851	215	8241.27	87	79
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**MC. Utilization**	**AGV Utilization**
H_50	R_100	Low	200	12882	1564	772	431.37	73	67
H_50	R_100	Low	400	24718	1892	1080	1245.57	76	68
H_50	R_100	Low	600	36167	1854	1043	1682.87	76	68
H_50	R_100	Low	800	46735	1939	1134	3033.88	78	70
H_50	R_100	Low	1000	57644	2214	1404	5412.45	79	70
H_50	R_100	Low	1200	68877	2427	1610	7139.48	80	70
H_50	R_100	Low	1400	79374	2476	1661	8600.37	80	71
H_50	R_100	Low	1600	90612	2519	1699	9762.95	81	71
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**MC. Utilization**	**AGV Utilization**
H_100	R_100	Low	200	13406	1814	1031	570.77	68	70
H_100	R_100	Low	400	25863	2476	1671	1904.98	71	71
H_100	R_100	Low	600	37843	2776	1964	3713.88	72	72
H_100	R_100	Low	800	48451	3014	2212	5990.17	75	74
H_100	R_100	Low	1000	59074	3377	2566	9328.05	77	75
H_100	R_100	Low	1200	70107	3630	2819	13538.93	77	76
H_100	R_100	Low	1400	81110	3735	2922	16132.22	78	77
H_100	R_100	Low	1600	92200	3813	2997	18856.43	79	77

Table 4.2. Comparison of H_100 and H_50 for F-LOW and F-HIGH (Cont'd)

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	MC. Utilization	AGV Utilization
H_50	R_100	High	200	11340	630	43	88.55	84	51
H_50	R_100	High	400	22854	693	69	212.23	82	50
H_50	R_100	High	600	35081	701	69	321.53	82	50
H_50	R_100	High	800	45313	828	157	858.48	85	51
H_50	R_100	High	1000	55158	908	210	1252.78	87	52
H_50	R_100	High	1200	66341	905	204	1422.85	87	51
H_50	R_100	High	1400	77414	895	191	1600.68	87	51
H_50	R_100	High	1600	88677	874	173	1724.23	87	52
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**MC. Utilization**	**AGV Utilization**
H_100	R_100	High	200	12079	1050	327	714.17	78	55
H_100	R_100	High	400	23819	1219	462	1694.1	79	57
H_100	R_100	High	600	36142	1278	502	2586.82	79	56
H_100	R_100	High	800	46882	1565	778	7376.13	82	58
H_100	R_100	High	1000	57513	1848	1057	13136.7	84	59
H_100	R_100	High	1200	67883	2047	1245	16894	85	60
H_100	R_100	High	1400	78688	2065	1260	19838.73	85	60
H_100	R_100	High	1600	89974	2067	1260	22475.12	86	61

Table 4.3. Comparison of different scheduling policies for F-LOW

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	Resch. No.	MC. Utilization
H_100	R_100	Low	200	12302	1222	445	81.65	15	74
H_100	R_100	Low	400	24806	1636	819	555.55	23	75
H_100	R_100	Low	600	35802	1666	859	959.59	32	76
H_100	R_100	Low	800	46484	1752	950	1453.6	37	78
H_100	R_100	Low	1000	57401	1961	1151	2166.8	42	79
H_100	R_100	Low	1200	68496	2166	1352	2394.2	46	80
H_100	R_100	Low	1400	79319	2217	1403	3032.05	50	80
H_100	R_100	Low	1600	90576	2274	1458	3754.48	55	81
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Resch. No.**	**MC. Utilization**
H_100	R_25	Low	200	11798	976	231	122.95	74	78
H_100	R_25	Low	400	23792	1128	359	350.93	144	79
H_100	R_25	Low	600	35182	1054	301	444.28	227	78
H_100	R_25	Low	800	45445	1133	372	741.7	273	80
H_100	R_25	Low	1000	55821	1234	461	1139.98	308	82
H_100	R_25	Low	1200	66635	1256	480	1416.22	371	82
H_100	R_25	Low	1400	78142	1248	471	1726.92	427	82
H_100	R_25	Low	1600	89412	1236	457	1764.57	488	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Resch. No.**	**MC. Utilization**
H_100	P_50	Low	200	11671	891	184	344.75	233	79
H_100	P_50	Low	400	23675	1105	359	1437.63	475	79
H_100	P_50	Low	600	35185	1026	305	1715.23	706	78
H_100	P_50	Low	800	45379	1105	370	3264.92	910	80
H_100	P_50	Low	1000	55942	1214	462	5493.25	1121	82
H_100	P_50	Low	1200	66736	1237	480	7837.1	1337	82
H_100	P_50	Low	1400	78007	1216	456	7836.37	1563	82
H_100	P_50	Low	1600	89421	1200	440	8074.78	1791	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Resch. No.**	**MC. Utilization**
H_100	P_200	Low	200	11758	949	203	134.4	59	78
H_100	P_200	Low	400	23752	1139	367	404.82	120	79
H_100	P_200	Low	600	35133	1061	305	479.07	178	78
H_100	P_200	Low	800	45576	1150	385	1086.58	230	80
H_100	P_200	Low	1000	55873	1260	484	1702.75	282	82
H_100	P_200	Low	1200	66702	1281	500	1928.62	336	82
H_100	P_200	Low	1400	77986	1267	483	2013.6	392	82
H_100	P_200	Low	1600	89422	1247	462	2357.92	449	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Resch. No.**	**MC. Utilization**
H_100	P_800	Low	200	12150	1206	423	124.02	17	76
H_100	P_800	Low	400	23953	1439	633	292.15	32	78
H_100	P_800	Low	600	35456	1358	563	347.4	49	77
H_100	P_800	Low	800	45832	1424	629	583.77	62	79
H_100	P_800	Low	1000	55823	1538	736	858.32	74	81
H_100	P_800	Low	1200	67093	1543	737	1020.02	89	81
H_100	P_800	Low	1400	78340	1533	725	1152.52	103	82
H_100	P_800	Low	1600	89814	1513	704	1355.12	118	82

Table 4.3. Comparison of different scheduling policies for F-LOW (Cont'd)

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	Resch. No.	MC. Utilization
H_100	A_1	Low	200	11543	848	160	328.7	208	79
H_100	A_1	Low	400	23648	1043	311	1163.42	417	79
H_100	A_1	Low	600	35122	977	260	1414.43	618	78
H_100	A_1	Low	800	45482	1056	325	3015.05	829	80
H_100	A_1	Low	1000	55865	1175	426	4993.65	1037	82
H_100	A_1	Low	1200	66621	1208	455	6301.68	1209	82
H_100	A_1	Low	1400	77812	1180	428	7309.37	1427	82
H_100	A_1	Low	1600	89156	1152	402	6876.58	1622	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Resch. No.**	**MC. Utilization**
H_100	A_5	Low	200	11830	972	230	90.1	43	78
H_100	A_5	Low	400	23757	1159	387	358.92	83	79
H_100	A_5	Low	600	35295	1090	331	426.55	125	78
H_100	A_5	Low	800	45432	1155	388	823.22	166	80
H_100	A_5	Low	1000	55718	1250	473	1478.85	208	82
H_100	A_5	Low	1200	66817	1278	496	1659.92	243	82
H_100	A_5	Low	1400	77902	1257	473	1815.08	287	82
H_100	A_5	Low	1600	89299	1248	462	1993.8	326	82

Table 4.4. Comparison of when to schedule policies for H_50

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	Sch. No.	Av. Sch. L.	MC. Utilization
H_50	A_1	Low	200	11784	936	219	193.63	214	54	79
H_50	A_1	Low	400	23809	1160	401	701.32	415	57	79
H_50	A_1	Low	600	35101	1084	346	844.38	611	57	78
H_50	A_1	Low	800	45475	1156	406	1608.27	821	55	80
H_50	A_1	Low	1000	55863	1277	513	2901.48	1030	54	82
H_50	A_1	Low	1200	66741	1313	543	3495.1	1202	55	82
H_50	A_1	Low	1400	78206	1301	530	4009.63	1429	54	82
H_50	A_1	Low	1600	89635	1299	525	4328.33	1641	55	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_50	P_55	Low	200	11736	965	242	178.48	213	55	79
H_50	P_55	Low	400	23736	1165	408	626.1	431	54	79
H_50	P_55	Low	600	35153	1096	358	807.27	639	54	78
H_50	P_55	Low	800	45440	1158	411	1401.8	826	54	80
H_50	P_55	Low	1000	55948	1286	525	2491.63	1017	54	82
H_50	P_55	Low	1200	66670	1327	559	3080.27	1212	54	82
H_50	P_55	Low	1400	78049	1316	546	3462.68	1419	54	82
H_50	P_55	Low	1600	89357	1297	526	3696.68	1625	54	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_50	R_14	Low	200	11778	977	243	192.22	264	44	78
H_50	R_14	Low	400	23711	1173	410	521.17	470	50	79
H_50	R_14	Low	600	35270	1112	363	659.25	732	48	78
H_50	R_14	Low	800	45593	1179	422	1047.1	885	51	80
H_50	R_14	Low	1000	55933	1307	538	1657.02	997	56	82
H_50	R_14	Low	1200	66683	1334	561	1938.3	1196	55	82
H_50	R_14	Low	1400	78124	1313	537	2175.07	1389	56	82
H_50	R_14	Low	1600	89811	1302	524	2397.72	1588	56	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_50	A_3	Low	200	11869	1060	305	102.53	71	163	78
H_50	A_3	Low	400	24081	1346	550	396.85	139	172	78
H_50	A_3	Low	600	35343	1264	481	505.03	205	171	78
H_50	A_3	Low	800	45339	1289	504	792.43	273	165	80
H_50	A_3	Low	1000	55714	1369	576	1183.62	343	162	82
H_50	A_3	Low	1200	66975	1400	601	1428.83	402	166	82
H_50	A_3	Low	1400	78235	1376	576	1621.87	476	164	82
H_50	A_3	Low	1600	89699	1360	560	1733.73	541	165	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_50	P_165	Low	200	11839	1014	259	62	71	165	78
H_50	P_165	Low	400	23789	1240	452	275.92	144	165	79
H_50	P_165	Low	600	35139	1145	377	319.95	212	165	78
H_50	P_165	Low	800	45541	1213	442	527.73	276	165	80
H_50	P_165	Low	1000	55789	1318	536	843.43	338	165	82
H_50	P_165	Low	1200	66676	1351	565	1017.6	404	165	82
H_50	P_165	Low	1400	78175	1337	549	1154.25	473	165	82
H_50	P_165	Low	1600	89585	1328	539	1280.23	542	165	82

Table 4.4. Comparison of when to schedule policies for H_50 (Cont'd)

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	Sch. No.	Av. Sch. L.	MC. Utilization
H_50	R_43	Low	200	11966	989	249	67.23	88	134	77
H_50	R_43	Low	400	23813	1263	481	180.98	147	161	78
H_50	R_43	Low	600	35209	1161	399	238.27	240	146	78
H_50	R_43	Low	800	45513	1210	444	407.37	292	155	80
H_50	R_43	Low	1000	55770	1301	521	574.05	332	167	82
H_50	R_43	Low	1200	66994	1344	557	669	392	170	82
H_50	R_43	Low	1400	78032	1325	536	763.8	452	172	82
H_50	R_43	Low	1600	89617	1315	525	844.27	518	172	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_50	A_1	High	200	11330	583	36	167.43	204	55	84
H_50	A_1	High	400	22873	621	46	352.43	401	57	82
H_50	A_1	High	600	34923	631	49	544.8	606	57	82
H_50	A_1	High	800	45379	758	134	2502.88	822	55	85
H_50	A_1	High	1000	55208	840	182	3601.35	1020	54	87
H_50	A_1	High	1200	66322	836	177	3937.85	1207	54	87
H_50	A_1	High	1400	77179	820	162	4143.38	1417	54	87
H_50	A_1	High	1600	88763	797	146	4270.68	1623	54	87
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_50	P_57	High	200	11285	568	34	128.88	202	55	84
H_50	P_57	High	400	22878	613	51	271.87	416	54	82
H_50	P_57	High	600	34946	621	53	417.82	634	55	82
H_50	P_57	High	800	45410	749	134	1909.72	818	55	85
H_50	P_57	High	1000	55281	845	194	2965.38	991	55	87
H_50	P_57	High	1200	66354	853	197	3409.48	1195	55	87
H_50	P_57	High	1400	77250	830	177	3566.95	1387	55	87
H_50	P_57	High	1600	88729	811	160	3717.37	1591	55	87
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_50	R_35	High	200	11274	562	25	90.03	240	46	85
H_50	R_35	High	400	22797	591	33	180.43	490	46	82
H_50	R_35	High	600	35008	599	36	274.77	739	47	82
H_50	R_35	High	800	45068	703	94	729.3	839	53	85
H_50	R_35	High	1000	54794	739	106	925.48	950	57	87
H_50	R_35	High	1200	66300	726	96	1014.75	1181	56	87
H_50	R_35	High	1400	77249	718	88	1115.22	1367	56	87
H_50	R_35	High	1600	88643	706	80	1205.18	1584	55	87
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_50	A_3	High	200	11580	734	71	103.23	70	165	83
H_50	A_3	High	400	23113	818	120	255	134	171	81
H_50	A_3	High	600	35170	810	107	416.5	204	171	82
H_50	A_3	High	800	45375	905	180	1274.42	273	165	85
H_50	A_3	High	1000	54973	926	192	1572.53	338	162	87
H_50	A_3	High	1200	66898	915	176	1667.08	402	166	86
H_50	A_3	High	1400	77372	898	163	1806.7	469	164	87
H_50	A_3	High	1600	88927	891	154	1952.48	538	165	87

Table 4.4. Comparison of when to schedule policies for H_50 (Cont'd)

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	Sch. No.	Av. Sch. L.	MC. Utilization
H_50	P_165	High	200	11403	691	56	133.03	69	165	84
H_50	P_165	High	400	22998	721	62	240.12	139	165	81
H_50	P_165	High	600	35112	731	63	381.53	212	165	82
H_50	P_165	High	800	45299	830	132	930.98	274	165	85
H_50	P_165	High	1000	54948	860	150	1158.43	333	165	88
H_50	P_165	High	1200	66520	842	135	1251.93	403	165	87
H_50	P_165	High	1400	77393	829	124	1372.03	469	165	87
H_50	P_165	High	1600	88793	813	112	1468.82	538	165	87
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_50	R_93	High	200	11303	608	35	65.35	82	137	84
H_50	R_93	High	400	22861	637	39	145.55	166	137	82
H_50	R_93	High	600	35062	662	48	246.18	246	141	82
H_50	R_93	High	800	45260	764	113	557.15	282	160	86
H_50	R_93	High	1000	54868	799	127	733.88	321	170	88
H_50	R_93	High	1200	66339	779	112	804.62	397	167	87
H_50	R_93	High	1400	77208	770	104	894.2	459	167	87
H_50	R_93	High	1600	88703	756	93	966.38	530	167	87

Table 4.5. Comparison of when to schedule policies for H_100

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	Sch. No.	Av. Sch. L.	MC. Utilization
H_100	A_1	Low	200	11543	848	160	389.53	208	55	79
H_100	A_1	Low	400	23628	1039	309	1422.28	412	56	79
H_100	A_1	Low	600	35020	973	261	1772.07	608	57	78
H_100	A_1	Low	800	45441	1047	322	3592.85	820	55	80
H_100	A_1	Low	1000	55842	1175	432	6057.43	1030	54	82
H_100	A_1	Low	1200	66780	1206	457	7010.72	1202	55	82
H_100	A_1	Low	1400	77945	1193	440	7686.63	1424	54	82
H_100	A_1	Low	1600	89419	1179	424	8150.1	1618	55	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_100	P_55	Low	200	11658	874	175	397.37	211	55	79
H_100	P_55	Low	400	23621	1098	359	1578.05	429	55	79
H_100	P_55	Low	600	35099	1024	301	1858.22	638	54	78
H_100	P_55	Low	800	45445	1090	357	3263.07	826	54	80
H_100	P_55	Low	1000	55665	1184	434	5176.7	1012	54	82
H_100	P_55	Low	1200	66642	1218	461	6344.88	1212	54	82
H_100	P_55	Low	1400	77884	1196	440	6979.2	1416	54	82
H_100	P_55	Low	1600	89439	1175	419	7532.17	1626	54	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_100	R_7	Low	200	11800	883	180	369.73	274	42	79
H_100	R_7	Low	400	23646	1104	362	1017.48	493	48	79
H_100	R_7	Low	600	36166	1018	296	1287.5	802	43	78
H_100	R_7	Low	800	45530	1109	374	2258.77	958	47	80
H_100	R_7	Low	1000	55721	1212	460	3484.63	1080	51	82
H_100	R_7	Low	1200	66759	1244	485	4198.3	1281	52	82
H_100	R_7	Low	1400	77943	1223	461	4704.18	1480	52	82
H_100	R_7	Low	1600	89257	1201	439	5107.83	1713	52	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_100	A_3	Low	200	11600	911	177	133.33	70	165	79
H_100	A_3	Low	400	23669	1124	359	540.28	137	171	79
H_100	A_3	Low	600	35134	1050	302	650.32	203	172	78
H_100	A_3	Low	800	45270	1133	377	1158.9	273	165	80
H_100	A_3	Low	1000	55698	1232	462	1808.22	343	162	82
H_100	A_3	Low	1200	66723	1242	468	2130.22	400	165	82
H_100	A_3	Low	1400	77929	1222	448	2404.08	474	164	82
H_100	A_3	Low	1600	89425	1207	433	2601.07	539	165	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_100	P_165	Low	200	11659	939	218	198.33	70	165	79
H_100	P_165	Low	400	23547	1173	414	684.87	142	165	79
H_100	P_165	Low	600	35152	1088	342	822.78	212	165	78
H_100	P_165	Low	800	45400	1175	419	1509.03	275	165	80
H_100	P_165	Low	1000	55715	1271	502	2262.03	337	165	82
H_100	P_165	Low	1200	66835	1267	494	2611.15	405	165	82
H_100	P_165	Low	1400	77965	1255	479	2972.77	472	165	82
H_100	P_165	Low	1600	89585	1240	464	3214.55	542	165	82

Table 4.5. Comparison of when to schedule policies for H_100 (Cont'd)

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	Sch. No.	Av. Sch. L.	MC. Utilization
H_100	R_23	Low	200	11630	924	202	120.28	86	134	79
H_100	R_23	Low	400	23675	1120	359	332.68	156	150	79
H_100	R_23	Low	600	35163	1042	301	419.57	139	139	78
H_100	R_23	Low	800	45336	1115	363	721.07	150	150	80
H_100	R_23	Low	1000	55947	1222	454	1172.6	164	164	82
H_100	R_23	Low	1200	66817	1242	471	1426.48	163	163	82
H_100	R_23	Low	1400	78111	1225	451	1643.18	166	166	82
H_100	R_23	Low	1600	89623	1210	437	1802.6	165	165	82
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_100	A_1	High	200	11241	528	32	216.83	204	54	85
H_100	A_1	High	400	22828	568	41	531.28	403	56	82
H_100	A_1	High	600	34912	578	45	863.55	614	56	82
H_100	A_1	High	800	45271	700	116	4006.07	829	54	85
H_100	A_1	High	1000	55065	780	161	5647.68	1026	53	87
H_100	A_1	High	1200	66273	776	156	6237.38	1207	54	87
H_100	A_1	High	1400	77188	758	141	6621.62	1424	54	87
H_100	A_1	High	1600	88657	736	125	6846.9	1632	54	87
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_100	P_55	High	200	11337	550	38	245.42	208	54	85
H_100	P_55	High	400	22838	584	44	521.87	425	53	82
H_100	P_55	High	600	34970	597	48	801.93	651	53	82
H_100	P_55	High	800	45240	730	132	3723.57	837	53	85
H_100	P_55	High	1000	55210	835	199	6058.4	1019	54	87
H_100	P_55	High	1200	66274	845	204	7051.28	1221	54	87
H_100	P_55	High	1400	77145	822	185	7385.63	1420	54	87
H_100	P_55	High	1600	88765	797	164	7618.42	1636	54	87
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_100	R_17	High	200	11232	544	32	159.53	247	45	85
H_100	R_17	High	400	22820	563	30	306.22	509	44	82
H_100	R_17	High	600	34928	583	38	580.92	760	45	82
H_100	R_17	High	800	45147	702	106	1828.87	853	52	85
H_100	R_17	High	1000	54797	742	121	2439.95	963	56	88
H_100	R_17	High	1200	66304	724	108	2656.12	1195	55	87
H_100	R_17	High	1400	77222	715	101	2951.33	1382	55	87
H_100	R_17	High	1600	88676	701	91	3153.83	1604	55	87
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_100	A_3	High	200	11313	567	29	137.92	68	165	85
H_100	A_3	High	400	22784	595	33	296.38	132	170	82
H_100	A_3	High	600	34954	603	36	495.35	201	173	82
H_100	A_3	High	800	45219	717	105	1781.38	272	165	86
H_100	A_3	High	1000	54913	756	120	2230.7	338	162	88
H_100	A_3	High	1200	66463	741	108	2371.67	400	165	87
H_100	A_3	High	1400	77197	732	100	2565.73	469	164	88
H_100	A_3	High	1600	88700	718	89	2714.4	537	164	88

Table 4.5. Comparison of when to schedule policies for H_100 (Cont'd)

How	When	Flex.	Jobs	Makespan	Av. Flowtime	Av. Tardiness	CPU Time	Sch. No.	Av. Sch. L.	MC. Utilization
H_100	P_168	High	200	11331	602	42	186.47	67	168	85
H_100	P_168	High	400	22812	616	40	317.13	139	163	82
H_100	P_168	High	600	35012	628	43	561.98	215	162	82
H_100	P_168	High	800	45142	743	113	2064.98	276	163	86
H_100	P_168	High	1000	54894	777	128	2492.33	334	164	88
H_100	P_168	High	1200	66414	757	112	2626.37	406	163	87
H_100	P_168	High	1400	77263	746	103	2856.95	472	163	88
H_100	P_168	High	1600	88751	732	93	2991.98	540	164	87
How	**When**	**Flex.**	**Jobs**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**	**Sch. No.**	**Av. Sch. L.**	**MC. Utilization**
H_100	R_48	High	200	11292	567	24	99.45	81	139	85
H_100	R_48	High	400	22923	600	32	207.28	162	140	82
H_100	R_48	High	600	34979	608	36	334.08	248	140	82
H_100	R_48	High	800	45052	717	99	772.73	282	159	86
H_100	R_48	High	1000	54947	746	107	946.88	323	169	88
H_100	R_48	High	1200	66335	726	93	1012.83	404	164	87
H_100	R_48	High	1400	77213	715	84	1097.85	466	165	87
H_100	R_48	High	1600	88673	704	76	1193.8	536	164	87

Table 4.6. Performances of Scheduling Policies for Different Scheduling Frequencies.

F-HIGH Full				F-HIGH Partial				F-LOW Full				F-LOW Partial			
	R	A	P		R	A	P		R	A	P		R	A	P
A_1	701	736	797	**A_1**	706	797	811	**A_1**	1201	1179	1175	**A_1**	1302	1299	1297
A_3	704	718	782	**A_3**	756	891	813	**A_3**	1210	1207	1240	**A_3**	1315	1360	1328
A_6	779	795	811	**A_6**		1304	1204	**A_6**	1268	1270	1290	**A_6**		1537	1469
A_9	893	917	928	**A_9**		1756	1620	**A_9**	1377	1375	1389	**A_9**		1824	1673
A_12	1034	1057	1075	**A_12**		2178	1991	**A_12**	1470	1495	1510	**A_12**		2173	1965
A_15	1199	1181	1221	**A_15**		2567	2427	**A_15**	1535	1555	1586	**A_15**		2505	2313

Table 4.7. Comparison of ARRIVAL and PERIODIC Policies for Different Partial Schedules

F-HIGH	P_495	A_9
H_50	1620	1759
H_60	1360	1456
H_70	1173	1250
H_90	975	962
H_100	928	917
F-LOW	**P_495**	**A_9**
H_50	1673	1824
H_60	1579	1648
H_70	1548	1539
H_80	1473	1471
H_90	1408	1423
H_100	1389	1375
F-HIGH	**P_165**	**A_3**
H_50	813	891
H_60	759	830
H_70	741	762
H_80	747	732
H_90	731	733
H_100	782	718
F-LOW	**P_165**	**A_3**
H_50	1328	1360
H_60	1296	1308
H_70	1337	1298
H_80	1298	1240
H_90	1275	1286
H_100	1240	1207

Table 4.8. Comparison of How to Schedule Policies for Partial Schedule and PV.

	PV=0	PV=0,4	PV0,4-PV0		PV=0	PV=0,4	PV0,4-PV0
F-LOW	**P_165**	**P_165**	**P_165**	**F-HIGH**	**P_165**	**P_165**	**P_165**
H_50	1424	1771	347	H_50	800	862	62
H_100	1330	1685	355	H_100	733	814	81
	PV=0	**PV=0,4**	**PV0,4-PV0**		**PV=0**	**PV=0,4**	**PV0,4-PV0**
H_50	1442	1752	310	H_50	897	944	47
H_100	1314	1663	349	H_100	717	796	79
	PV=0	**PV=0,4**	**PV0,4-PV0**		**PV=0**	**PV=0,4**	**PV0,4-PV0**
F-LOW	**R**	**R**	**R**	**F-HIGH**	**R**	**R**	**R**
H_50	1444	1726	282	H_50	773	870	97
H_100	1316	1649	333	H_100	706	829	123
F-LOW	**H_100 - H_50**			**F-HIGH**	**H_100 - H_50**		
PV	**A**	94	**R**	**PV**	**A**	**P**	**R**
0	128	81	128	0	180	67	67
0.4	89	86	77	0.4	148	48	41

	PV=0	PV=0,4	PV0,4-PV0		PV=0	PV=0,4	PV0,4-PV0
F-LOW	**P_495**	**P_495**	**P_495**	**F-HIGH**	**P_495**	**P_495**	**P_495**
H_50	1749	2118	369	H_50	1594	1630	36
H_100	1491	2077	586	H_100	939	1151	212
	PV=0	**PV=0,4**	**PV0,4-PV0**		**PV=0**	**PV=0,4**	**PV0,4-PV0**
F-LOW	**A_9**	**A_9**	**A_9**	**F-HIGH**	**A_9**	**A_9**	**A_9**
H_50	1868	2178	310	H_50	1710	1746	36
H_100	1430	2023	593	H_100	908	1109	201
F-LOW	**H_100 - H_50**			**F-HIGH**	**H_100 - H_50**		
PV	**A**	**P_495**	**R**	**PV**	**A**	**P**	**R**
0	438	258	-	0	802	655	-
0.4	155	41	-	0.4	637	479	-

Table 4.9. Performance measures for Machine breakdown for F-LOW and F-HIGH.

	No Br.	Br.		No Br.	Br.
F-LOW	**P_495**	**P_495**	**F-HIGH**	**P_495**	**P_495**
H_50	1749	3628	H_50	1594	2279
H_100	1491	4237	H_100	939	2381
	No Br.	**Br.**		**No Br.**	**Br.**
F-LOW	**A_9**	**A_9**	**F-HIGH**	**A_9**	**A_9**
H_50	1868	3969	H_50	1710	2861
H_100	1430	4162	H_100	908	2366

	No Br.	Br.		No Br.	Br.
F-LOW	**P_165**	**P_165**	**F-HIGH**	**P_165**	**P_165**
H_50	1424	3262	H_50	800	2022
H_100	1330	3452	H_100	733	2048
	No Br.	**Br.**		**No Br.**	**Br.**
F-LOW	**A_3**	**A_3**	**F-HIGH**	**A_3**	**A_3**
H_50	1442	3097	H_50	897	2075
H_100	1314	3345	H_100	717	1978
	No Br.	**Br.**		**No Br.**	**Br.**
F-LOW	**R**	**R**	**F-HIGH**	**R**	**R**
H_50	1444	3182	H_50	773	1889
H_100	1316	3208	H_100	706	1962

Table 4.10. The Results of Dispatch Policy

Flex	Jobs	Queue	PV	BD	Makespan	Av. Flowtime	Av. Tardiness	CPU Time
F-HIGH	200	10	0	0	11120	706	53	0.42
F-HIGH	400	10	0	0	21486	753	67	0.82
F-HIGH	600	10	0	0	32950	748	81	1.23
F-HIGH	800	10	0	0	43820	754	88	1.63
F-HIGH	1000	10	0	0	54220	758	88	2.03
F-HIGH	1200	10	0	0	65847	744	80	2.43
F-HIGH	1400	10	0	0	76829	738	76	2.81
F-HIGH	1600	10	0	0	87672	743	73	3.22
Flex	**Jobs**	**Queue**	**PV**	**BD**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**
F-HIGH	200	100	0	0	11332	628	31	0.44
F-HIGH	400	100	0	0	22913	647	40	0.8
F-HIGH	600	100	0	0	35104	664	46	1.25
F-HIGH	800	100	0	0	45118	754	100	1.63
F-HIGH	1000	100	0	0	54840	780	118	2.08
F-HIGH	1200	100	0	0	66453	766	107	2.4
F-HIGH	1400	100	0	0	77316	761	102	2.83
F-HIGH	1600	100	0	0	88684	748	94	3.25
Flex	**Jobs**	**Queue**	**PV**	**BD**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**
F-HIGH	200	10	0.4	0	11044	716	67	0.43
F-HIGH	400	10	0.4	0	21422	759	96	0.83
F-HIGH	600	10	0.4	0	32926	753	91	1.26
F-HIGH	800	10	0.4	0	43738	749	91	1.64
F-HIGH	1000	10	0.4	0	54251	750	89	2.03
F-HIGH	1200	10	0.4	0	65842	741	82	2.46
F-HIGH	1400	10	0.4	0	76888	737	79	2.84
F-HIGH	1600	10	0.4	0	87594	732	75	3.26
Flex	**Jobs**	**Queue**	**PV**	**BD**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**
F-HIGH	200	10	0	4	11524	1012	262	0.65
F-HIGH	400	10	0	4	22205	1152	402	1.07
F-HIGH	600	10	0	4	33987	1281	523	1.63
F-HIGH	800	10	0	4	44540	1375	608	2.09
F-HIGH	1000	10	0	4	55290	1408	638	2.44
F-HIGH	1200	10	0	4	66556	1424	650	3.02
F-HIGH	1400	10	0	4	77523	1434	656	3.54
F-HIGH	1600	10	0	4	88474	1430	651	3.88
Flex	**Jobs**	**Queue**	**PV**	**BD**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**
F-LOW	200	10	0	0	11593	1104	327	0.47
F-LOW	400	10	0	0	22533	1241	464	0.85
F-LOW	600	10	0	0	34235	1169	402	1.25
F-LOW	800	10	0	0	44960	1219	444	1.66
F-LOW	1000	10	0	0	55560	1268	486	2.12
F-LOW	1200	10	0	0	76977	1294	507	2.56
F-LOW	1400	10	0	0	78035	1338	547	2.96
F-LOW	1600	10	0	0	88560	1358	565	3.46

Table 4.10. The Results of Dispatch Policy (Cont'd)

Flex	Jobs	Queue	PV	BD	Makespan	Av. Flowtime	Av. Tardiness	CPU Time
F-LOW	200	100	0	0	11800	929	199	0.43
F-LOW	400	100	0	0	23669	1043	281	0.8
F-LOW	600	100	0	0	35346	1033	275	1.22
F-LOW	800	100	0	0	45427	1086	327	1.65
F-LOW	1000	100	0	0	55727	1189	417	2.08
F-LOW	1200	100	0	0	67031	1237	458	2.45
F-LOW	1400	100	0	0	78182	1245	465	2.88
F-LOW	1600	100	0	0	89363	1269	483	3.35
Flex	**Jobs**	**Queue**	**PV**	**BD**	**Makespan**	**Av. Flowtime**	**Av. Tardiness**	**CPU Time**
F-LOW	200	10	0.4	0	11911	1269	503	0.46
F-LOW	400	10	0.4	0	22671	1429	647	0.88
F-LOW	600	10	0.4	0	34287	1360	580	1.28
F-LOW	800	10	0.4	0	44895	1374	589	1.73
F-LOW	1000	10	0.4	0	55752	1399	608	2.13
F-LOW	1200	10	0.4	0	67548	1435	639	2.59
F-LOW	1400	10	0.4	0	78333	1453	654	3.09
F-LOW	1600	10	0.4	0	88375	1477	677	3.55

Table 4.11. The Performances of scheduling policies with and without PV.

POLICY	HOW	PV	F-LOW	F-HIGH
P_165	FULL	0	1330	733
A_3	FULL	0	1314	717
RATIO	FULL	0	1316	706
DISPATCH	FULL	0	1358	743
P_165	FULL	0.4	1685	814
A_3	FULL	0.4	1663	796
RATIO	FULL	0.4	1649	829
DISPATCH	FULL	0.4	1477	732
P_165	PARTIAL	0	1424	800
A_3	PARTIAL	0	1442	897
RATIO	PARTIAL	0	1444	773
DISPATCH	PARTIAL	0	1358	743
P_165	PARTIAL	0.4	1771	862
A_3	PARTIAL	0.4	1752	944
RATIO	PARTIAL	0.4	1726	870
DISPATCH	PARTIAL	0.4	1477	732

Table 4.12. T-test for the RATIO Policy and Dispatch Rule.

FLEX.	RATIO	DISPATCH	Difference
F-LOW	1210	1310	-100
F-LOW	1316	1296	20
F-LOW	1239	1369	-130
	MEAN = 1255	MEAN = 1358	MEAN = -70
			STD. DEV. = 56.1
			INTERVAL = (-23,133)
FLEX.	RATIO	DISPATCH	Difference
F-HIGH	704	760	-56
F-HIGH	676	735	-59
F-HIGH	646	706	-60
	MEAN = 675	MEAN = 734	MEAN = -58
			STD. DEV. = 1.5
			INTERVAL = (-62.34, -53.66)

Table 4.13. T-test for the RATIO and PERIODIC Policies.

FLEX.	RATIO	PERIOD	Difference
F-LOW	1210	1240	-30
F-LOW	1316	1341	-25
F-LOW	1239	1238	1
	MEAN = 1255	MEAN = 1273	MEAN = -18
			STD. DEV. = 10.97
			INTERVAL = (-50,14)
FLEX.	RATIO	PERIOD	Difference
F-HIGH	704	732	-28
F-HIGH	676	702	-26
F-HIGH	646	675	-29
	MEAN = 675	MEAN = 703	MEAN = -28
			STD. DEV. = 1.12
			INTERVAL = (-31.26, -24.74)