

**DOKUZ EYLÜL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED
SCIENCES**

**FLEXIBLE JOB SHOP SCHEDULING VIA
SIMULATION UNDER SEQUENCE
DEPENDENT SETUP TIMES**

**by
Erkin BÜYÜKKÖPRÜ**

September, 2005

İZMİR

FLEXIBLE JOB SHOP SCHEDULING VIA SIMULATION UNDER SEQUENCE DEPENDENT SETUP TIMES

**A Thesis Submitted to the
Graduate School of Natural and Applied Sciences of
Dokuz Eylül University
In Partial Fulfillment of the Requirements for
the Degree of Master of Science in Industrial Engineering Program**

**by
Erkin BÜYÜKKÖPRÜ**

**September, 2005
İZMİR**

M.Sc THESIS EXAMINATION RESULT FORM

We have read the thesis entitled **“FLEXIBLE JOB SHOP SCHEDULING VIA SIMULATION UNDER SEQUENCE DEPENDENT SETUP TIMES”** completed by **ERKİN BÜYÜKKÖPRÜ** under supervision of **Ph.D. ÖZCAN KILINÇCI** and we certify that in our opinion it is fully adequate, in scope and in quality, as a thesis for the degree of Master of Science.

.....

Supervisor

.....

(Jury Member)

.....

(Jury Member)

Prof.Dr. Cahit HELVACI
Director
Graduate School of Natural and Applied Sciences

ACKNOWLEDGEMENTS

I would like to thank my advisor Ph.D. Özcan KILIÇÇI for his support and guidance throughout the preparation of this thesis.

I would also thank to Levent Dinçer the Production Manager of TEBA for his help and guidance in the application stage of this study in the plant.

I am thankful to my mother and father for their long term encouragement and support and their trust in me.

Erkin BÜYÜKKÖPRÜ

FLEXIBLE JOB SHOP SCHEDULING VIA SIMULATION UNDER SEQUENCE DEPENDENT SETUP TIMES

ABSTRACT

The impact of setup on manufacturing performance, and the scarce research about the scheduling of the flexible job shop with sequence dependent setup times have motivated this research. A flexible job shop is an extension of the classical job shops in which there is more than one identical machine for the same purpose. The problem is to assign each operation to a machine (routing) and to order the operations on the machines (sequencing) such that the desired objectives are satisfied.

The flexible job shop scheduling problem is NP-hard and by the increase of problem size, an enormous computational effort is needed for an optimal solution. So there should be made a trade off between solution quality and time consumption. As this study is based on the observation of effects of sequence dependent setup times over system performances, it is not desired to reach an optimal solution.

The observation of the effects of sequence dependent setup times is based on the comparisons of two types of heuristic groups, one of which considers the setups in scheduling process, meanwhile other doesn't. In the considered manufacturing framework, setup is a vital issue whose duration depends on both the job just processed and the job to be processed. Also, there is only one mould for each specific operation so two parts, which need the same operation in a shop, cannot be processed at the same time at different parallel machines. Under these restrictions and high threat of long and frequent setup times, an effective solution which may contribute to the reduction of setup time and to an efficient usage of production time is needed.

The production environment which is a press work center located in an oven producing plant is modeled using the simulation software ARENA. Experimental design is used to see the effects of setups over performance measures such as mean tardiness, flow time, proportion of tardiness and work in process. This thesis

emphasizes the influence of the setup time over the considered performance measures, the reaction of proposed scheduling rules to setup time changes and the importance of the evaluation of the setup times separately from the processing times as an additional constraint in the objective function.

Keywords: Flexible job-shop scheduling, sequence dependent setup, dispatching rules, simulation

SİMÜLASYON İLE SIRAYA DAYALI KURULUM ZAMANLARININ VARLIĞINDA ESNEK ATÖLYE TİPİ ÇİZELGELEME

ÖZ

Kurulum işleminin genel üretim performansına etkisi ve sıraya bağlı kurulum sürelerinin olduğu esnek atölye tipi üretimdeki çizelgeleme konusundaki çalışmaların azlığı bu tezin yapılmasında esin kaynağı olmuştur. Esnek atölye tipi üretim, klasik atölye tipi üretimin genişletilmiş halidir. Bu üretim tipinde aynı amaç için aynı özelliklere sahip birden fazla makine bulunmaktadır. Problem, işlerin makinelerle atanması (rotalama) ve işlerin istenilen amaçlara hizmet edecek şekilde makinelerde sıralanması (sıralama) şeklinde tanımlanabilir.

Esnek atölye tipi üretim çizelgeleme problemi zor deterministik olmayan polinomial zamanlı algoritma sınıfındadır ve problem boyutunun artmasıyla optimal çözüme ulaşmak çok fazla hesaplama gerektiren bir uğraşıya dönmektedir. Bu nedenle çözüm kalitesi ve harcanan zaman arasında bir denge sağlanmalıdır. Bu çalışma sıraya dayalı kurulum zamanlarının sistem performansları üzerindeki etkilerinin gözlenmesi üzerine dayandığı için optimal bir çözüm aranmamaktadır.

Sıraya dayalı kurulum zamanlarının etkileri üzerine yapılan gözlem iki sezgisel grubun karşılaştırılması üzerine dayanmaktadır. Bir grupta kurulum çizelgeleme sürecinde göz önünde tutulurken diğer grupta göz ardı edilmiştir. Üzerinde durulan üretim yapısında kurulum süresi hem henüz işlenmiş olan hem de arkasından işlenecek olan işlere bağlı olan önemli bir unsurdur. Ayrıca her kalıp tipinden sadece tek adet olması, aynı işleme gereksinim duyan iki parçanın paralel, ayrı makinelerde işlenmesine engel oluşturmaktadır. Bu kısıtlar altında ve sık ve uzun kurulum zamanlarının tehdidi altında kurulum zamanının düşürülmesine ve üretim zamanının etkin kullanımına katkısı olacak etkin bir çözüme ihtiyaç vardır.

Çalışmaya konu olan fırın fabrikasının pres departmanının modeli ARENA adlı simülasyon programıyla modellenmiştir. Akış zamanı, ortalama gecikme, gecikme

orani ve süreç içersindeki parça sayısı gibi performans değişkenleri üzerinde kurulum zamanının ne gibi etkilerinin olduğunu gözlemlemek için Deneysel Dizayn uygulanmıştır. Bu tezle kurulum zamanının yukarıda sayılan değişkenler üzerindeki etkilerinin varlığına, tanımlanan çizelgeleme algoritmalarının kurulum zamanlarındaki değişimlere karşı gösterdiği tepkiye ve kurulum zamanlarının parça işleme zamanlarından ayrı olarak amaç fonksiyonunda ek bir kısıt olarak değerlendirilmesinin önemine vurgu yapılmıştır.

Anahtar Kelimeler: Esnek atölye tipi çizelgeleme, sıraya bağlı kurulum, dağıtım kuralları, simülasyon

CONTENTS

	Page
THESIS EXAMINATION RESULT FORM.....	ii
ACKNOWLEDGEMENTS	iii
ABSTRACT	iv
ÖZ.....	vi
 CHAPTER ONE INTRODUCTION	 1
1.1. Motivation of the Research	1
1.2. Purpose	2
1.3. Problem Complexity.....	3
1.4. Thesis Outline.....	3
 CHAPTER TWO REVIEW OF SCHEDULING	 5
2.1. Introduction	5
2.2. Principles of Scheduling.....	7
2.3. Objectives of Scheduling Problems.....	8
2.4. General Assumptions in Machine Scheduling	9
2.5. Classification of Scheduling Systems	12
2.6. Methodologies for Solving the General Scheduling Problem.....	13
2.6.1. Optimization Using Mathematical Techniques	14
2.6.1.1. Linear Programming.....	14
2.6.1.2. Branch and Bound Techniques.....	14
2.6.2. Approximation Methods.....	15
2.6.2.1. Dispatching Rules.....	16
2.6.2.2. Artificial intelligence (AI) Techniques	18
2.6.2.3. Neighborhood Search Methods	19
2.6.2.3.1. Random Search.....	20
2.6.2.3.2. Hill Climbing.....	20
2.6.2.3.3. Simulated Annealing	21
2.6.2.3.4. Tabu Search	22
2.6.2.3.5. Genetic Algorithms	23

2.6.3. Simulation Modeling.....	24
2.6.3.1. Use of Simulation in Scheduling.....	25
2.6.3.2. New Emerging Role of Simulation in Manufacturing	27
2.6.3.3. Applications of Simulation Modeling	28
2.6.3.4. Advantages of Simulation Modeling.....	29
CHAPTER THREE LITERATURE REVIEW	31
3.1. Introduction	31
3.1.1. Single Machine Problems.....	34
3.1.2. Parallel Machine Problems.....	36
3.1.3. Flowshop Problems	39
3.1.4. Job shop Problems.....	42
CHAPTER FOUR APPLICATION OF PROBLEM	53
4.1. Introduction	53
4.2. The Environment of Implementation	53
4.2.1. Company Information	53
4.2.3. Description of Press Workshop.....	54
4.3. Assumptions	58
4.4. Constraints.....	59
4.5. The System Modeling and Simulation	59
4.6. Decomposition of the Problem.....	68
4.6.1. Input Analysis.....	68
4.6.2. Verification and Validation	70
4.6.3. Defining Warm up Period	72
4.6.4. Variance Reduction Techniques.....	74
4.7. Evaluation of Proposed Methods.....	74
4.7.1. Analysis of Experimental Results	75
4.7.1.1. Flow Time	76
4.7.1.2. Mean Tardiness	80
4.7.1.3. Proportion of Tardiness	85
4.7.1.4. WIP	88
4.7.2. Summary of Statistical Analysis	91

CHAPTER FIVE CONCLUSION	94
5.1 Conclusion.....	94
5.2. Further Research.....	98
 REFERENCES	 100
APPENDIX A : Input Analysis of Batch Sizes.....	107
APPENDIX B : Part Types, Routes and Mean Process Times	110
APPENDIX C: Statistical Analysis of Dispatching Rules for Each Performance Measure	112
APPENDIX D : SIMAN Codes of Policy 1	116

CHAPTER ONE

INTRODUCTION

1.1. Motivation of the Research

The current need for shop performance improvements is tied to the issue of competitiveness. There is also a realization that to evaluate shop performance improvements, researchers must include the shop factors which are relevant. Recently, sources from industry and academia have suggested some factors. One of the common factors shared in the studies is setup time and in particular sequence dependent setups. A sequence dependent setup recognizes that the setup time of a job is a function of the preceding job on the machine and therefore the overall sequence.

There are numerous industrial systems that serve as examples for which the amount of setup time varies considerably depending on the processing sequence of the jobs. These include the cutting and stitching machines in commercial printing firm, the stamping operation in plastics manufacturing, die changing in metal processing, and roll slitting in paper industry.

The usual assumption for most job shop scheduling research is that jobs are sequence independent. The result of this assumption is that capacity management in these studies requires only the allocation of capacity over the specified time interval; since the sum of setup and run times remains constant for instances of sequence independence, capacity utilization also remains constant.

The purpose of this study is to focus on the effects of sequence dependent setups on different performance measures such as flow time, WIP, tardiness and proportion of tardiness. Sequence dependent setups have significant implications on shop performance beyond those directly affecting the shop capacity. Firstly, setup time constitutes a part of the flow time that directly affects the throughput rate of a production system. Secondly, the cost of unit setup time is usually higher than that of unit processing time since the cost of setup time includes loss of productivity due to machine down-time and the cost of setup itself such as labor cost of a setup

technician. Thirdly, the setup change can often be made only by a qualified technician since setup is often a complicated work requiring a higher level of expertise. The qualified personnel are limited resources that are not always available.

1.2. Purpose

The thesis has been carried out in the press department of an oven producing plant. The press workshop can be accepted as a flexible job shop in which there are more than one machine each of which has the same capability in terms of working speed and machine capabilities. Totally 15 main types of products are produced with many sub types for each main version. As a result of high variety of products, setup activities are required frequently in a way that the efficient production time is decreased. Considering this constraint, the main purpose of this thesis is to find an efficient way to decrease the total time spent for setup activity and to evaluate the difference on performance measures between the situations when the sequence dependent setups are considered or not.

To decrease the time spent for setups, it is clear that the same types should be processed consecutively. But it is not also enough to consider the problem as only reducing the setups. Because all orders have tight due dates that must be met. If only it is desired to reduce the setup times by processing the same subtypes consecutively, the risk of going beyond the desired due dates occurs. For instance, a part ,which has an urgent due date, may not be processed for a long time as subtypes of another part are processed consecutively although their due dates are less tight. This is a problem between meeting due dates and reducing setup times. The balance should be achieved in a way that while the setup times are reduced as much as available, the due dates should be also met.

Different dispatching rules are constituted and the performance of each rule is evaluated considering the objectives such as work in process, flow time, proportion of tardiness and mean tardiness. By this study, not only the effects of sequence dependent setup time over the system performances but also the reaction of the scheduling rules to the different levels of sequence dependent setups is observed.

1.3. Problem Complexity

The job shop scheduling problem (JSP) is one of the hardest discrete optimization problems. It consists of sequencing a set of jobs on a set of machines in order to minimize an objective function. Each machine processes at most one job at a time, and each operation is processed on one machine given. The flexible job shop scheduling problem (FJSSP) is NP-hard since it is an extension of the job shop scheduling problem. The NP-hardness of an optimization problem suggests that it is not always possible to find an optimal solution quickly. Especially, by the increase of problem size, an enormous computational effort is needed for an optimal solution. So there should be made a trade off between solution quality and time consumption. As this study is based on the observation of effects of sequence dependent setup times over system performances, it is not desired to reach an optimal solution.

The scheduling problem of a flexible job shop problem (FJSP) consists of a routing sub problem, that is, assigning each operation to a machine in a related set and the scheduling sub problem, which consists of sequencing the assigned operations on each machine in order to obtain a feasible schedule. There have been a few studies about scheduling at job shop environment with sequence dependent setups and most of them cannot present a highly useable algorithm that can be used for large problem sizes.

1.4. Thesis Outline

The primary goal of this thesis is to search the impact of sequence dependent setups over different performance measures and to compare the responsiveness of different dispatching rules to sequence dependent setups.

Chapter 2 gives an in-depth overview of scheduling problems with solution procedures.

Chapter 3 gives information about the content of sequence dependent setups and surveys the literature under the classification of different machine configurations.

In Chapter 4, the scheduling problem for a flexible job machines in an oven producing company is introduced. After the descriptions of problem, the assumptions and the constraints, the steps of the simulation methodology is accomplished sequentially. By the constitution of simulation models for the proposed dispatching rules, statistical analysis was done for the evaluation of dispatching rules considering different performance measures.

Finally, Chapter 5 summarizes the research work, and outlines the directions for future research.

CHAPTER TWO

REVIEW OF SCHEDULING

2.1. Introduction

Scheduling in general deals with assignment of activities to limited resources where a set of constraints has to be regarded. These constraints can be e.g. restrictions on the ordering of operations, due dates etc. Despite of many different scheduling definitions, all include similar constraints and objectives.

“Scheduling is a study that concerns the allocation of limited resources to the tasks, which can be in several forms, from service industry to manufacturing systems, or information processes” (Pinedo, 1995).

“Scheduling specifies the resources that each task needs at particular times. Any process that defines a subset of What $\times$ When $\times$ Where can be said to “do scheduling” (Parunak, 1990).

The ultimate objective of the scheduling task is to select a schedule that optimizes some pre-stated goal. In this context a schedule is formally defined as a set of start and completion times for a group of jobs on a group of machines that satisfy all problem constraints. According to Baker (1974), the three types of decision-making goals that are prevalent in scheduling are:

- Efficient utilization of resources.
- Rapid response to demands.
- Close conformance to prescribed deadlines.

A planner constitutes a scheduling function which covers the entire jobs or the group of tasks defined for the group of the jobs in the company. Scheduling function should have integration with several other important functions in the organization. Initially, it is affected by the production planning process, which handles medium and long-term planning for the entire organization. At this level, scheduling process should consider many components such as inventory levels, forecasts and resource

requirements to optimize at a higher level, the product mix and resource allocation for long term period, then it focuses on unexpected events in the shop floor such as machine breakdowns, or random processing times.

As known planning is a whole system in which scheduling is only one of the stages. The main difference between scheduling and planning under time/resource constraints is that, in scheduling we know the set of activities in advance, while in planning we have to generate the activities. This difference also explains the major interaction between traditional planning and scheduling: first plan/generate the set of activities and then schedule/allocate these activities to resources. The diagram in Figure 2.1 depicts the information flow and the place of scheduling activity in a manufacturing system.

The scheduling function uses mathematical techniques or heuristic methods to allocate those limited resources to the processing of tasks. A proper allocation of resources enables the company to optimize its objectives and achieve its goals. Resources may be machines in a workshop, runways at an airport, crews at a construction site, or processing units in a computing environment. Tasks may be operations in a workshop, takeoffs and landings at an airport or stages in a construction project. Each task may have a priority level, an earliest possible starting time, and a due date. The objectives may also take many forms, such as minimizing the time to complete all tasks or minimizing the number of tasks completed after their due dates. (Pinedo& Chao, 1999, p.2)

Although to define a scheduling problem in words is often easy, unfortunately, scheduling can be difficult to perform and implement. Since the time function goes into the scheme, solution branches grow up to a huge amount, at once. Then, implementation difficulties arise related to the modeling of the real world scheduling problems whereas technical difficulties come across the solution methodology and procedures. Resolving these difficulties takes skill and experience but is often financially and operationally well worth the effort. (Pinedo& Chao, 1999, p.5)

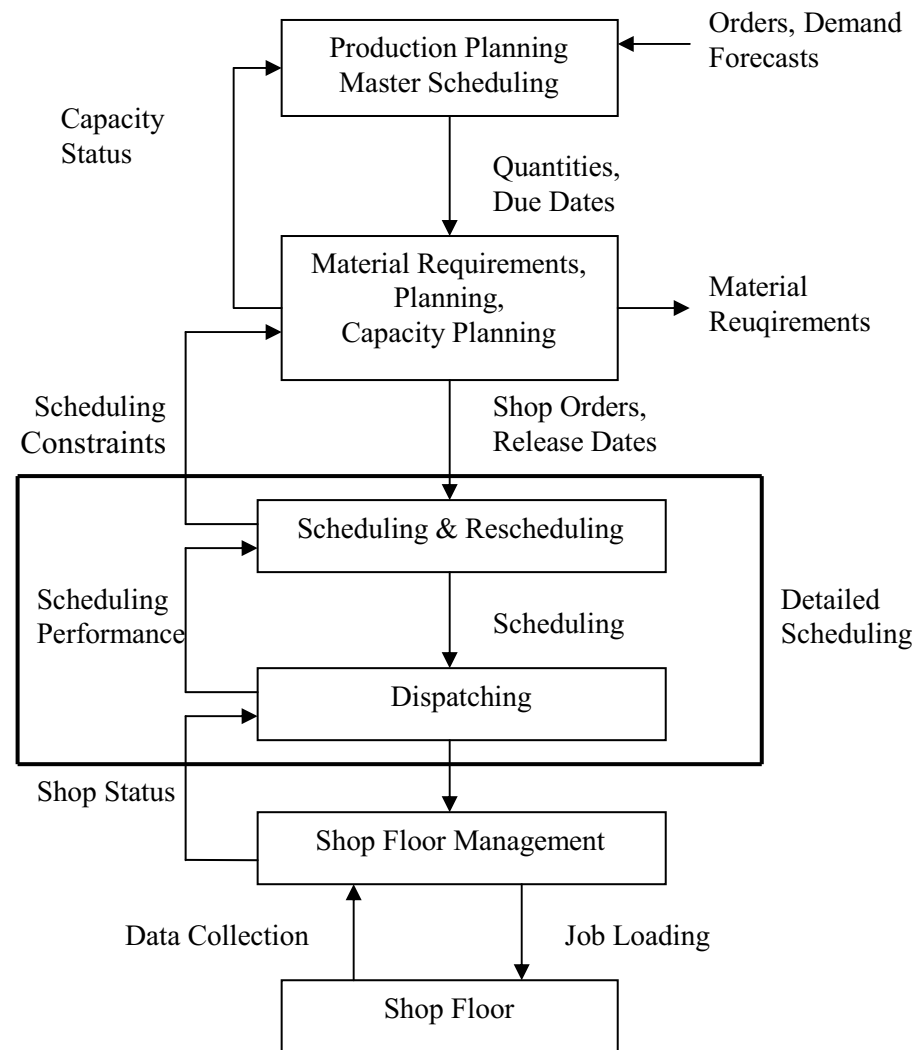


Figure 2.1 Information flow diagram in a manufacturing system
(Pinedo & Chao, 1999, p.7)

2.2. Principles of Scheduling

Scheduling is comprised of a series of sequential steps or a routing. General plans or schedules have the following inputs related with the sequential steps:

- The sequence of operations
- Necessary sequential constraints
- The time estimates for each operation
- Required resource capacities for each operation

In addition scheduling activity serves for finding the answers to the questions such as when an end item will be completed, and what jobs are to be completed during a specified time.

2.3. Objectives of Scheduling Problems

Different types of objectives may be under consideration in scheduling problems. The classical objective functions are either of the ‘minsum’ or the ‘minmax’ type.

The classical objective functions are built by considering the following elementary functions (Brandimarte, Villa, 1995, p110):

Flow time: $F_i = C_i - r_i$

Tardiness: $T_i = \max\{C_i - d_i, 0\}$

Earliness: $E_i = \max\{d_i - C_i, 0\}$

Lateness: $L_i = C_i - d_i$

The most significant minsum objective functions are:

$$\sum_{i=1}^n C_i = \text{Total completion time} \quad \sum_{i=1}^n w_i C_i = \text{Total weighted completion time}$$

$$\sum_{i=1}^n T_i = \text{Total tardiness} \quad \sum_{i=1}^n w_i T_i = \text{Total weighted tardiness}$$

$$\sum_{i=1}^n U_i = \text{Number of tardy jobs} \quad \sum_{i=1}^n w_i U_i = \text{Weighted number of tardy jobs}$$

The most significant minmax objective functions are:

$$L_{\max} = \max_i L_i = \text{maximum lateness}$$

$$T_{\max} = \max_i T_i = \text{maximum tardiness}$$

$$C_{\max} = \max_i C_i = \text{makespan}$$

r_i = Release Date; d_i = Due Date; w_i = Weight of a job; C_i = Completion time of job

The shop structure is also a part of a scheduling framework. For example, in a flow shop the jobs go through a fixed sequence of the same routing steps whereas in job shops have customized products with unique routing steps. Additionally shop structure is about its capacity. A solution may not be appropriate for each different problem sizes although the problem remains same.

The critical issues about product structures, which affect the scheduling environment, are:

- Existence of single part or assembly routings
- Type of processing time distribution
- Alternate routings
- Operation overlapping
- Lot sizes

Flexibility of the production system is an important criterion for work center capability. The extent to which capacity for a particular work center can be increased or decreased and the time delay to change the capacity both affects scheduling performance. Another issue in work center capacity is to focus on the bottlenecks. If the capacity of the bottleneck can be well utilized, then the overall scheduling performance would be improved too.

2.4. General Assumptions in Machine Scheduling

To simplify the complexity of manufacturing and service environments, most of the scheduling problems are solved under assumptions associated with job characteristics and objective functions. The following list gives a set of these assumptions and points out when some of them are inadequate to represent a realistic scheduling problem (Brandimarte & Villa, 1995, pg.104-106):

Single parts and batches of parts are always treated as a single job.

Although this assumption may be appropriate in many situations, in case of large lots, it may produce poor quality schedules. The classical machine scheduling formulation does not allow starting processing on a latter machine until all the parts in the lot are processed on the first one. However, a better schedule may be obtained by transferring a sub-batch of a part to the preceding machine before the completion of the whole batch; such sub-batches are called *transfer batches*.

Job cancellation is not allowed.

All the jobs are to be processed eventually. This is not always true; for instance, it may be useful to draw a distinction between high-priority and low-priority jobs; low priority jobs may be scheduled only if the execution of the high-priority ones leaves enough idle time on the machines.

Preemption is not allowed.

Preemption occurs when an operation is stopped and resumed at a later time. This is clearly unacceptable from the technological point of view in most manufacturing environments. However, this is not the case, when scheduling tasks on computer systems.

Each job visits all machines exactly once.

In practical problems, some machines may not be required for a certain job. In other cases, a machine may be visited more than once by the same job, e.g., reworking of defective parts.

Machines are always available.

In practice, machines may not be available because of controllable events (e.g., preventive maintenance) and uncontrollable events (e.g., failures).

Jobs are all known in advance.

This is the characteristic that distinguishes *static* and *dynamic* scheduling problems. In a dynamic scheduling environment, new jobs arrive at unpredictable times.

The problem is purely deterministic.

In practice, scheduling problems are stochastic in nature. Machine failures, unpredictable processing times, (e.g., in the case of manual operations) causes this uncertainty.

Processing times are independent of the schedule.

Essentially, this assumption implies that setup times are sequence-independent and can be included in the processing time. This is not the case when sequence-dependent setup times are to be dealt with. A classical case occurs in paint production, where sequencing a lot of white paint after the production of black paint is a very bad idea, due to the large setup time needed to clean the machine.

Machines are the only resources modeled.

In practice, it may be necessary to model additional resources such as transportation devices (e.g., automated guided vehicles), tools, or skilled labor.

Work-in process is allowed.

Jobs may wait in a queue until next machine required for processing is idle. In some metallurgical processes, this situation is not possible.

Machines are able to process one job at a time.

This is a correct assumption for many mechanical operations. Batch processors such as thermal treatment machine, PCB processors, etc violates the assumption.

Precedence constraints can be occurred.

Blocking of workstations is allowed/not allowed.

2.5. Classification of Scheduling Systems

Scheduling models and systems are classified into different categories hierarchically and these categories are summarized in Figure 2.2.

Scheduling problems are first divided into two categories as static and dynamic. In the static scheduling problem, a fixed set of jobs which are defined at the beginning of the scheduling process are to be planned. The typical assumptions are that, the entire set of jobs arriving at the same time and all the work centers being available at that time. Static scheduling problems are investigated by using either deterministic or stochastic processing times. Methods based on the deterministic times can be divided into subtypes of those producing optimum results and those utilizing heuristic scheduling procedures. As the computational difficulty increases exponentially with the problem size, optimization methods are used mostly for small problems. For more complex problems, heuristic procedures such as dispatching or sequencing rules are used.

Dynamic scheduling additionally considers the new jobs that are added during the production. Dynamic scheduling is based on the prioritization between jobs. The rules that are used in dynamic scheduling differ from the rules used in static scheduling in terms of the properties which are considered for scheduling process. Whereas in static scheduling the unchanged properties such as processing time or due dates are considered, in dynamic scheduling the properties which change during the scheduling process such as remaining process time are considered. Simulation is most commonly used for large scale dynamic scheduling cases.

The main difference between stochastic and deterministic models is the randomness of events and times. In stochastic models, machine breakdowns, unexpected orders, random process times and preemption are considered whereas in deterministic models not.

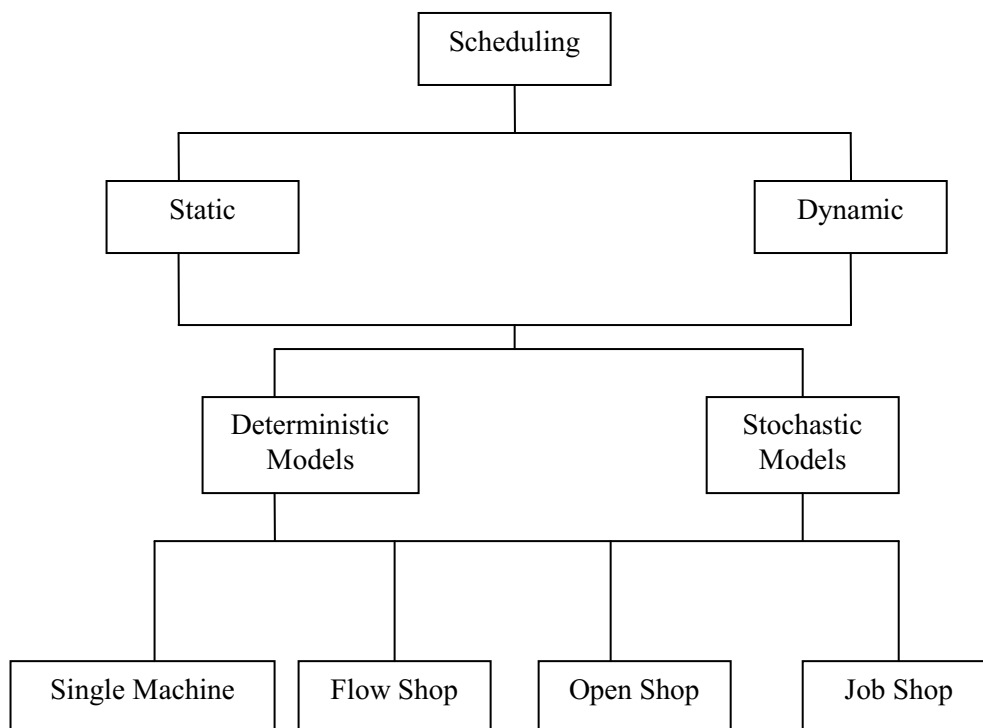


Figure 2.2 Classification of scheduling

2.6. Methodologies for Solving the General Scheduling Problem

For scheduling a high variety of different techniques are used. Generally they can be classified into two main groups according to their objective whether they consider optimization or not. The methods which seek for optimization are the most complicated ones so they require enormous computation time to reach the optimal solution. The bigger problem size is considered, more difficult to reach the optimality so they are not always applicable to all problems in real life. The approximation methods which don't look for optimal solutions have greater application areas since they require less computation time. Meanwhile the solution quality may be poor in some cases. In the following sections, considering the main classification of methods which is described above, the most common techniques are introduced. As the thesis is based on the simulation modeling, a more detailed introduction is given under its title.

2.6.1. Optimization Using Mathematical Techniques

2.6.1.1. Linear Programming

Linear programming is an optimization technique used to solve optimization problems. Problems are formulated or programmed as a series of linear expressions. These expressions are statements of constraints that bound the problem solution and the objective function.

Some scheduling problems can be formulated as linear programming problems with the objective defined in terms of the problem variables. However, as is the case with most scheduling problems of practical dimension, the computational demands can be extreme.

2.6.1.2. Branch and Bound Techniques

Branch and bound algorithms are straightforward and useful techniques for solving combinatorial problems. Two basic operations, branching and bounding are used in order to prevent complete enumeration of the solution space, and hence reduce the computational complexity of the problem. The branching procedure partitions the overall problem into a series of sub problems in a tree-like structure. The bounding procedure is used to estimate the lower bound of each “active” sub problem in an attempt to establish the likelihood that a particular sub problem will contain the optimum solution. In this manner a decision tree is dynamically created level by level. Applying branch and bound strategies at each of these levels determines the possible set of nodes on that level of the tree from which the search can continue.

As the complete problem is divided at each level, estimation of lower bounds for each sub problem allows large branches of the solution space to be removed from the investigation as the search is guided towards the base of the tree and hence find the optimal solution to the problem. The objective of branching and bounding strategies is to do this with the least possible computational effort and hence in the quickest possible time.

In terms of scheduling problems, the number of levels in the decision tree represents the number of operations in the problem. A schedule is any arrangement of these operations that satisfies all problem constraints. Each sub problem is identical to the problem at the preceding level with the exception of a single operation that has been fixed in the last available position in the emerging partial schedule. Repeating this process down through the decision tree and across its full width would constitute complete enumeration of the problem. Obviously, calculation of all possible permutations would uncover the optimal schedule but would take too long when faced with any problem of practical dimension (i.e. the fundamental problem). Branch and bound strategies seek to remove large groups of schedules that have been proven to lead to sub optimal results.

As the search advances through each level of the tree the most likely candidates for optimality begin to emerge. The optimum schedule is pieced together level-by-level until all jobs have been given appropriate positions in that schedule.

Work with enumerative techniques such as branch and bound algorithms have progressed over the years. Unfortunately however, even with tight bounding techniques these methods require excessive computational time when faced with the large dimensionality of complex practical problems. Even though success has been somewhat limited to problems of small to medium dimension, many of their strategies have been embedded in modern approximation techniques

2.6.2. Approximation Methods

Even when augmented by sophisticated rules and procedures, the most promising enumerative technique, branch and bound algorithms have failed to produce satisfactory results when faced with practical problems that, by their nature tend to be of large dimension. The realization that an enumerative technique was unlikely to ever offer a satisfactory solution to combinatorial problems brought about a paradigm shift. That's why; attention was concentrated on solving the General Scheduling Problem using approximation techniques. Unlike optimization techniques, approximation methods do not guarantee that the solution generated will optimize whatever criterion is under scrutiny. In many instances an optimum solution

to a problem is not required and a sub optimal solution is satisfactory, provided that it is within a few percent of the optimum. Approximation methods do just that.

Aggressive elimination strategies can very quickly arrive at a solution that is adequately close to the optimum. As the search technique closes in on the optimum the costs explode as gains are diminished in correspondence with the law of diminishing returns. Therefore, the cost of settling for a sub optimal solution compared with the optimum solution is much less than the cost of computing the optimum compared with the cost of computing a solution close to that optimum. This is the fundamental basis for accepting approximation algorithms as promising approaches to solving this age-old problem. The emphasis is now placed by many researchers on developing the most efficient approximation algorithm. (i.e. the algorithm that gets closest to the optimum in the shortest possible time). A more in-depth explanation of these and other techniques follows.

2.6.2.1. Dispatching Rules

Dispatching rules are the simplest form of approximation technique. Dispatching rules are a set of heuristics commonly used in scheduling. Most dispatching rules are “simple single pass priority dispatching rules....once a decision is arrived at by the operation of a rule, it is implemented without reconsideration of alternative courses of action.” (King and Spachis 1980). The choice of rule is system dependent and often determined by the objective criteria.

Job Slack (S) Give priority to the jobs with least ‘job slack’, i.e. the amount of contingency or free time, over and above the expected processing time, available before the job is completed at a pre-determined date.

$$S = t_0 - t_1 - \sum a_i$$

Where t_0 = future date

$\sum a_i$ = sum of remaining processing time

t_1 = present date

Job Slack Ratio The ratio of the total time remaining to the remaining slack time. This rule tends to minimize due date related objectives ($S/t_0 - t_1$)

Scheduled Start Date The date on which operations must be started so that a job will meet a required completion date.

Earliest Due Date Process the job with the earliest due date first. This rule tends to minimize the maximum lateness among the jobs waiting for processing.

Subsequent Processing Times Process the job with the most remaining processing time first.

Service in Random Order (SIRO) Rule The next job is selected at random from those waiting for processing. No attempt is made to optimize any objective.

Earliest Release Date First (ERD) Rule This rule is equivalent to the well-known first-come-first served rule. This rule in a sense minimizes the variation in the waiting time of the jobs at a machine.

Shortest Processing Time First (SPT) Rule This rule orders the jobs in non decreasing order of their processing times. This rule tends to minimize the sum of completion times.

Longest Processing Time First (LPT) Rule This rule orders the jobs in decreasing order of their processing times. When there are machines in parallel, this rule tends to balance the workload over the machines.

Shortest Setup Time First (SST) Rule Whenever a machine is freed, this rule selects for processing the job with the shortest setup time. This rule tends to minimize make span objective.

Least Flexible Job First (LFJ) Rule This rule is used when there are a number of non-identical machines in parallel and the jobs are subject to machine eligibility constraints. Whenever a machine is freed, the job that can be processed on the

smallest number of other machines is selected, that is, the job with the fewest processing alternatives.

Dispatching rules have been applied consistently to scheduling problems. They give quick and simple solutions that can be used to support real-time decision-making because they are not iterative procedures. Dispatching rules can be easily integrated as rules in intelligent control systems as part of a dynamic scheduler.

As would be expected, the use of dispatching rules is limited. The performance of rules varies under different conditions. Different rules perform better in some situations. Therefore, the choice of rule used depends on the problem in question. The rules themselves have been studied extensively, usually under the controlled conditions of a simulation model. The simulation model allows the performance of different rules to be compared and contrasted in different scenarios. Also, the volume of testing that can be done using a simulation model is significantly greater than could be achieved by testing in the physical system.

2.6.2.2. Artificial intelligence (AI) Techniques

The science of artificial intelligence attempts to make machines respond to tasks in ways that would normally consider intelligent if done by humans. In other words, to emulate in machines, human-like attributes such as intelligent decision-making and learning. This section discusses some of these ideas in the context of developing an automated and intelligent, dynamic scheduling system.

Rule-based expert systems are computer models in which domain knowledge relating to some specific problem is captured and represented as rules. Instead of attempting to develop a complex algorithm to represent the overall behavior of the system, the model of the system is broken down into a series of simple rules whose individual behavior combine to effectively model the behavior of the entire system. The term expert system refers to the origin of these rules. The expert is the purveyor of knowledge.

The structure of a rule-based expert system is fundamentally different to the algorithmic structure used in the development of procedural programs. The most outstanding difference is the way in which the expert system separates knowledge and facts. Knowledge (i.e. the rules) is stored in a knowledge base whereas the facts (i.e. the variable data that describes the state of the system and concerning which the rules reason) are stored in a separate database.

The expert system knowledge base stores the rules that describe the system in addition to information relating to these rules. The fact-base provides a data image of the system. The reasoning core of the expert system is its inference engine. It uses the fact-base to assess the status of the system and consequently invokes rules in the knowledge base. The explanation facility provides a human machine interface that helps developers and users alike in understanding exactly what the system is doing and more importantly, why?

2.6.2.3. Neighborhood Search Methods

Local search techniques attempt to locate sub optimal yet acceptable solutions in an exceptionally large solution spaces such as that generated by scheduling problems. A sub optimal solution can be reached by applying a series of small changes to a schedule that results in iterative improvement of the objective function. In order to develop a successful search strategy the design of the neighborhood structure is very important. The structure of the neighborhood describes the relationship between neighbors. It depends directly on the method used to generate the schedule. Each neighborhood comprises similar schedules that are constructed by two principle operations, adjacent pair-wise interchange of operations on the critical path of the schedule and one-step look-back interchange of operations. The search process begins by selecting a start point. Heuristics or simple rules are often used to make this selection and making small changes to the starting schedule results in iterative improvements to the objective function. In its simplest form only schedules that improve the objective function will be accepted. This has the effect of guiding the search to a local optimum point. The quality of the solution is heavily dependent on the starting schedule because schedules that do not improve the objective function

will always be rejected. A number of independent searches of the same solution space, with different start points increase the probability of finding a solution that is close to the optimum. The main problem with simple search techniques is their tendency to become trapped in unsatisfactory local minima. More advanced iterative improvement strategies do not automatically disregard schedules that do not improve the objective function and this allows them to escape from unsatisfactory local minima.

2.6.2.3.1. Random Search

Random search is an extremely basic method. It only explores the search space by randomly selecting solutions and evaluates their fitness. This is quite an unintelligent strategy, and is rarely used by itself. It does not take much effort to implement it, and an important number of evaluations can be done fairly quickly. For new unresolved problems, it can be useful to compare the results of a more advanced algorithm to those obtained just with a random search for the same number of evaluations. Random search has a few interesting qualities. However good the obtained solution may be, if it is not the optimal one, it can be always improved by continuing the run of the random search algorithm for long enough. A random search never reaches any stage such as a local optimum. Furthermore, theoretically, if the search space is finite, random search is guaranteed to reach the optimal solution. Unfortunately, this result is completely useless. For most of problems, exploring the whole search space takes far too long time.

2.6.2.3.2. Hill Climbing

Hill climbing is a standard local search procedure. It selects the first or the best improvement in the neighborhood to replace the seed. If the neighborhood of a solution is small, then it is reasonable and feasible to generate them all and select the best. If the neighborhood is very large, then to even generate them all might take a very long time. In this case it may be best to select the first improvement. One possible variation is selecting the best improvement from a fixed number of sampled schedules in the neighborhood. It is worthwhile noticing that there is no way to

escape the first encountered local optimum since any move in the local neighborhood produces a worse schedule.

On functions with many peaks (*multimodal* functions), the algorithm is likely to stop on the first peak it finds even if it is not the highest one. Once a peak is reached, hill climbing cannot progress anymore, and that is the problem when this point is a local optimum. A simple idea to avoid from the first local optimal consists in repeating several hill climbs each time starting from different randomly chosen points. This method is sometimes known as iterated hill climbing. By discovering different local optimal points, it gives more chance to reach the global optimum. It works well if there are not too many local optimums in the search space.

2.6.2.3.3. *Simulated Annealing*

Threshold algorithms accept inferior schedules as possible candidate solutions provided the deterioration in the objective function is less than some threshold value. Simulated annealing is a sophisticated variation on the simple concept of threshold algorithms. In the case of simulated annealing, the thresholds are positive and stochastic. In other words the threshold that determines whether a candidate schedule can be disregarded or not, varies over the period of the search.

The algorithm searches a neighborhood that is generated by some mechanism randomly changing the schedule to another in the same locality. The objective function is tested and in the case of an observed improvement the new schedule is taken as being the best. Otherwise the schedule is tested according to Metropolis' criterion before the decision to disregard the schedule as the source of an overall improvement, is made.

Metropolis' criterion states that if:

$$e^{(-\Delta E / T)} \geq \delta$$

where :

δ : A random number in the range 0 to 1 generated from a uniform distribution.

ΔE : The difference in the objective function between the current best schedule and the new candidate schedule.

T : The temperature describe by the cooling schedule.

If Metropolis' criterion is satisfied then a schedule generating an inferior measure of performance to that of the best known schedule will not be automatically disregarded and can therefore be used to generate offspring that may ultimately lead to superior schedules.

General steps in a basic simulated annealing method can be described as follows (Brandimarte, Villa, 1995, p341).

1. Choose an initial solution, initial optimal value and initial temperature.
2. Choose a new optimal solution randomly from the neighborhood and compute the fitness value.
3. Calculate the acceptance probability (P).
4. Accept the new solution with probability of P. If this solution is accepted, replace with the old one.
5. If the termination condition is met, stop the algorithm, otherwise update the parameters and go to step2.

2.6.2.3.4. Tabu Search

Tabu Search is an iterative technique that is characterized by its use of flexible or adaptive memory. As has already been discussed, the principal limitation of neighborhood search procedures is that they tend to become trapped in local minima. In order to avoid this, the Tabu search technique relies on specialized memory structures to avoid entrapment in local minima and achieve an effective balance of intensification and diversification. An evaluation function chooses the best solution at each iteration. The mechanism used by Tabu search to escape local minima can lead to other problems. The very fact that this technique allows the search to return to

less desirable solutions instead of making continual and iterative improvements can force the search to revisit previously evaluated schedules and even trap the search in an infinite loop. The search uses a ‘Tabu list’ to prevent these problems occurring. The Tabu list stores a group of moves or alterations to the schedule that are forbidden or Tabu at that particular time. Optimality is not guaranteed by approximation techniques. In terms of Tabu search, the closeness of the final solution to the optimum can vary depending on factors such as the list length as well as other variables.

2.6.2.3.5. Genetic Algorithms

A genetic algorithm is a directed random search technique, whose algorithms are analogous to the natural process of evolution. The evolutionary process analyses the performance or “fitness” of a species in their particular environment. From one generation to the next it tries to preserve characteristics and attributes that perform well and eliminate those that perform poorly in an attempt to continually improve the species and increase its chances of survival. In a similar manner, genetic algorithms attempt to arrive at the best solution to a problem by evolving from a reasonable through a process of continuous improvement at each iteration. In the context of the scheduling problem the technique identifies characteristics of schedules that have produced good solutions, and attempts to retain these characteristics and eliminate poor characteristics.

Genetic algorithms like all search techniques require a start point. This starting schedule or group of schedules can be generated in a number of ways. One approach is to randomly generate a series of starting points. The resulting search is highly diverse and therefore is less likely to become trapped in substandard minima. The inherent problem of diversity is a lack of intensification about quality solutions and as a direct result computational time can be excessive. Randomly generated initial populations are mainly used either to develop generic genetic algorithms or in developing applications for problems about which little or no a priori knowledge is available. When this domain knowledge can be tapped, a heuristic or empirical technique can be used to generate a more accurate starting population.

There are many strategies for deciding which part of solutions to retain and which to discard. In the domain of genetic algorithms these strategies are defined by a set of functions called genetic operators. The three most common types of genetic operators used are, selection, crossover, and mutation.

The aim of selection operators is to discriminate between good and bad schedules and therefore produce more copies of individuals whose fitness level is high. The selection procedure has a significant influence on driving the search towards good solutions in the shortest possible time.

The crossover operator takes two parents from the selection and uses these to generate children (i.e. new schedules in the form of binary strings). Analogous to reproduction processes in nature, children typically inherit characteristics of both parents. The crossover operator randomly selects a cut point. It then preserves and swaps alternate portions between parent strings.

The mutation operator seeks to diversify the search forcing it to jump into new neighborhoods and hence avoid convergence on local minima. It achieves this by randomly altering some of the '1's and '0's in the parent string. The rate of mutation must be set at a level that will guide the search to the global minimum point without having to search too many possible solutions and neighborhoods, and hence incur excessive computational time.

2.6.3. Simulation Modeling

Simulation has been successfully employed as an analysis tool for predicting the effect changes have on existing and hypothetical systems. This insight allows for more informed appraisals of alternatives, greatly enhancing the planning function.

Simulation modeling allows microscopic analysis of complex system dynamics giving the intimate understanding required to maximize the efficiency of such systems. As well as being used to predict the future and explain the operation of complex processes, simulation models are also used in real-time control systems to provide decision support for automated (intelligent) decision makers.

Simulation is certainly more tractable than mathematical programming formulations of FMS scheduling problems. With simulation, there is no concern about feasibility, since there is no need to make any simplifying assumptions. The simulation model can be built as close to reality as one needs to however, simulation is carried out with just one rule for each type of decision, then simulation does not serve any decision support purposes then, the only purpose of simulation would be prediction-when a job can be expected to be completed, what machine utilizations can be expected, etc. Simulation can work as a decision support tool when there is the possibility to simulate under different decision alternatives. The informed decisions could be made by looking at the simulation results. Current research suggests that a combination of simulation and ‘meta-heuristic’ optimization techniques- applied to analytically intractable problems- can yield optimal or near optimal solutions.

2.6.3.1. Use of Simulation in Scheduling

Discrete event simulation has been successfully employed as an analysis tool for predicting the effect of changes on existing and hypothetical systems. This insight allows for more informed appraisals of alternatives, greatly enhancing the planning function.

Simulation modeling allows microscopic analysis of complex system dynamics giving the intimate understanding required to maximize the efficiency of such systems. As well as being used to predict the future and explain the operation of complex processes, simulation models are also used in real-time control systems to provide decision support for automated (intelligent) decision makers.

Current research suggests that a combination of simulation and ‘meta-heuristic’ optimization techniques applied to analytically intractable problems can yield optimal or near optimal solutions.

The versatility of simulation modeling allows it to explore new challenges easing comfortably into a variety of roles. Its marriage to optimization and approximation search techniques has lead to the development of sophisticated yet practical and

understandable systems that can take-on mammoth scheduling tasks. Even though many of these techniques are still in their infancy, promising results could have far reaching consequences for optimization problems in other domains.

To date, much discussion has centered on using simulation to assist in supporting long-term strategic decisions. The success of simulation modeling in this role has however been mixed.

The simulation model is an important part of the overall control system.

“The major function of the simulation model is to evaluate control policies in a flexible manufacturing cell by examining the effect of the scheduling rules on an on-line test base.”

“Thus at the end of all simulation passes, the best scheduling rule that results from the simulation is applied to the physical manufacturing system. The basic principle behind the simulator is the use of a deterministic simulation as a short-term predictive tool for alternative control strategies in a manufacturing.” Wu (1989).

Rembold et al (1993) echo this concept in their description of the Amherst-Karlsruhe dynamic scheduling system.

“The heart of the system is a knowledge-based selector for scheduling methods and a library of logic scheduling algorithms. The system knows from a given order and manufacturing status which logic scheduling algorithms have to be used to obtain the desired manufacturing goal and to meet due dates.”

Taha (1997) goes on to explain that one of the main advantages simulation modeling has over rigid mathematical systems is the flexibility that results from its simplicity, stating:

“Simulation models are much more flexible in representing systems than their mathematical counterparts. The main reason for this flexibility is

that simulation views the system at elemental level, whereas mathematical models tend to represent the system from a more global standpoint.”

In this thesis, simulation is used as a tool to evaluate the effects of different scheduling dispatching rules over different performance measures. Also the proposed models have the capability to give information about where and in which sequence each part can be processed thorough the process. Additionally it gives the necessary information about the parts that missed the required due dates. By the help of this information, the production planner has supportive information to make the relevant adjustments over the production system. Detailed knowledge about the simulation study is given at Chapter 4.

Simulation is begun to be used in many different service and production areas for different purposes and proposes many advantages compared to other techniques. The increasing role of simulation in manufacturing, its application areas and the possible advantages of using simulation are introduced under the following sections.

2.6.3.2. New Emerging Role of Simulation in Manufacturing

Simulation, in addition to giving the user insight into how complex systems function and how variables interact with each other, provides the user an informative approximation of “what-if” scenarios. To date much discussion has centered on using simulation to assist in supporting long-term strategic decisions. The success of simulation modeling in this role has however been mixed. In the past decade simulation modeling has taken on a new role providing analytical support to real-time decision makers. In the context of completely automated and flexible manufacturing systems, these decision makers are often AI components. Thus simulation models have become completely integrated as modules of automated control systems. McNally and Heavey (2002) describe this emerging niche.

“Other researchers have been proposing the extension of simulation tools beyond a traditional design role (Dewhurst et al 2001, Kosturiak and Gregor 1999). With this approach the same model can be extended with

control functions and interfaces with the environment (shop floor data collection systems and production planning and control databases) to support dynamic scheduling of production orders, capacity plans, labor allocations etc.”

“One area to emerge over the last decade has been the real time control and planning of manufacturing systems using computer simulation; especially in the area of flexible manufacturing systems. The simulation model is linked to the controllers of the flexible manufacturing system. Real time activities primarily refer to daily operations that require efficient, timely, and adaptive responses to short-term planning, scheduling and execution problems.”

2.6.3.3. Applications of Simulation Modeling

The applications of simulation modeling are wide and varied. McLean et al (2001), give a summary of general problem classes that are not industry specific. Simulation can be used in manufacturing to:

- Model “as-is” and “to-be” manufacturing and support operations from the supply chain level down to the shop floor.
- Evaluate the manufacturability of new product designs.
- Support the development and validation of process data for new products.
- Assist in the engineering of new production systems and processes.
- Evaluate their impact on overall business performance.
- Evaluate resource allocation and scheduling alternatives.
- Analyze layouts and flow of materials within production areas, lines and workstations.
- Perform capacity planning analyses.

- Determine production and material handling resource requirements.
- Train production and support staff on systems and processes.
- Develop metrics to allow the comparison of predicted performance against “best in class” benchmarks to support continuous improvement of manufacturing operations.

2.6.3.4. Advantages of Simulation Modeling

Pegden et al (1990) present the following benefits of using simulation modeling to deal with complex problems:

- New policies, operating procedures, decision rules, information flows, organizational procedures, and so on can be explored without disrupting ongoing operations of the real system.
- New hardware designs, physical layouts, transportation systems, and so on, can be tested without committing resources for their acquisition.
- Hypotheses about how or why certain phenomena occur can be tested for feasibility.
- Time can be compressed or expanded allowing for the speed up or slow down of the phenomena under investigation. Bottleneck analysis can be performed indicating where works in process, information, materials, and so on are being excessively delayed.
- A simulation study can help in understanding how the system operates rather than how individuals think the system operates.
- “What if” questions can be answered. This is particularly useful in the design of new systems.
- Insight can be obtained about the interaction of variables.

- Insight can be obtained about the importance of variables on the performance of the system.

There is however a number of drawbacks associated with complex simulation models. The modeling process can be extremely costly in both time and other resources. Also, a complete knowledge and understanding of the system dynamics is required to develop a successful model. This level of knowledge may not exist or may be dispersed and difficult to compile. Other problems can plague even the most meticulously developed models.

“Simulation computations, though usually simple, are voluminous. It is thus unthinkable to execute a simulation model without the use of a computer.”

“Moreover, the execution of some simulation models, even on the fastest computers, may be slow.” Taha (1997)

CHAPTER THREE

LITERATURE REVIEW

3.1. Introduction

Setup includes work to prepare the machine, process, or bench for product parts or the cycle. This includes obtaining tools, positioning work in process material; return tooling, cleanup, setting the required jigs and fixtures, adjusting tools and inspecting materials. The sequence dependent setup can be either separable or inseparable from the process of a job. If the setup procedure is inseparable, then the setup cannot begin unless the job is physically ready to be processed. On the other hand, if the setup procedure is separable then the setup can begin even if the job is not ready for a machine as long as the machine is free.

There are two types of setup time variations: Between-job variation and within-job variation (Bobrowski & Kim, 1997). Between-job variation occurs due to different setup requirements in different jobs. These requirements create the sequence dependency of setup times. Most previous studies on set up time have focused on sequence dependency and ignored within-job variation. Within-job variation occurs due to the variation in skills of setup crews and other “noise” in the system. Within-job variation can be reduced by improving the skills of setup crews through training or by standardizing setup change procedures.

The setup operation has for long been considered negligible and hence ignored or considered as part of the processing time for the case of setup time. While this assumption simplifies the analysis and/or reflects certain applications, it adversely affects the solution quality for many applications which require explicit treatment of setup. Such applications, coupled with the emergence of production concepts like time based competition and group technology, have motivated increasing interest to include setup considerations in scheduling problems.

The importance of setup times have been investigated in several studies. Wilbrecht and Prescott (1969) found that sequence dependent setup times are

significant when a job is operated at or near full capacity. Flynn (1987) determined that applications of both sequence dependent setup procedures and group technology principles increase output capacity in a cellular manufacturing shop. She applied the Repetitive Lots (RL) procedure which capitalizes on the sequence dependency of setup times by scanning a queue of waiting jobs and seeking to find a job identical to the job that was just processed on the machine. Wortman (1992) underlined the importance of considering sequence dependent setup times for the effective management of manufacturing capacity. Krajewski et al (1987) concluded that, regardless of the production system in use, simultaneous reduction of setup times and lot sizes is the most effective way to reduce inventory levels and improve customer service.

Furthermore, treating setup time separately from processing time allows operations to be performed simultaneously and hence improves performance. This concept is inherent in recent production management philosophies and techniques such as just-in time (JIT), optimized production technology (OPT), group technology (GT), cellular manufacturing (CM), and time based competition.

Sequence dependent setups are specially considered at some manufacturing industries. In the production of integrated circuits such as microprocessors, memory chips, microcontrollers and other semiconductor devices in microelectronic factories like wafer fabrications sequence dependent setup time has critical importance for reducing the time in manufacturing. One type of wafer processing equipment that is characterized by sequence dependent setups is an ion implanter which focuses on an accelerating beam of a desired dopant's electrons into specific locations on the wafer's surface within a gaseous environment. The type of gas contained in an implanter's processing chamber depends upon which dopant is to be implanted into the wafer. Once an implanter has finished processing of a current wafer, the gas currently contained in the process may have to be purged and replaced with a different gas depending in the process chamber.

In paint fabrics, the paint operations often require changeover. Every time a new color is used, the painting devices must be cleaned. The clean up time often depends

on the color just used as well as the color about to be used. In practice, the best sequence is to go from light to dark colors because the cleanup process is easier.

In paper bag factories setup is needed when machine switches between types of paper bags and the setup duration depends on the degree of similarity between consecutive batches such as size and number of colors.

In several textile industry applications setup for weaving and dying operations depends on the job sequence. In the container and bottle industry, the settings change depending on the sizes and shapes of the containers. In the plastic industry, different types and colors of products require sequence dependent setups. Similar practical situations arise in the chemical, pharmaceutical, food processing, metal processing and paper industries. Also, in an automatic turning center, the setup time depends on the difference in the number and type of the tools currently mounted on the turret and the ones required for the next work piece.

Sequence dependent setups have significant implications on shop floor performance (Kim & Bobrowski ,1994)

- Setup time constitutes a part of flow time that directly affects the throughput rate of a production system
- The cost of unit setup time is usually higher than that of unit processing time since the cost of setup time includes loss of productivity due to machine down time and the cost of setup itself, i.e. labor cost of a setup technician.
- The setup change can often be made only by a qualified technician since setup is often a complicated work requiring a higher level of expertise. These qualified personnel are limited resources that are not always available.

In the case of sequence dependent setup times, the total amount of setup time depends on the level of WIP in the shop and this relationship must be known to the order function to determine the appropriate level of WIP in the shop. Generally an increase in the level of WIP in the shop (longer queues) leads to lower setup times for a given output and to lower utilization of the facilities for a given output cause of setup time savings so the maximum output will be obtained. Short average flow

times which can be obtained by a low level of WIP can increase the setup times and costs.

Scheduling research involving sequence dependent setup dates back over four decades. Most researches have investigated single and parallel machine problems. Flow time related performance criteria have been popular. Some researches developed optimization procedures with respect to different cost functions but very few considered due date related objectives.

The following four sections discuss the studies considering scheduling with sequence dependent setup times at different machine configurations.

3.1.1. Single Machine Problems

Arriving tasks require service from a single available resource. Tasks are performed by the resource one at a time and each task has different properties such as processing time, due date, setup time. Different researches consider different performance criteria and use different approaches for solution. This section addresses the setup literature in a single machine environment.

For a cost function which includes total setup cost Burstall (1966) studies over a problem with a machine which may be in different states and each job can be processed in different states of machine with different cost. He proposes to break down the problem into smaller sub problems and uses a branch and bound technique to reach at the objective of minimizing the total cost. Barnes and Vanston (1981) study over the minimization of the sum of setup cost and linear delay penalties. They develop several branch and bound procedures based on different branching rules and compare those with a hybrid of branch and bound and dynamic programming. They report that the hybrid approach performs better than any of the pure branch and bound procedures. Bianco et al. (1988) use a branch and bound procedure for the maximum flow time problem with release times. Two different lower bounds are used, one based on the linear programming relaxation of the integer programming formulation and the other based on Lagrangian relaxation. An upper bound with a constructive heuristic and specify a set of dominance criteria are formulated. Uzsoy

et al. (1991) develop procedure which decomposes a complex job shop with sequence dependent setup times and re-entrant work flows into separate work centers. The job shop is presented as a disjunctive graph, and the release times and due dates of the jobs at each work center are estimated. Then a separate scheduling problem is solved of each work center. One of the arising problems is the single machine scheduling problem under precedence constraints. They propose an optimal branch and bound algorithm and a local improvement heuristic for the lateness objective.

Miller et al. (1999) propose a hybrid genetic algorithm for the single machine, single stage scheduling problem with a fixed planning horizon to minimize the sum of setup, inventory holding and backlog costs.

Uzsoy et al. (1992) propose heuristics for the dynamic problem and dynamic programming procedure for the static problem to minimize the maximum lateness and the number of tardy jobs. They also develop worst case error bounds for the special case in which setup times are bounded from above by the processing times. Lee et al. (1997) propose a three phase heuristic which computes some parameters in the first phase. In the second phase, an initial schedule is developed using priority rule based on these parameters. This schedule is improved in the third phase via a local improvement search procedure. They aimed the minimization of weighted total tardiness. Farn and Muhlemann (1979) review an exact algorithm and several heuristics to solve the setup time minimization problem. They analyze both the static and dynamic problems, and conclude that the best procedure for the dynamic problem isn't necessarily the same as that for the static problem.

Kolahan et al. (1995) use a tabu search approach to find a good sequence such that, once the tool replacement decision has been made in an automated machine center, the total expected production cost which includes sequence-dependent setup cost is minimized.

Monma and Potts (1989) consider batch scheduling problems in which the number of batches is fixed and the triangular inequality of setups is satisfied. They show that under these conditions the problems of maximum lateness, total weighted

completion time and the number of tardy jobs can be efficiently solved via dynamic programming.

Raman et al. (1989) investigate the dynamic scheduling of an automated manufacturing center in which jobs are processed in batches and constant setup times between batches are incurred. For the mean flow time and mean tardiness criteria, they developed the characteristics of the optimal solution. They also provided an implicit enumeration approach to solve the problem with the mean tardiness objective.

Table 3.1 summarizes the literature of single machine problems which considered the sequence dependent setups in scheduling process of jobs.

3.1.2. Parallel Machine Problems

Arriving jobs can be processed on any one of a number of available machines. Each job with different characteristics has a single operation that can be performed on any machine. Setup times are separate from processing times, and the magnitude of a setup depends upon the type of previous setup and the next job to be processed. In the following pages the researches are introduced in sequence of the techniques commonly used.

Deane and White (1975) address a static problem with independent batch type jobs which have sequence dependent setups on multiple parallel processors. They present a branch and bound algorithm which minimizes the total setup cost while meeting work load balancing restrictions. They explain that their algorithm is feasible for solving small to moderate size problems and discuss the extension to handle non-identical processors. Bitran and Gilbert (1990) study a parallel machine problem with two magnitudes of sequence dependent setups. They construct a network model to represent the cost structure that is used to define product families. They develop a branch and bound procedure that allocates the job families so as to minimize the total setup cost. Marsh and Montgomery (1973) develop a branch and bound algorithm for the identical parallel machine problem where the objective is to minimize the total setup time.

Table 3.1 Literature of scheduling researches considering single machine environments

Authors	Technique	Objective	Constraints
Burstall (1966)	Branch and Bound	Total setup cost	
Barnes & Vanston (1981)	Branch and Bound	Total setup cost, linear delay penalties	
Bianco et al.(1988)	Branch and Bound	Maximum flow time	Non zero release times
Uzsoy et al. (1991)	Branch and Bound	Maximum lateness	Precedence constraints, re-entrant work flows
Farn& Muhlemann (1979)	Heuristics	Comparison the performance of heuristics for both dynamic and static scheduling problems	
Monma & Potts (1989)	Dynamic Programming	Maximum lateness, total weighted completion time, number of tardy jobs	Fixed number of batches, triangular inequality of setups
Raman et al.(1989)	Implicit Enumeration Approach	Mean flow time, mean tardiness	
Uzsoy et al. (1992)	Dynamic Programming, Heuristics	Maximum lateness, number of tardy jobs	Precedence constraints, non zero release times
Kolahan et al. (1995)	Tabu Search	Cost function	
Lee et al. (1997)	Heuristic using local improvement search procedure	Weighted total tardiness	
Miller et al. (1999)	Genetic Algorithm	Number of setups, inventory holding and backlog costs	

They assume that all jobs are available at time zero and have no due date restrictions. They discuss an extension of their procedure to the non identical machine case. Kang et al. (1999) propose a sequence splitting technique to break down the lot sizing and scheduling problem into tractable sub problems. They use a heuristic approach based on branch and bound to minimize the total setup cost.

Guinet (1993) studies parallel machine problems in the make-to-stock and make-to-order environments. He uses the total flow time and mean tardiness as performance criteria. Solution methodologies for the make-to-stock case are the Hungarian method and an assignment algorithm. He applies a transportation algorithm and a sequential assignment procedure for the make-to-order case.

Ovacik and Uzsoy (1993) consider a special case of the parallel machine problem where the setup time associated with each job is bounded from above by its respective processing time. They construct worst-case error bounds for the maximum lateness and makespan problems.

Ovacik and Uzsoy (1995) present a family of rolling horizon procedures to minimize the maximum lateness with parallel identical machines, sequence dependent setups and dynamic job arrivals. They first develop a technique for the single machine problem and then extend it to solve parallel machine problem. They report that the performance of their technique can be improved if better lower bounds are found.

Lee and Pinedo (1997) develop a three phase heuristic to minimize the weighted tardiness on identical parallel machines. The first phase consists of computing some factors for the particular instance of the problem. In the second phase, a sequence is constructed via a dispatching rule which makes use of the factors computed in the previous phase. Phase three starts with this schedule and attempts to improve it using a simulated annealing technique.

Franca et al. (1996) use a tabu search methodology to minimize the makespan of a set of jobs with sequence dependent setups on identical machines. They obtain an initial solution by assigning each job to the machine which results in the smallest

increase in the current make span. This solution is improved via a tabu search procedure where moving a job from the busiest machine to another machine constitutes a neighborhood move. The solution found by the tabu search procedure is further improved by post processing the sequence on the busiest machine.

Dearing and Henderson (1984) analyze a scenario with identical parallel processors, and constraints on demand, total setup time and other operational characteristics. They model the problem as a mixed integer linear program to impose preferences among products and to smooth the number of setups from week to week.

Ghosh (1994) shows that the problem with the objective of minimizing the weighted completion time in the presence of sequence dependent setups is strongly NP-hard for arbitrary numbers of batches and machines which are identically parallel. When the number of batches is arbitrary, the problem is strongly NP-hard even if the jobs are identical. However, when the number of batches is fixed, the total completion time problem is solvable in polynomial time.

Table 3.2 summarizes the literature of scheduling problems in parallel machine environment considering the sequence dependent setups.

3.1.3. Flowshop Problems

In a m-machine flow shop, arriving jobs have to be processed on m machines in the same technological order. Each job has m operations and operation i of each job is to be performed on machine i. Operation times for each job on different machines may be different.

Corwin and Esogbue (1974) show the optimality of permutation schedules for the two machines flow shop problem with make span minimization objective where the setup times are sequence independent in the first machine and sequence dependent in the second. They present dynamic programming formulations for the problem.

Uskup and Smith (1975) consider a two stage production sequencing problem with due date constraints and sequence dependent setup times on both machines that are proportional to the processing times.

Table 3.2 Literature of scheduling researches considering parallel machine environments

Authors	Technique	Objective	Constraints
Deane and White (1975)	Branch and Bound	Total setup cost	Workload balancing restrictions
Bitran and Gilbert (1990)	Branch and Bound	Total setup cost	Minor and major setups
Marsh and Montgomery (1973)	Branch and Bound	Total setup time	Identical and non-identical machines
Kang et al. (1999)	Branch and Bound	Total setup time	
Dearing and Henderson (1984)	Mixed integer linear program	Total setup cost	
Guinet (1993)	Hungarian Method	Total flow time, mean tardiness	
Ovacik and Uzsoy (1993)	Algorithm using worst case error bounds	Maximum lateness, makespan	Bounded setup time
Ghosh (1994)	Dynamic programming	Weighted completion time	Arbitrary number of machines and batches
Ovacik and Uzsoy (1995)	Rolling horizon procedure	Maximum lateness	Dynamic release times
Franca et al. (1996)	Tabu search	Makespan	
Lee and Pinedo (1997)	Heuristic combining a dispatching rule and simulated annealing	Weighted tardiness	

Their problem is different from the classic flow shop problems in that they allow different processing orders in two stations. They propose a branch and bound procedure to minimize the total setup cost without violating the due date constraints.

Gupta and Darrow (1986) show that the static two-machine flow shop scheduling problem with the objectives of make span minimization is NP-complete in the presence of sequence dependent setups. They also show the permutation schedules for this problem may lead to suboptimal solutions. They present four approximate algorithms. They report that two of these algorithms become ineffective when the setup times are in the range as the processing times.

Gupta (1986) presents a proof of NP-completeness for the static flow shop scheduling problem in which jobs need to be processed in the same technological order in sequential machines so as to minimize the make span. He suggests that a TSP formulation of the scenario based on the TSP formulation.

Srikanth and Ghosh (1986) formulate the flow shop problem with sequence dependent setups as a mixed integer linear problem. The objective is make span minimization or the minimization of the mean flow time. They use a reduced number of integer binary variables compared with the previous formulations. They solve several instances of the problem with combinations of up to six jobs and six machines and find optimal solutions in all cases.

Bianco et al. (1988) show that the flow shop scheduling problem with sequence dependent setup times is equivalent to the TSP with additional visiting time constraints. They give a mathematical formulation and two lower bounds for the TSP, and propose two heuristics algorithms to solve the scheduling problem.

Szwarc and Gupta (1987) consider a flow shop scheduling problem with additive sequence dependent setups. They develop a polynomially bounded approximate method which finds permutation schedules to minimize the make span. They show that their algorithm yields optimal results for the two machine case.

Parthasarathy and Rajendran (1997) address the flow shop scheduling problem with the objective of minimizing the mean weighted tardiness. They propose a simulated annealing procedure with a neighborhood structure based on a random insertion perturbation scheme. Later, they extend their algorithm to cover the

objectives of minimizing the maximum weighted tardiness and minimizing the total tardiness of the jobs.

Rios-Mercado and Bard (1998) present two insertion based heuristics. One is the extension of a procedure for the general flowshop problem without sequence dependent setups and the other is a greedy randomized adaptive search procedure. Their methods perform well for the problems in which the setup times are smaller than the processing times. They also propose a two phase lower bound for this problem. The first phase involves solving the assignment relaxation of the associated TSP problem. The lower bound obtained in the first phase is improved in the second phase via idle time insertion.

Table 3.3 summarizes the literature of scheduling problems in parallel machine environment considering the sequence dependent setups.

3.1.4. Job shop Problems

A few researches have been done over job shop scheduling problems with sequence dependent setups. Almost all approaches to the problem involved the use of heuristic procedures due to complexity of the problem. Due date and processing time related performance criterias have been popular. Some researchers chose to optimize with respect to total cost of setup. The common investigated topic is the evaluation of the effect of sequence dependent setups over different performance measures and also to compare the reaction of different methods to setups. The optimization objective hasn't been almost ever considered due to the high complexity of problem sizes.

Setup considerations in job shops started almost three decades ago when Prescott and Wilbrecht (1969) discovered using a simulation study that sequence dependent setup times play a critical role in the performance of a job shop operating at full or near full capacity. They used a setup oriented rule which gives the highest priority to a job with the shortest setup time. In their study, it was desired to get a lower in process inventories and shorten scheduled production time.

Table 3.3 Literature of scheduling researches considering flowshop machine environments

Authors	Technique	Objective	Constraints
Corwin and Esogbue (1974)	Dynamic Programming	Makespan	Sequence dependent on one machine of two machines flowshop
Uskup and Smith (1975)	Branch and Bound	Total setup cost	Due dates
Gupta and Darrow (1986)	Approximate algorithms	Makespan	
Srikar and Ghosh (1986)	Mixed integer linear programming	Makespan, mean flow time	
Gupta (1986)	Symmetrical traveling salesman problem	Makespan	Limited storage
Szwarc and Gupta (1987)	Polynomially bounded approximate method	Makespan	Additive setup
Bianco et al. (1988)	Heuristic algorithms with 2 lower bounds for traveling salesman problem	Evaluation the similarity of Flexible jobshop problem with sequence dependent setups to traveling salesman problem	Additional visiting time constraints
Parthasarthy and Rajendran (1997)	Simulated annealing	Mean weighted tardiness, total tardiness of jobs	
Rios-Mercado & Bard(1998)	Heuristics	Makespan	

As a result of the study it was concluded that, SIMSET produced statistically largest number of jobs going out of the shop per week and it gave the best results whether a weight was given to all criterias or not. This indicated that setup times must play a critical role in the performance of the machine shop. But they had no explanations why SIMSET was superior. They could only explain that as the shop was working at full capacity, it seemed logical that setup times would be an important criterion. O'Grady and Harrison (1988) investigated the effectiveness of list priority rules. They proposed a search sequencing rule which sets the priority of a job in each machine's queue as a linear combination of the operation times, due dates and the setup times. They evaluated the performance of their rule against that of the existing ones using simulation model of a job shop. Gupta et al. (1993) discussed an integrated simulation based approach to operations planning and scheduling in a flexible manufacturing system (FMS). Several hypotheses were formulated with respect to the performance of a number of performance measures such as system utilization and maximum tardiness $\sum C_i / n_i, \sum T_i / n_i$. Kim and Bobrowski (1994) examined the implications of setup times by testing different dispatching rules in environments defined by due date tightness, setup times and cost structure. They classified and tested these scheduling rules based on whether setup time and/or due date information were used. They found that an increase in setup time had a minor effect on the setup oriented rules compared to the effect on only due date oriented rules. They also stated that setup times must be given explicit consideration in solving the scheduling problems especially when they are sequence dependent. Also, they searched for the implications of sequence dependent setups over the capacity management. Ovacik and Uzsoy (1994) considered the maximum lateness criterion in a complex job shop with sequence dependent setups. They use deterministic simulation to predict the arrivals of jobs at each work center and solved the resulting the sub problems separately. They developed scheduling techniques with myopic dispatching rules in a job shop environment, semiconductor testing facility and reentrant flow shop; they found that performance improvements were more significant for shop configurations with constraint resources and reentrant product flow. Low (1995) proposed a heuristic algorithm that could also handle some complex constraints such as multiple machines, multiple resources and alternate

routes. He evaluated the performance of his algorithms for various criteria using simulation. The statistical results showed that his algorithm performs better than algorithms that consider either no setup or sequence independent setups. Kim and Bobrowski (1997) investigated the impact of setup time variations on sequencing decisions. In this study setup times were stochastic including the job variations and simulation was used. The study investigated that; how much variation in setup time could be tolerated while retaining the benefit of the setup conscious sequencing rules when compared to conventional sequencing rules and how important it was to accurately estimate setup time. During the simulation step, experimental design was used to make a comparison between setup time's variations, due date tightness and sequencing rules. Results showed that setup time variation had a negative impact on shop performance but it didn't diminish the advantages of setup conscious sequencing rules over conventional sequencing rules in dealing with setup times. The most significant impact of the setup time variation was observed in the due dates performances such as proportion tardy and tardiness. Also, it was observed that, the difference between setup oriented rule and conventional rules were increased as the setup time variation increased and as due dates became tighter.

Hershauer (1970) studied the operations of a commercial printing shop in which certain machines had sequence-dependent setup times and costs. He developed a linear additive priority function that considered the sequence dependent setup times. The function gives the high priority to the jobs that would reduce setup time by adding a certain value to the total priority value. Thus, the setup time was indirectly considered in this study.

Generally setup times are frequently not available to the production scheduler and when they are, they frequently are only averages which conceal the sequence dependency of the setup times. That's why White and Wilson (1977) developed a procedure that classifies setup operations and predicts the setup times and then a method was developed for sequencing the jobs so as to minimize total setup time. A procedure was developed that uses self recorded total setup times, a setup operation classification scheme, and a statistical technique of multiple classification analysis to develop estimation tool for sequence dependent setup times. In other words

considering the setup characteristics used in prediction equation sequence dependent setup times were estimated.

Zhou and Egbelu (1989) designed heuristic algorithm to minimize the make span in a flexible manufacturing cell with multiple jobs, multiple machine stations and sequence dependent setup times. Their algorithm generates an initial solution and displays it on a computer graphic interface. Then, this solution is modified interactively by human expert. Choi and Korkmaz (1997) formulated the problem of job shop problem with sequence dependent setup times as a mixed integer program and a simple, polynomial time heuristic procedure was presented to solve it. The objective of problem was to minimize the time required to complete all jobs. The procedure was based upon sequentially identifying a pair of operations that provide a minimum lower bound on the makespan of the associated two job/m machine problem with release time. At each iteration of heuristic, an operation was chosen from 2 candidate operations: one with the earliest start time and the other with minimum local makespan. During the computation of earliest start time and minimum local makespan, sequence dependent setup times were considered.

There are no reported results in a job shop scheduling with batch processing times except those of Sotoskov et al. (1998) who developed constructive heuristics for minimizing several objective functions where batch setup times are sequence independent.

Xioquing and Noble (1999) investigated a job shop scenario with release times, due dates and sequence dependent setups. Their objective was to minimize the weighted sum of squared tardiness. They decomposed the problem into series of single machine problems and scheduled the machines one at a time in the order of their criticality. Their criticality of each machine was determined based on its marginal contribution to the overall objective function. The single machine problems were solved using Langrangian relaxation and shifting bottleneck approaches. The modified shifting bottleneck approach to job shop scheduling with sequence dependent setups was compared with the earliest due date (EDD), apparent tardiness cost (ATCS) and similar setup (SIMSET) dispatching rules. The MSBSS approach

resulted in superior solution quality compared to the results of other dispatching rules in reasonable computational time.

Artigues and Roubellat (2002) presented a new polynomial insertion algorithm of an operation inside a multi resource job shop schedule with sequence dependent setup times for the objective of maximum lateness minimization. The proposed algorithm could be used to deal with unforeseen jobs in online scheduling as well as a basis tool for off line schedule generation procedures. The setup times were decomposed into three elementary setup operations: Association operation, dissociation operation and family changeover operation. Association operation was needed before the processing of the part on a machine. After the completion of jobs on machine the resource had to be dissociated if the resource was required by a consecutive operation. A family changeover operation had to be performed when switching from family to another one. By this study it was expressed that the proposed algorithm might be useful at the further studies as a tool for neighborhood generation scheme in local search procedures as an alternative to classical insertion algorithms when sequence dependent setup times were considered.

Ballicu, Giua, and Scatzu, (2002) used a MILP for general job shop scheduling problems with sequence dependent setups and an arbitrary number of machines. Two cases were handled. First, it was assumed that the setup needed to be performed only on the machine might start as soon as the machine had completed the precedent operations. In the second case, it was assumed that the setup could only be performed if both machine and job were available. The results of MILP approach were compared with the results of LEKIN software. The objective function was the minimization of makespan. It was observed that MILP gives significantly better solutions than heuristic approaches in LEKIN.

Mason, Fowler and Carlyle (2002) presented a modified shifting bottleneck heuristic, which accommodates the various types of machine/tool group sub problems, for scheduling of jobs in “complex” job shop which was characterized by re-entrant product flow through a number of different tool groups. It was developed considering the performance measure of weighted tardiness which was desired to be

minimized. During the solution step for each sub problem of each tool group, some restrictions were considered for batching machines. Therefore, Lee and Pinedo's parallel machine ATCS index was adapted to accommodate batch processing tool groups. In experimental test stage different batching horizons were evaluated with the proposed SB heuristic to determine if any benefit could be realized from delaying the formation and subsequent processing of a batch in order to potentially form a fuller batch. The inclusion of the batching horizon did not lead to a noticeable increase in solution time. Four different dispatching rules were evaluated along with the proposed SB heuristic (FIFO, CR, EDD and weighted SPT). It was observed that when sequence dependent setups were included into the problem instances, the performance of proposed heuristic improved. When the number of job cases were increased the difference between performances of the best dispatching rule (WSPT) and the proposed heuristic increased which indicated that the proposed heuristic appeared to be a viable approach to scheduling complex job shops.

Lockett and Muhlemann (1971) studied over minimizing the total number of tool changes in a job shop with sequence dependent setups. Here the changeover of tools for the next job is dependent not only on the previous job, but also on jobs before it considering whether the stations were empty. First order serial changeovers were due to the tool requirements of the previous job, higher order serial changeovers were due to tool requirements of jobs before the previous jobs because the previous job didn't require tools in certain stations. If each job had required a tool in every station, then the problem would have been of the traveling salesman type. A branch and bound algorithm was used but it has been shown to be computationally restrictive. So different heuristic rules are evaluated. The best heuristic was found as the traveling salesman without backtracking which was derived from the branch and bound solution. Gupta (1982) studied a static job shop scheduling problem. In this problem, each job was allowed to have a different route and a different number of operations, and each machine could be visited by the same job more than once. The objective was to minimize the total setup cost. He developed a branch and bound procedure. At the branching step of the algorithm, the partial feasible schedule which had the minimum lower bound was based on the sum value of accumulated setup cost and potential minimum setup cost. If there were two or more partial schedules having the

same lower bound, then the one which had the more number of operations was selected. It was realized that the proposed algorithm could be used to solve small size problems. Further, it would provide an insight into the structure of optimal solutions which might be used to devise good heuristic rules as the heuristic rules were generally more appropriate to solve larger size scheduling problems. They showed that there was no single dominating rule that yielded satisfactory results with respect to all performance measures. However, the local dispatching rules outperformed the global dispatching rules, which consider information about other machines in the shop, indicating that benefits of using global rules might not be always applied.

Candido, Khator, and Barcia (1998) presented a robust procedure to solve job shop scheduling problems with large number of more realistic constraints such as jobs with several subassembly levels, alternative processing plans for parts and alternative resources for operations, requirement of multiple resources to process an operation, resource calendars, batch overlap and sequence dependent setups. Also, the approach algorithms were used to obtain a set of initial solution. Each initial solution was enhanced by a local improvement procedure which stopped the search when a local minimum was reached. Then a hybrid genetic algorithm which incorporated a local hill climbing procedure was applied to the set of local optimum schedules. When sequence dependent setup times were present, the overlap time of operation as the minimum operation completion time can be under or overestimated. It didn't guarantee that all optimal solutions will be generated. But it reduced the number of non-active schedules compared to a strategy which used shortest setup time in the setup table of successor of the operation.

In Chan and Dae-Sik (2002) presented a mixed integer program model of a job shop scheduling problem with the alternative operation sequences and sequence dependent setup constraints simultaneously considered for the purpose of makespan minimization along with a local search scheme that significantly enhanced many greedy based dispatching rules. In this study various dispatching rules with and without the local search scheme were tested to identify the best rule among them and it was concluded that the local search scheme indeed enhanced the performance of

the dispatching rules and the earliest start time first (EST) dispatching rule gave the best overall relative performance.

As it can be realized from the studies described above that, although there are some studies about the effect of setup and due date information on the performance of dispatching rules, there is no extensive study that compares the performance of different classes of scheduling rules, e.g. global rules vs. myopic dispatching rules, with respect to multiple performance measures in shops with sequence dependent setup times of various characteristics.

Table 3.4 summarizes the literature of scheduling problems in jobshop machine environment considering the sequence dependent setups.

Table 3.4 Literature of scheduling researches considering jobshop machine environments

Authors	Technique	Objective	Constraints
Prescott and Wilbrecht (1969)	Simulation	Production time, minimization of process inventories	Production at full capacity
O'Grady and Harrison (1988)	Simulation	Performance evaluation of the proposed rule considering miscellaneous objectives	
Gupta et al (1993)	Simulation	Maximum tardiness, system utilization	
Kim & Bobrowski (1994)	Simulation	Performance evaluation of different dispatching rules considering miscellaneous performance measures and capacity management	
Ovacik and Uzsoy (1994)	Simulation	Maximum lateness	Limited resources reentrant product flow
Low (1995)	Simulation	Miscellaneous objectives	

Table 3.4-Continue

Authors	Technique	Objective	Constraints
Kim and Bobrowski (1997)	Simulation	Evaluation the impact of setup time variations on sequencing decisions	Stochastic setup times
Hershauer (1970)	Algorithm based on linear additive priority function	Setup cost, setup times	
White and Wilson (1977)	A procedure for setup time prediction and an algorithm for scheduling	Total setup time minimization and setup time estimation	
Zhou and Egbelu (1989)	Heuristic algorithm	Makespan	
Choi and Korkmaz (1997)	Polynomial time heuristic procedure	Makespan	
Sotoskov et al. (1998)	Heuristics	Miscellaneous	Batch processing times, sequence independent batch setup times
Xioqing and Noble (1999)	Langrangian relaxation and shifting bottleneck approaches	Weighted sum of squared tardiness	Non zero release times and due dates
Artigues and Roubellat (2002)	Polynomial insertion algorithm	Maximum lateness	Dynamic scheduling
Ballicu, Giua, and Scatzu, (2002)	Mixed integer linear program	Makespan	
Mason, Fowler and Carlyle (2002)	Modified shifting bottleneck heuristic	Weighted tardiness	Reentrant product flow
Lockett and Muhlemann (1971)	Branch and bound , heuristic rules	Total number of tool changes	
Gupta (1982)	Branch and bound	Total setup cost	Reflowing is allowed

Table 3.4-Continue

Authors	Technique	Objective	Constraints
Candido, Khator, and Barcia (1998)	Local improvement procedure, genetic algorithm, hill climbing	Mean completion time	Several subassembly levels, alternative processing plans, multiple resources for a process, resource calendars, batch overlaps
In Chan and Dae-Sik (2002)	Local search algorithm	Makespan	

CHAPTER FOUR

APPLICATION OF PROBLEM

4.1. Introduction

This chapter presents a scheduling problem for a job shop in which more than one machines are available at each work center in a press department of an oven company. Different scheduling rules are applied and the results are compared considering the objective values of work in process, proportion of tardiness, mean tardiness and flow time measures. The main purpose is to observe the reaction of each dispatching rules to scheduling process with sequence dependent setup times. Experimental design is also used to evaluate the effects of factors such as dispatching rules and amount of setup times over the objective functions stated above and to determine the interaction effects between these factors.

4.2. The Environment of Implementation

4.2.1. Company Information

TEBA GÜNKOL , an oven producing company, was founded in İzmir Torbalı. It produces different types of ovens for both the inside market and the foreign countries. Today it is one of the biggest companies in its sector although it is in a heavy economic crisis. This study may also help the company to get over its problems in the areas of increasing productivity and efficiency of the production planning process.

The factory is composed of 7 departments in production level. The flow of materials between the departments is shown in Figure 3.

At Günköl, rolls of sheets usually are bought from well-known industrial companies. These rolls of sheets are cut and converted into different sizes of plain sheets as required orders in the roll cutting workshop. Some plain sheets are distributed to the subcontractors and the remaining sheets are sent to press workshop

for further processes. The reasons, to send some of the sheets to subcontractors, are to lower the unit price of material by purchasing in large amounts and to reduce the risk of variation of material quality by making wholesale.

The processes in press workshop are done in two ways; main sheet shaping with the help of hydraulic process and shaping with eccentric presses which need limited motion availability.

Welding process may occur by combining the press products with center welding. Materials which are combined by more than one welding process get into coating process according to their color alternatives with the endurance to heat and scratching. In the coating process, first the sheets are hanged to the hooks and after injected coat dust, materials are coated in 700-800 °C furnaces. Consequently, surfaces are obtained heat and shock resistance.

At the serigraphy department, the words or symbols are written to painted indicator panels. At the end, semi-products are sent to Assembly Line Workshop. These products are distributed to the assembly lines of the workshop according to their sub-production plans. The final product is formed by combining all the required items coming from different workshops or subcontractors.

4.2.2. Problem Statement

The press workshop, whose problem motivates this thesis, is one of the seven departments in the plant. It produces several types of sheets at different shapes and sizes which are the main parts of the ovens. The press workshop's objective is to meet demand of parts subject to constraints on long setup times and alternative machines availability.

4.2.3. Description of Press Workshop

Press workshop supplies different dimensional and kind of plain sheets as resources which came from roll cutting workshop, sub-contractors and vendors. See Figure 4.1. The press workshop is the heart of the plant as the required parts in every

department are sent from press workshop. That is why; the workload of press workshop is the heaviest one in the plant.

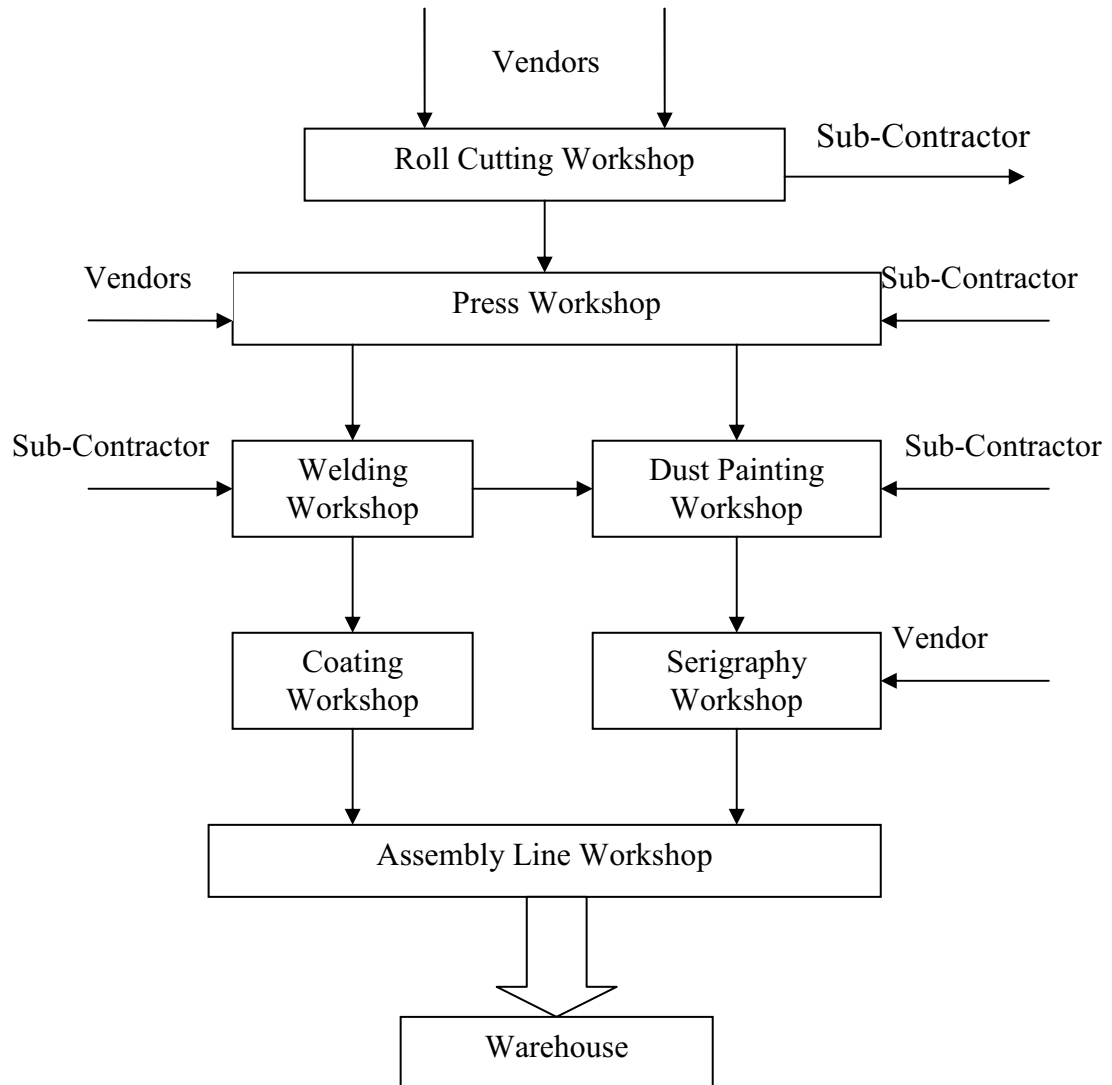


Figure 4.1 Günköl general layout and process flow

To meet the heavy workload, there is more than one for each type of machine at the press workshop. It can be accepted as a flexible job shop in which alternative machines are available for the same process. Totally, there are 18 machines located in six different work centers. The first work center consists of 3 guillotines. The second work center which consists of 3 eccentric presses. The third and fourth work centers which consist of 2 semi automatic heavy hydraulic presses and 2 skirt cutting

machines respectively. The fifth and sixth work centers which consist of 4 hydraulic machines and 4 bend sets respectively. The machines in each work center have the same capability in terms of working speed and machine capabilities. So a part can be processed at any of the available machine in the same period of time in a workshop. The types, codes and number of machines for each type are summarized at Table 4.1.

Table 4.1 List of machines in press workshop

Machine type	Number of Machines	Machine Code
<i>Guillotine</i>	3	DG1260
<i>Eccentric Press</i>	3	PE1500
<i>Semi Automatic Heavy Hydraulic Press</i>	2	PH5000
<i>Skirt-Cutting Machine</i>	2	DE0850
<i>Bend Set</i>	4	SD4000
<i>Hydraulic Press</i>	4	PH1500

In press workshop over 1000 types of products are produced. Besides the main types, there are also subtypes (different versions) of products. Each type of products has different routes in press workshop and each type of product requires more than one process which must be accomplished at different work center. The parts of the same subtype have the same process routes through the press workshop.

For each type of parts there are different types of moulds. Through its route a part needs different moulds at each work center for different operations. For the same subtypes, the same moulds are used for the same processes. Only a small adjustment is made to adapt the mould for the work on another subtype. For different subtypes, different moulds are used even for the same operations on the same machines. Considering the variety of parts there are over 1000 moulds in the press workshop.

The main problem is the long time consumption during changing the moulds. Because they are over 100 kg and so it is a heavy operation to change a mould. Approximately it takes almost half an hour to change a mould to process different type of parts sequentially. But if the same subtypes are processed consecutively, then the same mould is used. The adjustment on the mould takes only 5 minutes which is really low compared to change with a new mould.

There is only one mould for an operation of a part type on a machine. That is why, although there are alternatives for the same type of machines, the same subtypes cannot be processed at the same time in the same work center. The job should wait for the end of the other's finish of its process to get the mould. As a result, this mould constraint limits the effective use of machines in the same work center.

In a highly competitive market the usage of time efficiently provides big advantages to meet the customer demands. Considering this objective, it is clear that the subtypes should be processed consecutively to reduce the time consumption by setups. But it is not also enough to consider the problem as only reducing the setups. Because all orders have tight due dates that must be met. If only it is desired to reduce the setup times by processing the same subtypes consecutively, the risk of going beyond the desired due dates occurs. For instance, a part, which has an urgent due date, may not be processed for a long time as subtypes of another part are processed consecutively although their due dates are less tight. This is a problem between meeting due dates and reducing setup times. The balance should be achieved in a way that while the setup times are reduced as much as available, the due dates should be also met.

The plant works 8 hours per shift, 3 shifts a day, and 5 days per week. It results 120 hours per week.

4.3. Assumptions

The solution methodology is also based on the following assumptions:

1. The order or the precedence of the operations for each order is fixed and known.
2. Processing times are same for the same operation performed on alternative machines in a work center.
3. Tool changing times are considered as sequence dependent setup times.
4. Setup times are independent of machines (though sequence dependent), i.e., the set up times from one specific operation to another are the same on different machines.
5. Parts are produced in batches and the order quantity of a part determines the batch size of that part.
6. A machine can process at most one part (with only one mold) at a time.
7. There is no machine failure during the scheduling horizon.
8. Preemption of the job strings is not allowed, i.e., once they are started, they should be processed until completion.
9. All processing times of jobs are stochastic as it is calculated considering the batch sizes which are created in uniform distribution.
10. Setup times are assumed deterministic.
11. Operators are continuously available for the specified scheduling horizon.
12. Operator transfer times between machines –in relation to job processing times- are negligible.

13. All the operators are fully cross-trained workers.

14. The parts that share the same mould have an ability to be processed on the identical set of machine types with the same amount of operator requirements.

15. A machine of the same type is assumed identical in respect of operation capabilities and working speed.

16. The input queue capacities at each machine are assumed as infinite.

4.4. Constraints

- There is only one mould for each product type. That's why although there are idle machines in a work center, parts which are the same subtypes cannot be processed at the same time at different machines.
- Each job has a definite due date that must be met for customer satisfaction.

4.5. The System Modeling and Simulation

As expressed at the aim of this thesis, different simulation models are constituted for each dispatching rule. ARENA 3 software is used for the creation of models.

Table 4.2 Characteristics of the scheduling rules

<u>DUE DATE</u>	SETUP	INFORMATION
	Yes	No
Yes	Policy 4	Policy 2
No	Policy 3	Policy 1

The Policies described at Table 4.2 are classified according to whether they consider setup and due dates in scheduling process. The detailed description of the

each Policy's concept and the algorithm of the scheduling process are given at the following subsections for each rule.

Policy 1: The queuing theory of this policy is based on the shortest processing time rule. According to this policy the part which has the least processing time is processed first. This heuristic considers neither the setup requirement nor the job due date information. It is effective for flow time related performance. The logic of the model is written below;

- 1) Create an entity according to the distribution function
- 2) Assign the required parameters to entity such as,
 - a) The part type with equal probability
 - b) The batch size according to its distribution function
 - c) The enter time into the system
- 3) According to part type calculate the total process time of part
- 4) According to the actual simulation time assign the due date
- 5) Send the part to its first station for processing according to its route
- 6) At the work center, initially hold the arrival time to the work center
- 7) For each machine in that work center and its queue pair, investigate whether it is the only part or not.
 - a) If it is the only part do nothing and go step 8
 - b) Otherwise update the total workload of that machine- queue pair.
- 8) Search if there is a part, which is the same subtype, just being processed at any of the machines in that work center
 - a) If so, go to step 11 to enter the queue of that machine
 - b) Otherwise do nothing and go to step 9
- 9) Search the entire machine queues in that work center to find a same subtype waiting for processing.
 - a) If there is a same subtype in one of machine's queues, go to step 11 to enter the queue of that machine
 - b) Otherwise do nothing and go to step 10
- 10) Investigate the workload of each machine's workload in the work center and send the entity to the queue of the machine with least workload.

- 11) Calculate the processing time of entity and update the workload of the machine
- 12) Enter the queue of the machine and give a priority to the part considering its process time. The less is the process time, the higher precedence gets the part.
- 13) When both the operator and the machine are idle, seize the machine for processing.
- 14) Check the status of the machine,
 - a) If it is same with the part type then assign setup time as 5 minutes, assign the status of machine as the part type and update the total workload of the machine.
 - b) Otherwise assign the setup time as 27 minutes and assign the status of the machine as the part type.
- 15) Make the setup process.
- 16) Process the part in machine for the defined period of time.
- 17) After the completion of process on the part release the operator and machine.
- 18) Search for the condition of the idleness of machine and the emptiness of queue
 - a) If so then update the total workload of the machine as zero
 - b) Otherwise do nothing and go to step 19.
- 19) Send the machine to the next stop at its route.

Policy 2: The queuing theory of this policy is based on selecting the part with smallest critical ratio. According to this policy the part which has the least slack and the most remaining process time has the highest precedence to be processed. While this heuristic does not consider the setup requirement of jobs, it is effective for the due date related shop performance measures. The logic of the model is written below;

- 1) Create an entity according to the distribution function of inter arrival time.
- 2) Assign the required parameters to entity such as,
 - a) The part type with equal probability
 - b) The batch size according to its distribution function
 - c) The enter time into the system
- 3) According to part type calculate the total process time of part

- 4) According to the actual simulation time assign the due date and the remaining slack.
- 5) Send the part to its first station for processing according to its route
- 6) At the work center, initially hold the arrival time to the work center
- 7) For each machine in that work center and its queue pair, investigate whether it is the only part or not.
 - a) If it is the only part do nothing and go to step 8
 - b) Otherwise update the total workload of that machine-queue pair.
- 8) Search if there is a part, which is the same subtype, just being processed at any of the machines in that work center
 - a) If so, go to step 11 to enter the queue of that machine
 - b) Otherwise do nothing and go to step 9
- 9) Search the entire machine queues in that work center to find a same subtype waiting for processing.
 - a) If there is a same subtype in one of machine's queues, go to step 11 to enter the queue of that machine
 - b) Otherwise do nothing and go to step 10
- 10) Investigate the workload of each machine's workload in the work center and send the entity to the queue of the machine with least workload.
- 11) Calculate the processing time of entity and update the workload of the machine
 - a) If the queue of the machine is empty and the machine is idle then, go to step 12
 - b) If the queue of the machine is empty but the machine is busy then, just hold the enter time of part into the queue of machine and go to step 12
 - c) If the queue is not empty and the machine is busy then, calculate the waiting time, update the slack value of part, hold the enter time of part into the queue of machine and finally go to step 12.
- 12) Calculate the priority value of the part,
 - a) If slack value is greater than zero calculate the priority by dividing the slack value with total remaining process time
 - b) Otherwise calculate the priority by multiplying the slack value with total remaining process time.

- 13) Enter to the queue and sequence in the queue according to its priority. A smaller priority part has, earlier it is processed.
- 14) When both the operator and the machine are idle, seize the machine for processing.
- 15) Check the status of the machine,
 - a) If it is same with the part type then assign setup time as 5 minutes, assign the status of machine as the part type and update the total workload of the machine.
 - b) Otherwise assign the setup time as 27 minutes and assign the status of the machine as the part type.
- 16) Search the condition of the emptiness of queue
 - a) If so then update the value of waiting time as 0
 - b) Otherwise do nothing and go to step 17
- 17) Make the setup process.
- 18) Process the part in machine for the defined level of time.
- 19) After the completion of process on the part release the operator and machine.
- 20) Search for the condition of the idleness of machine and the emptiness of queue
 - a) If so then update the total workload of the machine as zero and go to step 21
 - b) Otherwise do nothing and go to step 21
- 21) Update the total remaining process time and the slack value
- 22) Send the machine to the next stop at its route.

Policy 3: The queuing theory of this policy is based on selecting the part which is the same subtype with the part that is just being processed. If there is not such a part, then the parts are processed according to first in first out (FIFO) scheduling rule. A variable called “Line value” is designated to each incoming part which is used to define the priority of parts in queue. The logic of the model is written below;

- 1) Create an entity according to the distribution function
- 2) Assign the required parameters to entity such as,
 - a) The part type with equal probability
 - b) The batch size according to its distribution function
 - c) The enter time into the system

- 3) According to part type calculate the total process time of part
- 4) According to the actual simulation time assign the due date
- 5) Send the part to its first station for processing according to its route
- 6) At the work center, initially hold the arrival time to the work center
- 7) For each machine in that work center and its queue pair, investigate whether it is the only part or not.
 - a) If it is the only part do nothing and go step 8
 - b) Otherwise update the total workload of that machine- queue pair.
- 8) Search if there is a part, which is the same subtype, just being processed at any of the machines in that work center
 - a) If so, go to step 11 to enter the queue of that machine
 - b) Otherwise do nothing and go to step 9
- 9) Search the entire machine queues in that work center to find a same subtype waiting for processing.
 - a) If there is a same subtype in one of machine's queues, go to step 11 to enter the queue of that machine
 - b) Otherwise do nothing and go to step 10
- 10) Investigate the workload of each machine's workload in the work center and send the entity to the queue of the machine with least workload after assigning a new line value to represent only that subtype and go to step 12.
- 11) Assign the predetermined line value of that subtype as new coming part's line value which will determine the priority index of that part in machine queue.
- 12) Calculate the processing time of entity and update the workload of the machine
- 13) Enter the queue of the machine and sequence the parts according to their line values from lowest to highest values.
- 14) When both the operator and the machine are idle, seize the machine for processing.
- 15) Check the status of the machine,
 - a) If it is same with the part type then assign setup time as 5 minutes, assign the status of machine as the part type and update the total workload of the machine.

- b) Otherwise assign the setup time as 27 minutes and assign the status of the machine as the part type.
- 16) Make the setup process.
- 17) Process the part in machine for the defined level of time.
- 18) After the completion of process on the part release the operator and machine.
- 19) Search for the condition of the idleness of machine and the emptiness of queue
 - a) If so then update the total workload of the machine as zero
 - b) Otherwise do nothing and go to step 20.
- 20) Send the machine to the next stop at its route.

Policy 4: This heuristic considers the setup requirement of jobs. It scans the queue for a similar subtype that its processing is just finished on the machine. When there is no identical job, it selects a job with the smallest critical ratio which is the same with the ratio used in Policy 2. When there are parts which belong to same subtypes, they are also sequenced consecutively considering their critical ratios between each other. This heuristic behaves like Policy 2 when there are no same subtypes at the queue; otherwise it behaves like Policy 3. By this way, this heuristic supplies both the minimization of setup time and the improvement of due date performance. The logic of the model is written below;

- 1) Create an entity according to the distribution function
- 2) Assign the required parameters to entity such as,
 - a) The part type with equal probability
 - b) The batch size according to its distribution function
 - c) The enter time into the system
- 3) According to part type calculate the total process time of part
- 4) According to the actual simulation time assign the due date and the remaining slack value
- 5) Send the part to its first station for processing according to its route
- 6) At the work center, initially hold the arrival time to the work center
- 7) For each machine in that work center and its queue pair, investigate whether it is the only part or not.
 - a) If it is the only part do nothing and go step 8

- b) Otherwise update the total workload of that machine- queue pair.
- 8) Search if there is a part, which is the same subtype, just being processed at any of the machines in that work center
 - a) If so, go to step 11 to enter the queue of that machine
 - b) Otherwise do nothing and go to step 9
- 9) Search the entire machine queues in that work center to find a same subtype waiting for processing.
 - a) If there is a same subtype in one of machine's queues, go to step 11 to enter the queue of that machine
 - b) Otherwise do nothing and go to step 10
- 10) Investigate the workload of each machine's workload in the work center and send the entity to the queue of the machine with least workload
- 11) Search for the queue and the machine if it is empty and idle respectively.
 - a) If the queue is empty and the machine is idle, then hold the enter time of part into the queue, calculate the process time of part in the machine and update the workload of the machine, increase the value of counter which indicates the number of parts in machine-queue pair consecutively.
 - b) If the queue is empty but the machine is not idle, then hold the enter time of part into the queue, calculate the process time of part in the machine and update the workload of the machine consecutively.
 - c) If the queue is not empty, then calculate the additional waiting times of parts just arrived to queue, update the slack value of new coming part, hold the enter time of the part into the queue and, update the total workload of machine consecutively.
- 12) To determine the priority index of each part in a machine queue follow the procedure described below,
 - a) Search the queue if another part which is a member of same subtype exists in machine queue
 - i) If so, check the value of slack
 - i.1) If the slack value of part is smaller than zero, calculate the priority index by multiplying total process time and slack values and adding the index value of that subtype which was calculated before.

i.2) If the slack value of part is equal or bigger than zero, calculate the priority index by dividing slack with total process time value and adding the index value of that subtype which was calculated before.

ii) Otherwise, go to step b

b) Check if there is a part just being processed at the machine which belongs to the same subtype.

i) If so, check the value of slack

i.1) If the slack value is equal or bigger than zero, then designate the index value of the part's subtype as a very low value to take the part to the front of queue and calculate the priority index by dividing slack with total process time value and adding the index value of that subtype.

i.2) If the slack value is less than zero, then designate the index value of the part's subtype as a very low value to take the part to the front of queue and calculate the priority index by multiplying total process time and slack values and adding the index value of that subtype

ii) Otherwise, check the value of slack.

ii.1) If the slack value is equal or bigger than zero, then designate the index value of the part's subtype by dividing the slack value with the total process time of the part, and calculate the priority index by dividing slack with total process time value and adding the index value of that subtype .

ii.2) If the slack value is less than zero, then designate the index value of part's subtype by multiplying the total process time and slack values of the part, and calculate the priority index by multiplying the total process time and slack values of the part and adding the index vale of that subtype.

13) Enter the queue of the machine and sequence the parts according to their priority index values from lowest to highest values.

- 14) When both the operator and the machine are idle, seize the machine for processing.
- 15) Check the status of the machine,
 - a) If it is same with the part type then assign setup time as 5 minutes, assign the status of machine as the part type and update the total workload of the machine.
 - b) Otherwise assign the setup time as 27 minutes and assign the status of the machine as the part type.
- 16) Search for the number of parts waiting in queue,
 - a) If the queue is empty, then designate the value of waiting time as zero
 - b) Otherwise do nothing, and go to step 17.
- 17) Make the setup process.
- 18) Process the part in machine for the defined level of time.
- 19) After the completion of process on the part release the operator and machine.
- 20) Search for the condition of the idleness of machine and the emptiness of queue
 - a) If so then update the total workload of the machine as zero
 - b) Otherwise do nothing and go to step 21.
- 21) Update the total process time and slack values of part after it finishes its job in that work center.
- 22) Send the machine to the next stop at its route.

4.6. Decomposition of the Problem

4.6.1. Input Analysis

All the input data of the system has been taken from the firm's historical database except the interval time of arrivals. Normally, all the production orders are given at time zero and ready to be realized because the production planning is given to production department which consists of the three day's production at the future. So each part's production order is given three days before its production time. But to make a study about the comparison of the performance levels of different scheduling rules, the parts should be sent through the simulation run continuously. It is not possible to send entire parts into the simulation model at time zero. The inter arrival

time are generated from exponential distribution which has a mean of 37 minutes. This value is chosen to hold the system's average utilization over 80 percent when shortest processing time scheduling rule is used. Under lower utilization rates the comparison of scheduling rules may not give completely accurate results.

Considering 500 historical data of batch sizes, the distribution of batch sizes is determined as uniform distribution with minimum and maximum values of 200 and 600 respectively. The historical data and the test statistics are given at Appendix A.

Totally 15 types of parts are introduced into the system which have different routes and processing times. The parts are created with equal probability into the system. These parts represent the main types of products so none of them are subtypes of each other. The routes and processing times of each part types are given at Appendix B.

The processing times are taken from the historical database of the plant. As the unit processing times are so short and the interference of worker on machines during processing is so less, that the processing times are determined as deterministic.

The due dates are assigned by the following equation when a job arrives at the shop. (Kim & Bobrowski, 1994)

$$Due\ Date = Next\ Shift + k \times (Total\ Processing\ Time + NOP \times AST),$$

where k = a positive real number,

NOP = number of operations in the job which is determined as 4

AST = average set up time

k is chosen as 4,7 to arrange the due dates as tight. This level has been used by Kim & Bobrowski to have a result of 50% tardy jobs.

Average setup time is determined as 27 minutes which is fifth of the average processing time. This proportion is adequate to show the difference in performance of sequencing rules, but not so large to give setup oriented rules too much

advantage. If the subtypes are processed consecutively then the average setup time is chosen as 5 minutes which is enough to make the additional adjustments on the mould without changing it.

4.6.2. Verification and Validation

Model verification ensures that the computer programming and implementation of the conceptual model are correct. For this purpose different methods are used. In this thesis there are four different models, each of which represents one of scheduling rules. So the verification has been done for each individual model independently. During the verification process trace element is used to observe the movements of a part through the model. By the help of trace, it is possible to observe that whether a part moves in a way harmonized to the system logic or not. By the help of trace it was seen that the parts followed the routing specified, and the correct tools and process times were used. Specifically, the following events were checked for correct occurrence:

- Creation of Order

Flow of work part: Creation, choosing the most appropriate machine in a work center, waiting for machine, seize of machine and operation, truly further movements in the system between work centers, completion of an order, data collection on performance measures

- Use of alternate routing
- Priority rule (queue discipline) operation

Animation has been used as a second tool for verification process. Sometimes it is not possible to realize the mistakes by checking the outputs of the model. Animation provides a visual control over the model which is helpful to see the part's flow. Animation also gives the opportunity to see the model's behavior and situation during the simulation. For example bottleneck machines can be determined as they will probably have long queues.

At the validation phase, each model is tested under different conditions and then the results of the initial state and the final situation are compared to check if the models act in a way that they should. For this purpose, Sensitivity Analysis is used and the parameter is chosen as the arrival time. Initially, the inter arrival time is chosen as expo (37) then it is increased to expo (38).

Each of the system is run for 20 replications each of which continues 132000 minutes long with a warm up period of 12000 minutes. The performance measures that will be compared are the flow time, proportion of tardiness and the mean tardiness. The results are given at tables.

Table 4.3 For inter arrival time with expo (37)

	FLOW TIME (hour/job)	PROPORTION OF TARDINESS (%)	WIP (part/minute)
POLICY 1	24,46	6,07	81,98
POLICY 2	24,38	8,20	43,94
POLICY 3	21,43	5,60	33,34
POLICY 4	17,28	4,10	26,39

Table 4.4 For inter arrival time with expo (38)

	FLOW TIME (hour/job)	PROPORTION OF TARDINESS (%)	WIP (part/minute)
POLICY 1	21,01	4,24	64,42
POLICY 2	22,76	6,45	38,43
POLICY 3	18,28	2,74	27,36
POLICY 4	15,31	2,18	22,45

According to the values at Tables 4.3-4.4, it is obvious that each of the models act as expected when the inter arrival time of parts is increased. As the parts come more seldom, the system is in less congestion and this reduces the average flow time of parts in system. As the flow time is decreased, fewer parts exceed their due dates and the total number of parts in system decreases.

4.6.3. Defining Warm up Period

The warm up period is determined considering the objective value of flow time. For each scheduling policy, the models are run for 100000 minutes. The graphs below show the cumulative values of flow time in simulation run.

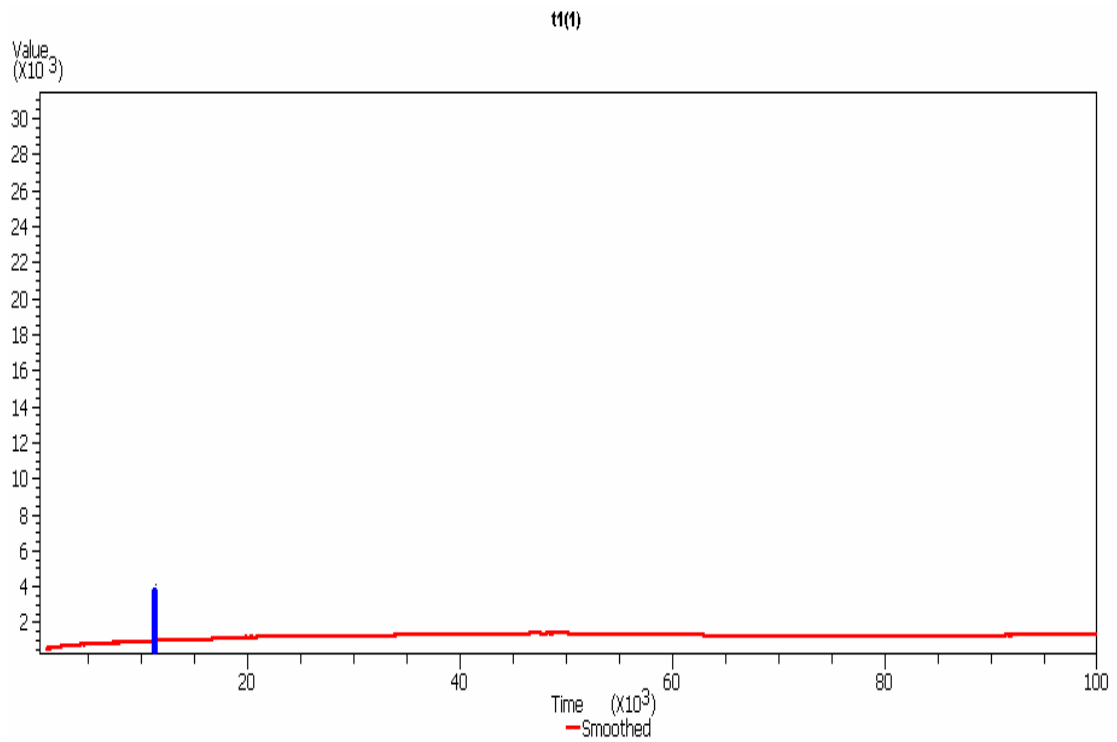


Figure 4.2 Moving average plot for policy 1

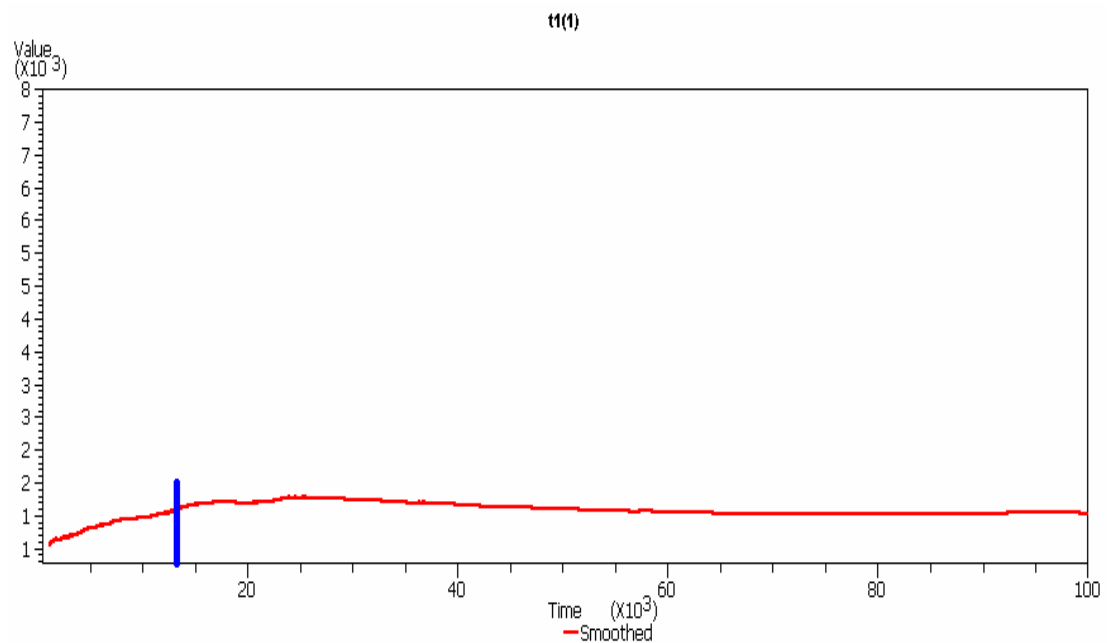


Figure 4.3 Moving average plot for policy 2

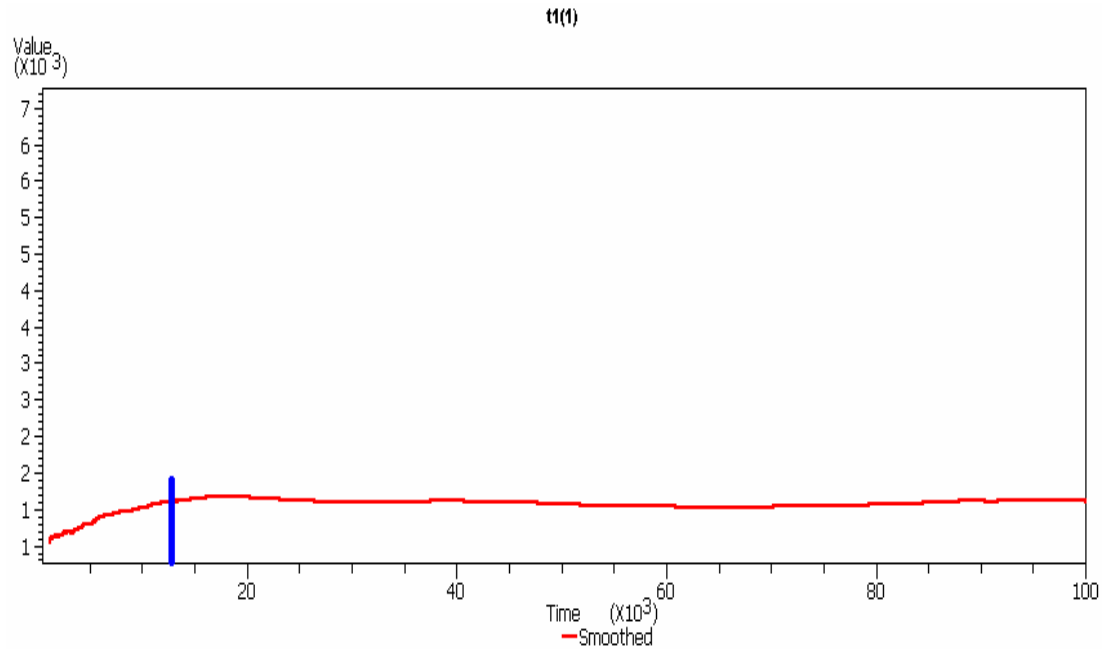


Figure 4.4 Moving average plot for policy 3

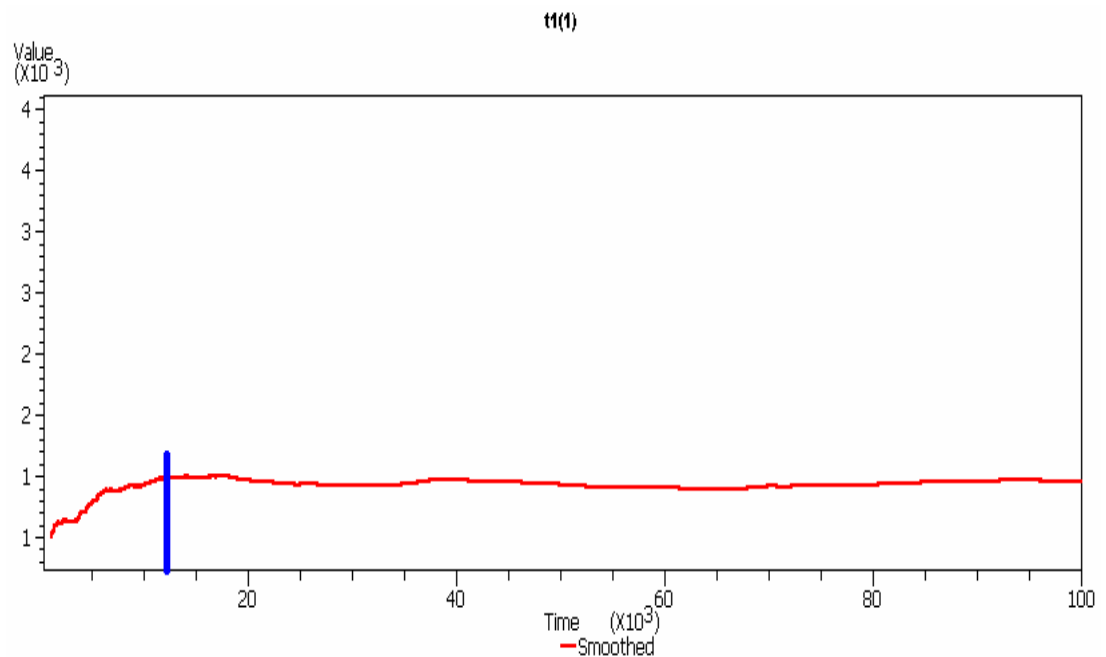


Figure 4.5 Moving average plot for policy 4

Considering the graphs above the warm up period is chosen as 12000 minutes. Generally, it is accepted that the average run length should be at least 10 times greater than the warm up period so the run length of each run is determined as 132000 minutes. (Banks et al, 2000, p 432)

A reasonable number of replications to use in most situations is 10 and it probably makes little sense to have more than 20 because improvements in the t-statistic diminish beyond that point. (Pegden et al, 1990, p 188) That's why the number of replications is chosen as 20.

4.6.4. Variance Reduction Techniques

Since the objective was to measure the relative performance of alternative rules, it was logical to compare them under identical conditions. The use of common random numbers variance reduction scheme ensured that each job arrived at the same time and assigned identical set of processing times for all the rules analyzed. This method gives results with greater statistical precision such as smaller confidence intervals. The basic idea is that we should compare the alternative configurations “under similar experimental conditions” so that we can be more confident that any observed differences in performance are due to differences in the system configurations rather than to fluctuations of the “experimental conditions” For this purpose SEEDS Element was used in which different streams are introduced both for the interarrival time and total processing times of a part.

4.7. Evaluation of Proposed Methods

This chapter presents an evaluation and comparisons of the proposed methods considering different performance measures. For the purpose of evaluation, experimental design is used considering two factors with different levels. The factors are chosen as;

1. Control Approach: The scheduling and controlling methodology is a factor in the experiment. Four different approaches (levels) were evaluated which are described at Problem Statement
2. Setup time: Two levels of setup times are defined. The first one is fifth of average processing time and the other is almost fourth of average processing time.

The factorial design is summarized by Table 4.5

Table 4.5 Factorial design of simulation experiments

SETUP TIME CONTROL POLICY	LOW (27 minutes)	HIGH (32 minutes)
POLICY 1		
POLICY 2		
POLICY 3		
POLICY 4		

4.7.1. Analysis of Experimental Results

A total of 160 experiments were carried out, 20 replications for each of 8 combinations of the factor levels. Each experiment resulted in observations of 4 responses:

1. Average flow time, in hours
2. Average tardiness, in hours
3. Proportion of late jobs in total production
4. Work in process

For all 4 responses, the linear model is;

$$X_{ijm} = \mu + C_i + S_j + CS_{ij} + \varepsilon_{m(ij)}$$

where

X_{ijm} for Average flow time, Average tardiness, Proportion of late jobs in total production, Work in process

μ = Average Response over all the populations

C_i = Effect of the control policy (i = 1,...,4)

S_j = Effect of setup time (j=1,2)

CS_{ij} = Interaction between control policy and setup time

$\mathcal{E}_{m(ij)}$ = Random error for replication m within the cell i,j

The conclusions drawn from the output are presented in the following sub-sections.

4.7.1.1. Flow Time

With respect to the dependent variable of average flow time, the analysis of variance is presented below in Table 4.6

Table 4.6 ANOVA with average flow time as dependent variable

Source	DF	SS	MS	F	P
Policy	3	2795	931,7	21,93	0
Setup	1	801,6	801,6	18,87	0
Interaction	3	286,3	95,4	2,25	0,085
Error	152	6457	42,5		
Total	159	10339,8			

For the evaluation of interaction effects, F value is calculated for $\alpha = 0,05$. As $F^* = 2,25 < F_{0,95,3,152} = 2,60$, it can be said that the policy and setup don't interact in their effects on average flow time. The P-value of this test is $P\{F(3,152) > 2,25\} = 0,085$. The lack of interaction effects is not statistically significant as $P \geq \alpha$. The plot given in Figure 4.6 also shows the lack of interaction between main factors. Although the curves are not completely parallel to each other, they don't also intersect.

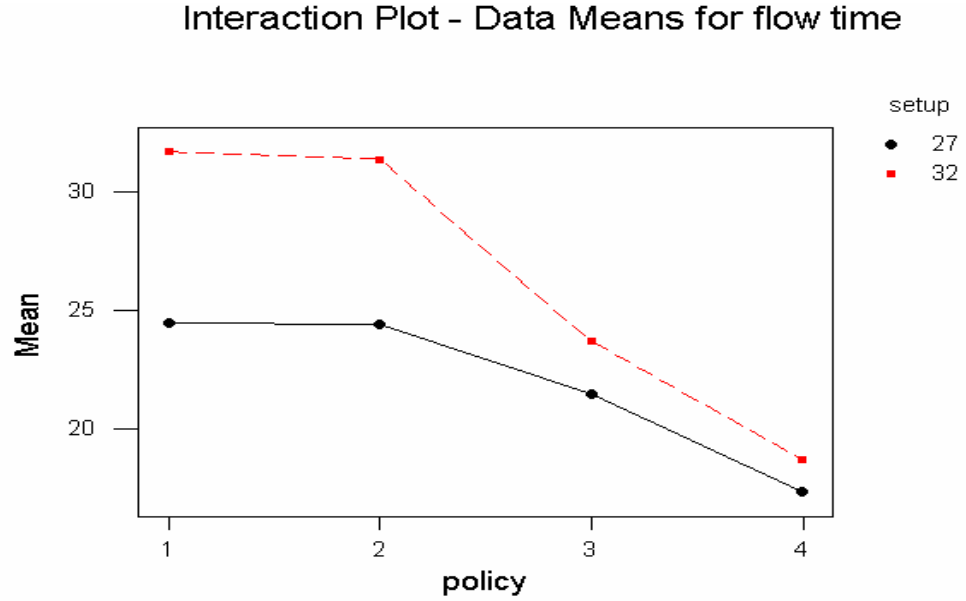


Figure 4.6 Interaction Plot for flow time performance measure

Since the factors do not interact, the effects of main factors can be calculated. For the factor of policy the value of $F_{0,95,3,152}$ is calculated as 2,6. As $F^*=21,93 > 2,6$ then it is clear that the Policy factor level means are not equal, or that some definite effects associated with policy type exist. The P-value of this test is $P\{F(2,6) > 21,93\} = 0$. As $\alpha \geq P$ value, it is clear that the main effect of policy factor is statistically significant for the flow time performance measure. In the same way the main effects of setup factor is evaluated considering the F value with F^* from ANOVA table. As $F^*=18,87 > F_{0,95,1,152} = 3,84$ then it is also clear that the factor of setup time's level means are not equal, or that some definite effects associated with setup time factor exist. The P-value of this test is $P\{F(3,84) > 18,87\} = 0$. As $\alpha \geq P$ value, it is clear that the main effect of setup factor is statistically significant for the flow time performance measure.

To find the significant differences among the control policies, Tukey test for multiple pairwise comparisons was carried out. The results of Tukey test is given at Tables C.1-C.2 in Appendix C. A significance level of 0.05 was used. The results for flow time measure are shown at Figure 4. 7. Mean values of performance measure

are shown in parenthesis. A line is drawn, considering the calculated confidence intervals, under any set of means for which the differences are not statistically significant at $\alpha = 0.05$.

The confidence intervals of multiple comparisons between control policies are obtained as;

$$\begin{array}{ll}
 -3,54 \leq \mu_1 - \mu_2 \leq 3,94 & 1,57 \leq \mu_2 - \mu_3 \leq 9,06 \\
 1,78 \leq \mu_1 - \mu_3 \leq 9,26 & 6,14 \leq \mu_2 - \mu_4 \leq 13,63 \\
 6,35 \leq \mu_1 - \mu_4 \leq 13,83 & 0,82 \leq \mu_3 - \mu_4 \leq 8,31
 \end{array}$$

POLICY 4	POLICY 3	POLICY 2	POLICY 1
<u>(17,99)</u>	<u>(22,56)</u>	<u>(27,88)</u>	<u>(28,08)</u>

Figure 4.7 Multiple pairwise comparison of control policies for flow time

From the multiple comparisons, it can be concluded that Policy 4 has statistically significantly the lowest flow time. The ordinary rules (policy 1 and policy 2) show no big differences on flow time performance. Although Policy 3 is significantly better than ordinary rules it is not as good as Policy 4 as there is a statistical significance between these two policies. Although Policy 1 is generally good at flow time measure, here it has a bad performance. Because it doesn't consider the sequence dependent setup times so the mean flow times of jobs dramatically increase. The longer flow times imply a lower throughput rate and lower shop productivity, which cause a slow delivery of goods to the customers. Also, longer flow time has an effect on shop utilization and the WIP. Since the parts are held longer in the system it increases the WIP value. The long flow time is a result of long setup times which don't contribute to the productivity. The loss of productivity causes a lower utilization of machines which must wait for long setup activities.

The mean values of flow time at different levels of setup time are shown at Table 4.7.

Table 4.7 Flow time measure at different levels of setup factor

		POLICY 1	POLICY 2	POLICY 3	POLICY 4
<i>Setup Time (27 min)</i>	Mean (hr/part)	24,46	24,37	21,44	17,29
	Stand. Deviation	4,14	8,81	3,24	2,25
<i>Setup Time (32 min)</i>	Mean (hr/part)	31,71	31,39	23,68	18,69
	Stand. Deviation	6,74	12,78	3,7	2,54

With customer service a key aspect of shop performance, lower mean flow times alone are not sufficient to judge the impact of scheduling on this key element. Table 4.7 shows the standard deviations for flow times for each of the sequencing rules. The ordinary rules (policy 1, policy 2) show greater variation than the setup oriented rules (policy 3, policy 4) besides having the higher mean performance. The implication is that the setup oriented rules give greater customer service by delivering on or about the due date with more accuracy and reliability.

The general results of simulation experiment for flow time objective are shown at Figure 4.8. It can be seen that, although minimum process time is known to perform well on flow time measures, it didn't perform as well as the setup oriented rules (policy 3, policy 4) because of the longer setup time spent per job. Also the

sensitivity against the increase in setup time is much more at ordinary sequencing rules when compared to setup oriented sequencing rules. It is a sign for which sequencing rules should be used under which situations. In the investigation for flow time performance measure setup oriented sequencing rules give much better solutions than the ordinary sequencing rules when the amount of setup time per job increases.

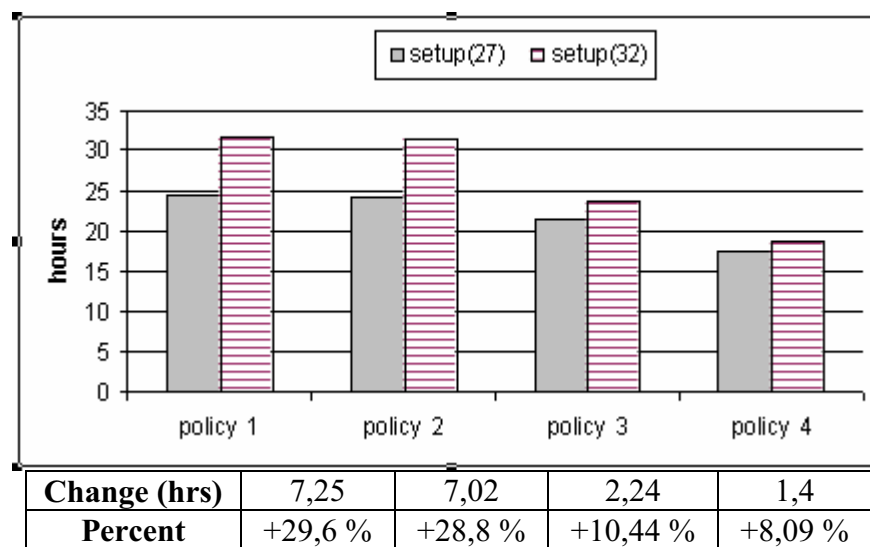


Figure 4.8 Setup time effects over flow time values at different scheduling rules

4.7.1.2. Mean Tardiness

With respect to the dependent variable of average mean tardiness, the analysis of variance is presented below in Table 4.8.

Table 4.8 ANOVA with mean tardiness as dependent variable

Source	DF	SS	MS	F	P
Policy	3	287761	95920	63	0
Setup	1	9712	9712	6,38	0,013
Interaction	3	8261	2754	1,81	0,148
Error	152	231414	1522		
Total	159	537149			

For the evaluation of interaction effects, F value is calculated for $\alpha = 0,05$. As $F^* = 1,81 < F_{0,95,3,152} = 2,60$, it can be said that the policy and setup don't interact in their effects on mean tardiness. The P-value of this test is $P\{F(3,152) > 1,81\} = 0,148$. The lack of interaction is not statistically significant as $P \geq \alpha$. The plot given in Figure 4.9 also shows the lack of interaction between main factors. Although the curves are not parallel, they don't also intersect.

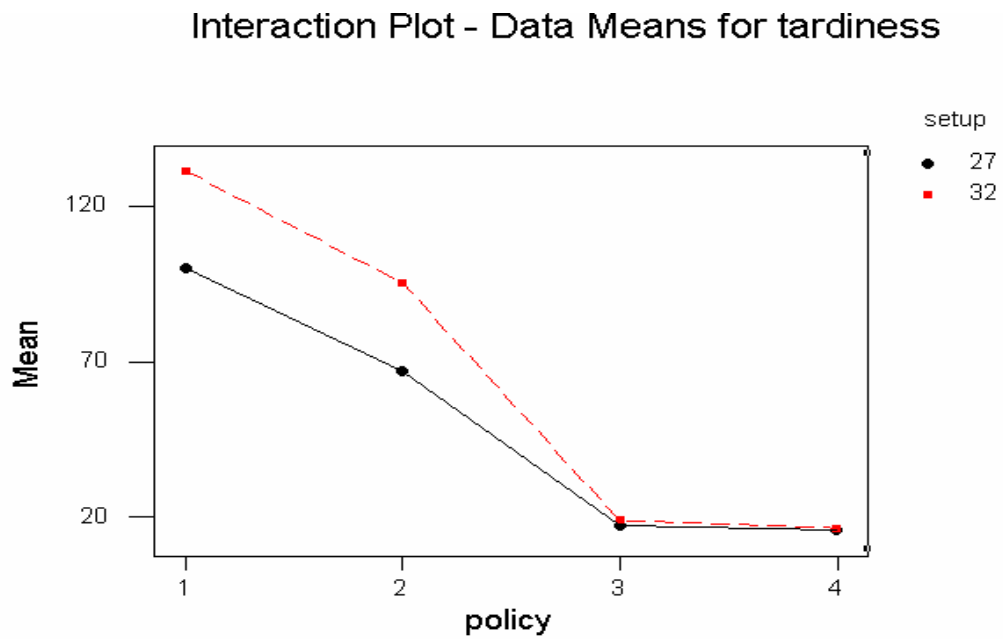


Figure 4.9 Interaction plot for mean tardiness performance measure

Since the factors do not interact, the effects of main factors can be calculated. For the factor of policy the value of $F_{0,95,3,152}$ is calculated as 2,6. As $F^* = 63 > 2,6$ then it is clear that the Policy factor level means are not equal, or that some definite effects associated with policy type exist. The P-value of this test is $P\{F(2,6) > 63\} = 0$. As $\alpha \geq P$ value, it is clear that the main effect of policy factor is statistically significant for the mean tardiness performance measure. In the same way the main effects of setup factor is evaluated considering the F value with F^* from ANOVA table. As $F^* = 6,38 > F_{0,95,1,152} = 3,84$ then it is also clear that the factor of setup time's level means are not equal, or that some definite effects

associated with setup time factor exist. The P-value of this test is $P\{F(3,84) > 6,38\} = 0,013$. As $\alpha \geq P$ value, it is clear that the main effect of setup factor is statistically significant for the mean tardiness performance measure.

To find the significant differences among the control policies, Tukey test for multiple pairwise comparisons was carried out. The results of Tukey test is given at Tables C.3-C.4 in Appendix C. A significance level of 0.05 was used. The results for mean tardiness measure are shown at Figure 4.10. Mean values of performance measure are shown in parenthesis. A line is drawn, considering the calculated confidence intervals, under any set of means for which the differences are not statistically significant at $\alpha = 0.05$.

The confidence intervals of multiple comparisons between control policies are obtained as;

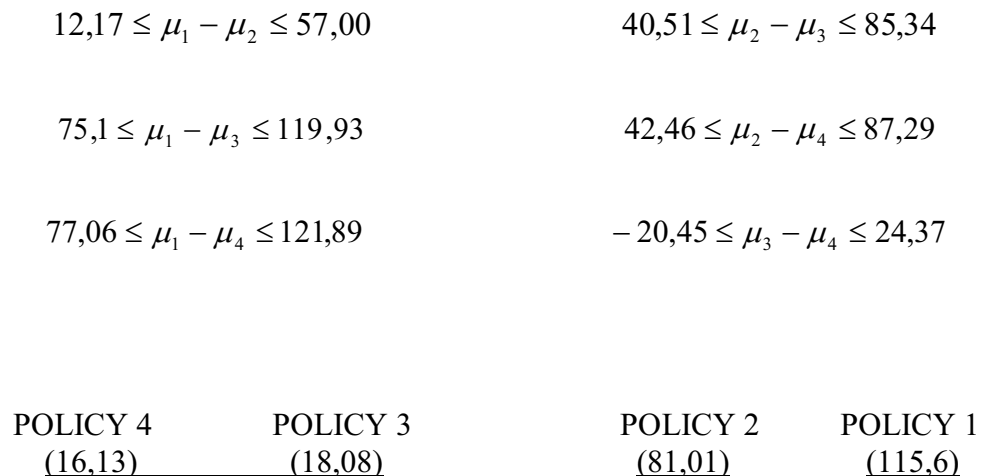


Figure 4.10 Multiple pairwise comparison of control policies for mean tardiness

From the multiple comparison, it can be concluded that setup oriented sequencing rules have significantly the lower mean tardiness values than the ordinary sequencing rules. Although Policy 4 has a better performance than Policy 3, the difference is not statistically significant. Although Policy 2 considers the due date performance measure, it has worse tardiness performance than both of the setup oriented sequencing rules. But it has a statistically significantly better performance than

Policy 1 which doesn't consider the due date performance measure. This is not surprising to see that Policy 1 has the worst performance as it is biased against long processing time jobs causing long processing time jobs to be extremely late. Although both Policy 4 and Policy 2 consider the due date, Policy 4 has a statistically significantly lower mean tardiness because it also considers the sequence dependent setups. This implies the effect of sequence dependent setup over system performance. Policy 4 tends to evenly spread tardiness among jobs, resulting in more jobs to be late but with smaller tardiness.

The mean values of mean tardiness at different levels of setup time are shown at Table 4.9.

Table 4.9 Mean tardiness measure at different levels of setup factor

		POLICY 1	POLICY 2	POLICY 3	POLICY 4
<i>Setup Time (27 min)</i>	Mean (hr/part)	99,96	66,74	17,23	15,73
	Stand. Deviation	30,67	60,08	5,58	7,47
<i>Setup Time (32 min)</i>	Mean (hr/part)	131,25	95,28	18,94	16,52
	Stand. Deviation	36,93	78,19	4,57	6,53

Table 4.9 gives the standard deviations and mean values for mean tardiness of each sequencing rules. As Policy 1 doesn't consider the due date performance measure it has the worst performance. As Policy 2 takes the due date into account, it

gives lower mean tardiness values than Policy 1 but it does not consider the sequence dependent setups. So it causes higher tardiness values than the setup oriented sequencing rules. The ordinary sequencing rules have greater variations which makes them less reliable to use in scheduling process. As a result, the setup oriented rules give greater customer service by delivering on or about the due date with more accuracy and reliability.

The general results of simulation experiment for mean tardiness objective are shown at Figure 4.11. It can be seen that, although minimum process time is known to perform well on flow time measures, it didn't perform well in respect of the mean tardiness because of the longer setup time spent per job. Higher setup times resulted in higher mean tardiness for all sequencing rules as was expected since the production flow times were lengthened. However, the sensitivity against the increase in setup time is much more at ordinary sequencing rules when compared to setup oriented sequencing rules. Setup oriented rules showed a relatively small deterioration as the setup times got higher, this was due to capability to control flow time by reducing setup time. It is a sign for which sequencing rules should be used under which situations. In the investigation for mean tardiness performance measure, setup oriented sequencing rules give much better solutions than the ordinary sequencing rules when the amount of setup time per job increases.

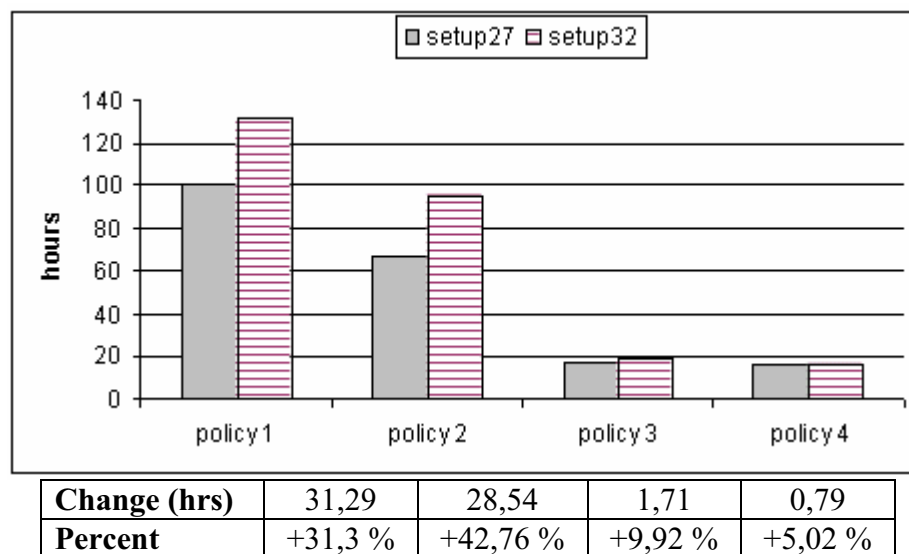


Figure 4.11 Setup time effects over mean tardiness values at different scheduling rules

4.7.1.3. Proportion of Tardiness

With respect to the dependent variable of proportion of tardiness, the analysis of variance is presented below at Table 4.10

Table 4.10 ANOVA with proportion of tardiness as dependent variable

Source	DF	SS	MS	F	P
Policy	3	593,8	197,9	18,56	0,000
Setup	1	126,2	126,2	11,83	0,001
Interaction	3	36,6	12,2	1,14	0,334
Error	152	1620,7	10,7		
Total	159	2377,3			

For the evaluation of interaction effects, F value is calculated for $\alpha = 0,05$. As $F^* = 1,14 < F_{0,95,3,152} = 2,60$, it can be said that the policy and setup don't interact in their effects on proportion of tardiness. The P-value of this test is $P\{F(3,152) > 1,14\} = 0,334$. The lack of interaction is not statistically significant as $P \geq \alpha$. The plot given in Figure 4.12 also shows the lack of interaction between main factors. Although the curves are not parallel, they don't also intersect.

Interaction Plot - Data Means for proportion

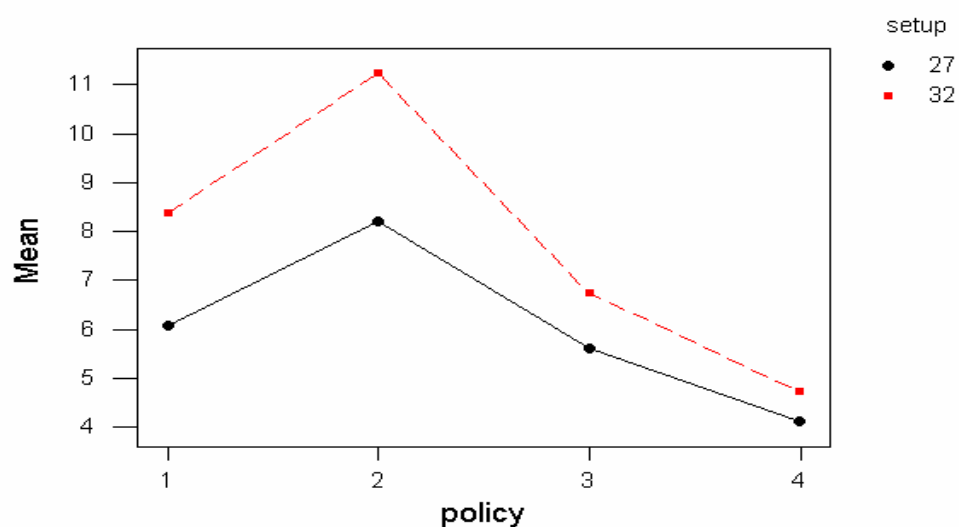


Figure 4.12 Interaction plot for proportion of tardiness performance measure

Since the factors do not interact, the effects of main factors can be calculated. For the factor of policy the value of $F_{0,95,3,152}$ is calculated as 2,6. As $F^*=18,56 > 2,6$ then it is clear that the Policy factor level means are not equal, or that some definite effects associated with policy type exist. The P-value of this test is $P\{F(2,6) > 18,56\} = 0,000$. As $\alpha \geq P$ value, it is clear that the main effect of policy factor is statistically significant for the proportion of tardiness performance measure. In the same way the main effects of setup factor is evaluated considering the F value with F^* from ANOVA table. As $F^*=11,83 > F_{0,95,1,152} = 3,84$ then it is also clear that the factor of setup time's level means are not equal, or that some definite effects associated with setup time factor exist. The P-value of this test is $P\{F(3,84) > 11,83\} = 0,001$. As $\alpha \geq P$ value, it is clear that the main effect of setup factor is statistically significant for the proportion of tardiness performance measure.

To find the significant differences among the control policies, Tukey test for multiple pairwise comparisons was carried out. The results of Tukey test is given at Tables C.5-C.6 in Appendix C. A significance level of 0.05 was used. The results for proportion of tardiness measure are shown at Figure 4.13. Mean values of performance measure are shown in parenthesis. A line is drawn, considering the calculated confidence intervals, under any set of means for which the differences are not statistically significant at $\alpha = 0.05$.

The confidence intervals of multiple comparisons between control policies are obtained as;

$$4,33 \leq \mu_1 - \mu_2 \leq -0,62$$

$$1,67 \leq \mu_2 - \mu_3 \leq 5,42$$

$$-0,81 \leq \mu_1 - \mu_3 \leq 2,93$$

$$3,44 \leq \mu_2 - \mu_4 \leq 7,19$$

$$0,94 \leq \mu_1 - \mu_4 \leq 4,7$$

$$-0,1 \leq \mu_3 - \mu_4 \leq 3,64$$

POLICY 4	POLICY 3	POLICY 1	POLICY 2
<u>(4,4)</u>	<u>(6,17)</u>	<u>(7,23)</u>	<u>(9,72)</u>

Figure 4.13 Multiple pairwise comparison of control policies for proportion of tardiness

From the multiple comparison, it can be concluded that setup oriented sequencing rules have significantly the lower proportion of tardiness values than the ordinary sequencing rules. Although Policy 4 has a better performance than Policy 3, the difference is not statistically significant. Although Policy 2 considers the due date performance measure, it has the worst performance. Because it is based on minimizing the value of tardiness of each jobs but not the total number of delayed parts. By this sequencing rule, a higher priority is given to part which exceeds it's due date so the parts which have time till their due dates wait more in queues which causes a decrease in their slack values too at the next steps. Policy 1 has a statistically significantly better performance than Policy 2 although it doesn't consider the due date performance measure. As a result, the setup oriented rules give greater customer service as they cause less number of lateness in delivering the finished products.

The general results of simulation experiment for proportion of tardiness objective are shown at Figure 4.14. Higher setup times resulted in higher proportion of tardiness for all sequencing rules as it was expected since the production flow times were lengthened. However, the sensitivity against the increase in setup time is much more at ordinary sequencing rules when compared to setup oriented sequencing rules. Setup oriented rules showed a relatively small deterioration as the setup times got higher, this was due to capability to control flow time by reducing setup time. It is a sign for which sequencing rules should be used under which situations. In the investigation for proportion of tardiness performance measure, setup oriented sequencing rules give much better solutions than the ordinary sequencing rules when the amount of setup time per job increases. Between setup oriented sequencing rules Policy 4 reacts less than Policy 3 in case of increase of setup time.

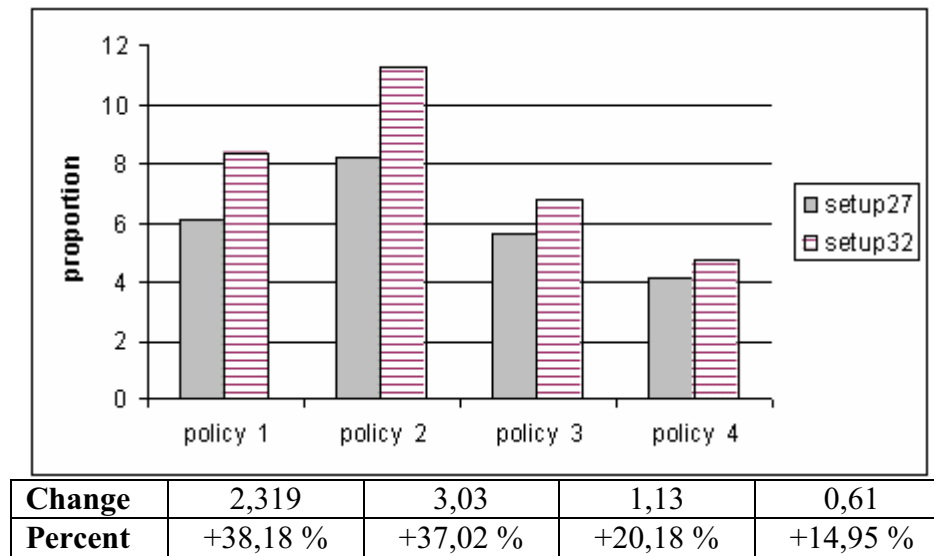


Figure 4.14 Setup time effects over proportion of tardiness values at different scheduling rules

4.7.1.4. WIP

With respect to the dependent variable of WIP, the analysis of variance is presented below at Table 4.11.

Table 4.11 ANOVA with WIP as dependent variable

Source	DF	SS	MS	F	P
Policy	3	114972	38324	54,38	0,000
Setup	1	6739	6739	9,56	0,002
Interaction	3	4962	1654	2,35	0,075
Error	152	107121	705		
Total	159	233794			

For the evaluation of interaction effects, F value is calculated for $\alpha = 0,05$. As $F^* = 2,35 < F_{0,95,3,152} = 2,60$, it can be said that the policy and setup don't interact in their effects on proportion of tardiness. The P-value of this test is $P\{F(3,152) > 2,35\} = 0,075$. The lack of interaction is not statistically significant as $P \geq \alpha$. The plot given in Figure 4.15 also shows the lack of interaction between main factors. Although the curves are not parallel, they don't also intersect.



Figure 4.15 Interaction plot for WIP performance measure

Since the factors do not interact, the effects of main factors can be calculated. For the factor of policy the value of $F_{0,95,3,152}$ is calculated as 2,6. As $F^*=54,38 > 2,6$ then it is clear that the Policy factor level means are not equal, or that some definite effects associated with policy type exist. The P-value of this test is $P\{F(2,6) > 54,38\} = 0,000$. As $\alpha \geq P$ value, it is clear that the main effect of policy factor is statistically significant for the WIP performance measure. In the same way the main effects of setup factor is evaluated considering the F value with F^* from ANOVA table. As $F^*=9,56 > F_{0,95,1,152} = 3,84$ then it is also clear that the factor of setup time's level means are not equal, or that some definite effects associated with setup time factor exist. The P-value of this test is $P\{F(3,84) > 9,54\} = 0,002$. As $\alpha \geq P$ value, it is clear that the main effect of setup factor is statistically significant for the WIP performance measure.

To find the significant differences among the control policies, Tukey test for multiple pairwise comparisons was carried out. The results of Tukey test is given at Tables C.7-C. 8 in Appendix C. A significance level of 0.05 was used. The results

for proportion of tardiness measure are shown at Figure 4.16. Mean values of performance measure are shown in parenthesis. A line is drawn, considering the calculated confidence intervals, under any set of means for which the differences are not statistically significant at $\alpha = 0.05$.

The confidence intervals of multiple comparisons between control policies are obtained as;

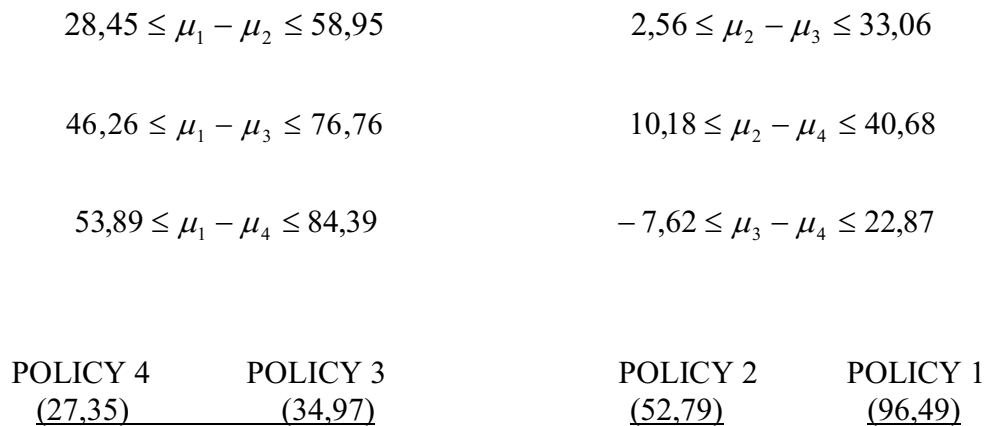


Figure 4.16 Multiple pairwise comparison of control policies for WIP

From the multiple comparison, it can be concluded that setup oriented sequencing rules have significantly the lower proportion of tardiness values than the ordinary sequencing rules. Policy 2 causes significantly less parts late in system than Policy 1. Policy 1 has the worst performance since the parts with high process times should spend much time in system until the parts with smaller process times are processed. As a result, the setup oriented rules provides a less congestion as there are fewer parts waiting in system during the production.

The general results of simulation experiment for WIP objective are shown at Figure 4.17.

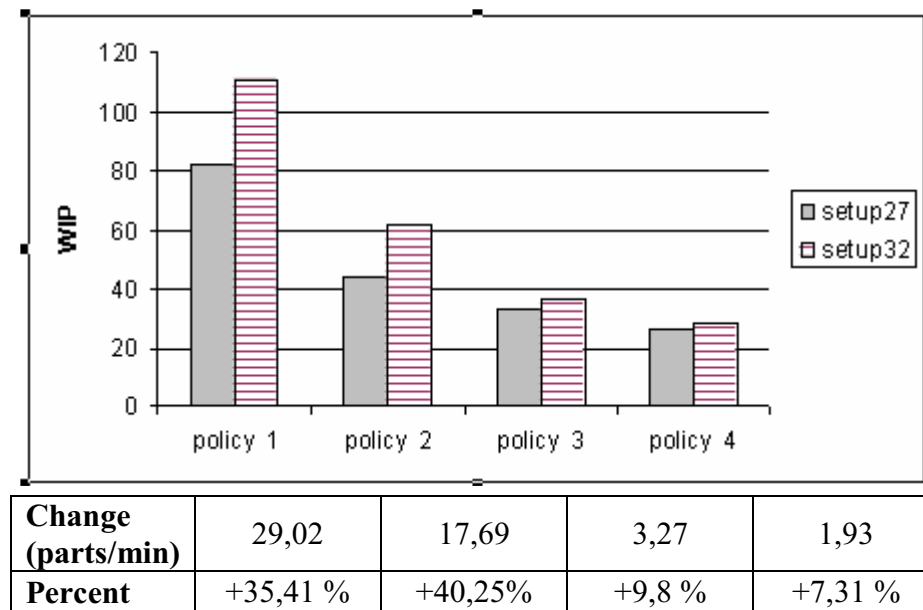


Figure 4.17 Setup time effects over WIP values at different scheduling rules

Higher setup times resulted in higher WIP for all sequencing rules as it was expected. However, the sensitivity against the increase in setup time is much more at ordinary sequencing rules when compared to setup oriented sequencing rules. Setup oriented rules showed a relatively small deterioration as the setup times got higher, this was due to capability to control flow time by reducing setup time. In the investigation for WIP performance measure, setup oriented sequencing rules give much better solutions than the ordinary sequencing rules when the amount of setup time per job increases. Between setup oriented sequencing rules policy 4 reacts less than policy 3 in case of increase of setup time. Policy 2 reacts at most due to increase in setup times because it tries to minimize the lateness of each part. But for the purpose of minimization of lateness, the total numbers of parts - being late - are increased dramatically. This causes the parts to spend more time in system which means that more WIP.

4.7.2. Summary of Statistical Analysis

The results of the main experiment demonstrated that sequence dependent setup time has a significant impact on shop performance. There was a significant difference, operationally and statistically, in performance between setup oriented and

ordinary sequencing rules. The ordinary rules were not able to handle sequence dependent setup times as effectively as the setup oriented rules. Table 4.12 presents a summary of the mean value of the performance measures.

Table 4.12: Summary of performance measures

PERFORMANCE MEASURES		POLICY 1	POLICY 2	POLICY 3	POLICY 4
Flow Time (hr/part)	Setup time (27min)	24,46	24,37	21,44	17,29
	Setup time (32min)	31,71	31,39	23,68	18,69
Average Tardiness (hr/part)	Setup time (27min)	99,96	66,74	17,23	15,73
	Setup time (32min)	131,25	95,28	18,94	16,52
Proportion of Tardiness (%)	Setup time (27min)	6,07	8,20	5,60	4,1
	Setup time (32min)	8,39	11,24	6,74	4,75
WIP (part/min)	Setup time (27min)	81,98	43,94	33,34	26,39
	Setup time (32min)	111,01	61,63	36,61	28,32

It is clear that the most significant impacts of setup time using the ordinary sequencing rules are seen on average tardiness and WIP performance measures. It is natural to observe that the amount of tardiness gets bigger when the long setup times are not taken into consideration. The amount of tardiness has a great effect on customer satisfaction. This result has very important implications for customer service. The ordinary rules may not be able to provide good customer service in terms of delivery when setup time is sequence dependent.

As the parts stay in system more as a result of longer flow times, the WIP is also highly affected by setup times under ordinary sequencing rules. These high rates of WIP may cause problems in production due to congestions in machines and full capacity usage.

Beyond the bad results of ordinary sequencing rules in case of sequence dependent setups, the reactions to increase in setup times are also higher in ordinary sequencing rules than the setup oriented rules. This also proves how well the set up oriented rules may come over in case of high setup times; because the effect of increase in setup times is seem less in setup oriented rules. An increase in setup time causes dramatic increases in all performance measures when ordinary sequencing rules are used. These results also give an impression about the importance of setup times over system's overall performance. That's why it is obvious that sequence dependent setup is a critical element which should be taken seriously under consideration in objective functions during the scheduling activities.

CHAPTER FIVE

CONCLUSION

5.1 Conclusion

The setup operation has for long been considered negligible and hence ignored or considered as a part of the processing time for the case of setup time. While this assumption simplifies the analysis and/or reflects certain applications, it adversely affects the solution quality for many applications which require explicit treatment of setup. Such applications, coupled with the emergence of production concepts like time based competition and group technology, have motivated increasing interest to include setup considerations in scheduling problems. The importance of setup times has been investigated and it was found that sequence dependent setup times were significant for the effective management of manufacturing capacity and to reduce inventory levels and improve customer service.

The studies over sequence dependent setups were mostly focused over single, parallel machines and then flow shop production environments. Job shop hasn't been a popular topic for scheduling studies with sequence dependent setups. It is probably related with the complex nature of job shops which is not easy to find efficient algorithms that are capable of finding optimal solutions in a relatively short period of time.

The subject of this thesis has been inspired from a real problem of an oven producing plant about scheduling activity with sequence dependent setups in a flexible job shop. Flexible job shop is an extension of job shops in which alternative machines are available for the same process. The scheduling problem of a flexible job shop problem (FJSP) consists of a routing sub problem, that is, assigning each operation to a machine in a related set and the scheduling sub problem, which consists of sequencing the assigned operations on each machine in order to obtain a feasible schedule.

The application of problem has been carried out in the press department of the oven producing plant. The plant produces several different kinds of ovens. The press workshop produces the required plain sheets in different shapes and sizes that will be used in the other departments. The press workshop is the heart of the plant as the main required parts -plain sheets- to every department are sent from press workshop. That is why; the workload of press workshop is the heaviest one in the plant.

Totally, there are 18 machines located in six different work centers. To meet the heavy workload, there is more than one for each type of machine at the press workshop. The machines in each work center have the same capability in terms of working speed and machine capabilities. So a part can be processed at any of the available machine in the same period of time in a work center. Besides the main product types there are subtypes of some main types which are different in small details. So they require the same molds for the same processes. But a modification is also needed to use the same mold for different subtypes. Also, there is only one mould for an operation of a part type on a machine. That is why, although there are alternatives for the same type of machines, the subtypes cannot be processed at the same time in the same work center. The job should wait for the end of the other's finish of its process to get the mould. As a result, this mould constraint limits the effective use of machines in the same work center.

When different types of parts are processed on the same machine a change of molds are needed which requires a longer setup time. Since many types of parts are produced in press workshop, setup is a highly time consuming activity. As mentioned before the main purpose of this thesis was to find an efficient way to decrease the time spent for setup activity and to evaluate the difference on performance measures in the situations when the sequence dependent setups were considered or not. Considering this objective, it was clear that the subtypes should be processed consecutively to reduce the time consumption by setups. But it was not also enough to consider the problem as only reducing the setups. Because all orders had due dates that must be met. If only it was desired to reduce the setup times by processing the same subtypes consecutively, the risk of going beyond the desired due dates could occur. For instance, a part ,which had an urgent due date, might not be

processed for a long time as subtypes of another part were processed consecutively although their due dates were less tight. This was a problem between meeting due dates and reducing setup times. The balance should be achieved in a way that while the setup times were reduced as much as available, the due dates should be also met.

The NP-hardness of an optimization problem suggested that it wasn't always possible to find an optimal solution quickly. Especially, by the increase of problem size, an enormous computational effort is needed for an optimal solution. So there should be made a trade off between solution quality and time consumption. As this study was based on the observation of effects of sequence dependent setup times over system performances, it was not desired to reach an optimal solution. So, myopic dispatching rules were used which are ideal for large size problems instead of comprehensive algorithms which were not much usable for large systems as they needed high computational times and the high complexity decreased the efficiency of these algorithms.

For the scheduling problem in job- shop environments under sequence dependent setups, different methods were used. These methods were described in detail at Section 3.1.4. From Table 3.4 it was clear that the studies were mostly done by simulation. Because the studies about job shop scheduling were rare and the existing ones were based on the initial investigation about the effects of sequence dependent setups. Not very comprehensive algorithms were proposed in these studies and simulation was used as a tool to compare the performances of simple dispatching rules. By this thesis, new algorithms were constituted and compared in respect of different performance measures. Simulation was chosen as the evaluation tool like as the past studies did. But in this study, different from the past studies, the algorithms had the ability to give the final sequenced results of jobs between machines which could be used in real life for scheduling.

In Section 4.5 the logic and algorithms of dispatching rules were introduced. The models were created by ARENA software. As simulation was used it was possible to work over a dynamic manufacturing environment in which the orders came randomly and the arrival and processing time values were stochastic. In reality, the production

planning department of the factory gives the exact daily production list of products. So the system is not actually dynamic. But to compare the performances of dispatching rules using simulation, it was better to work on dynamic production environment which gave the chance of longer evaluation with larger volume of production under unexpected conditions.

Under Section 4.6 the steps of simulation study were introduced. Under Section 4.7 the evaluation of proposed dispatching rules were made considering four performance measures flow time, tardiness, proportion of tardiness and WIP. Flow time was chosen as it was a sign of customer satisfaction when considered with WIP. By Anova Tests the main effects and interaction effects of dispatching rules and the setup change were investigated. It was relevant to understand the sources of effects on performance measures. By Tukey Tests it was searched whether the performance of rules are statistically different and which one was the best or the worst. It was important to realize if the difference was statistically significant because the practical difference might be misleading to understand the reality.

The results of the main experiment demonstrated that sequence dependent setup time had a significant impact on shop performance. There was a significant difference, operationally and statistically, in performance between setup oriented and ordinary sequencing rules. The ordinary rules were not able to handle sequence dependent setup times as effectively as the setup oriented rules.

It was clear that the most significant impacts of setup time using the ordinary sequencing rules are seen on average tardiness and WIP performance measures. It was natural to observe that the amount of tardiness got bigger when the long setup times were not taken into consideration. The amount of tardiness had a great effect on customer satisfaction so this result had very important implications for customer service. The ordinary rules might not be able to provide good customer service in terms of delivery when setup time was sequence dependent.

As the parts stayed in system longer as a result of longer flow times, the WIP was also highly affected by setup times under ordinary sequencing rules. These high rates

of WIP may cause problems in production due to congestions in machines and full capacity usage.

Beyond the bad results of ordinary sequencing rules in case of sequence dependent setups, the reactions to increase in setup times were also higher in ordinary sequencing rules than the setup oriented rules. This also proved how well the set up oriented rules might come over in case of high setup times; because the effect of increase in setup times was seem less in setup oriented rules. An increase in setup time caused dramatic increases in all performance measures when ordinary sequencing rules were used. These results also gave an impression about the importance of setup times over system's overall performance. That's why it was obvious that sequence dependent setup was a critical element which should be taken seriously under consideration in objective functions during the scheduling activities.

5.2. Further Research

This study focused on the impact of sequence dependent setups in scheduling of parts in a flexible job shop environment. Different rules were evaluated and the performances were compared in respect of different response variables. But all of the proposed dispatching rules were based on objective functions which considered single response variable which couldn't be given different importance levels in case of different situations. In reality, the scheduling process is a combination of many response variables which may conflict between each other. For example, an increase in the level of Work-in Process in the shop leads more possibilities to optimize the job sequence in the shop (longer queues). But increasing WIP may lead to an increase on flow time cause of congestions at machines. This implies the necessity of the evaluation and comparison of the importance of response variables between each other. This diversification can be achieved by assigning a weight to each response variable. That's why, further research can be done to constitute a multi objective function which includes different decision factors multiplied with relative weights. At different times of production, different response variables may be more crucial than others. By changing weights in that manner, the scheduling procedure can be re-arranged according to system's requirements.

As the study was based on the comparison of sequencing rules, it was done under some assumptions to simplify the production environment. But in real life there are lots of random events which may affect the scheduling process like machine failures, the availability of operators, the lack of opportunity of working with fully cross trained operators which may affect the processing and setup times and finite input queue capacities. Another further research can be done to enhance the dispatching rules by adding the additional constraints by using the advantage of simulation.

REFERENCES

- Artigues C., & Roubellat F.(2002) An efficient algorithm for operation insertion in a multi-resource job shop Schedule with sequence dependent setup times. *Production Planning Control*, Vol.13, No.2, 175-186
- Baker, K.R. (1974) *Introduction to sequencing and scheduling*. New York: Wiley and Sons.
- Ballicu M., Giua A., & Scatzu C.(2002) *Jop shop scheduling models with setup times*. Dip. Ingegneria Elettrica ed Elettronica, Universita di Cagliari, Italy
- Banks, J., Carson. J.S., Nelson, B.L., Nicol, D.M., (2000) *Discrete-Event system simulation* (Third Edition). New Jersey: Prentice-Hall
- Barnes, J.W., & Vanston, L.K. (1981) Scheduling jobs with linear delay penalties and sequence dependent setup costs. *Operation Research*, 29, 146-160
- Bianco, L., Ricciardelli, S., Rinaldi, G.,& Sassano, A. (1988). Scheduling tasks with sequence dependent processing times. *Naval Research Logistics*, 35, 177-184
- Bitran, G.R., & Gilbert S.M. (1990). Sequencing production on parallel machines with two magnitudes of sequence dependent setup cost. *Journal of Manufacturing and Operations Management*, 3, 24-52
- Brandimarte, P.,&Villa A. (1995). *Advanced models for manufacturing systems management*, Florida: CRC Press, Inc
- Burstable, R.M., (1966). A heuristic method for a job shop scheduling problem. *Operational Research Quarterly*, 17, 291-304
- Candido, M.A.B., Khator, S.K., & Barcia, R.M. (1998) A genetic algorithm based procedure for more realistic job shop scheduling problems. *Int . J. Prod. Res.*, Vol:36, No. 12, 3437-3457

- Charles, H. White., & Richard C. Wilson. (1977) Sequence dependent setup times and job sequencing. *Int. J. Prod. Res.*, Vol. 15, No:2, 191-202
- Choi, I-C., & Korkmaz, O. (1997). Job shop scheduling with separable sequence dependent setups. *Annals of Operations Research*, 70, 155-170
- Corwin, B.D., & Esogbou A.O. (1974). Two- machine flowshop scheduling problems with sequence dependent setup times: a dynamic programming approach. *Naval Research Logistics Quarterly*, 21, 51:
- Deane, R.H., & White, E.R. (1975). Balancing workloads and minimizing setup costs in parallel processing shop. *Operational Research Quarterly*, 26, 45-53
- Dearing, P.M., & Henderson, R.A. (1984). Assigning looms in a textile weaving operation with changeover limitations. *Production and Inventory Management*, 25, 23-31
- Farn, C.D., & Muhleman, A.P. (1979). The dynamic aspects of a production scheduling problem. *International Journal of Production Research*, 17, 15-21
- Flynn, BB. (1987). The effects of setup time on output capacity in cellular manufacturing. *Int J Prod Res*, 25, 1761±72.
- Franca, P.M., Gendreau, M., Laporte, G., & Muller, F.M. (1996). A tabu search heuristic for the multiprocessor scheduling problem with sequence dependent setup times. *International Journal of Production Economics*, 43, 79-89
- Ghosh, JB. (1994). Batch scheduling to minimize total completion time. *Oper. Res. Lett*, 16:271±5.
- Guinet., A., & Dussauchoy, A. (1993). Scheduling sequence dependent jobs on identical parallel machines to minimize completion time criteria. *International Journal of Production Research*, 31, 1579-1594

- Gupta, S.K. (1982). n jobs and m machines jobshop problems with sequence dependent setup times. *International Journal of Production Research*, 20, 643-656.
- Gupta, J.N.D. (1986). Flowshop schedules with sequence dependent setup times. *Journal of the Operations Research Society of Japan*, 29, 206-219
- Gupta, J.N.D.,& Darrow, W.P. (1986). The two-machine sequence dependent flowshop scheduling problem. *European Journal of Operational Research*, 24, 439-446
- Gupta, M.C., Gupta, Y.P.,& Evans, G.W. (1993). Operations planning and scheduling problems in advanced manufacturing systems. *International Journal of Production Research*, 31, 4, 869-900
- Hershauer, J.C. (1970). *An empirically derived simulation to explore sequencing decisions*. Unpublished PhD Thesis, Indiana University.
- In Chan C.,& Dae-Sik C. (2002). A Local search algorithm for job shop scheduling problems with alternative operations and sequence dependent setups. *Computers Industrial Engineering*, 42, 43-58
- Jones, A.,& Rabelo, L.C. (1998). *Survey of Job Shop Scheduling Techniques*.
- Kang, S., Malik, K.,& Thomas, L.J. (1999). Lotsizing and scheduling on parallel machines with sequence dependent setup costs. *Management Science*, 45, 273-289
- Kim, S.C.,& Bobrowski, P.M. (1994). Impact of sequence-dependent setup times on job shop scheduling performance. *Int. J. Prod. Res.*, 32, 1503±20.
- Kim, S.C.,& Bobrowski, P.M. (1997). Scheduling jobs with uncertain setup times and sequence dependency. Omega. *Int. J. Mgmt. Sci.*, 25, 4, 437-447
- King, J.R., & Spachis, A. S. (1980). *Scheduling: Bibliography & Review*. *International Journal of Physical Distribution and Materials Management*, Vol. 10. West Yorkshire, England: MCB Publications Limited.

- Kolahan, F., Liang, M., & Zuo, M. (1995). Solving the combined part sequencing and tool replacement problem for an automated machine center: A Tabu search approach. *Comput. Ind. Eng.*, 28:731±43.
- Krajewski, L.J., King, B.E., Ritzman, L.P., & Wong, D.S. (1987). Kanban, MRP, and shaping the manufacturing environment. *Manage. Sci.*, 33:39±57.
- Lee, Y.H., Bhaskaran, K., & Pinedo, M. (1997). A heuristic to minimize total weighted tardiness with sequence dependent setups. *IEEE Transactions*, 29, 45-52
- Lee, Y.H., & Pinedo, M. (1997). Scheduling jobs on parallel machines with sequence dependent setups. *European Journal of Operational Research*, 100, 464-474
- Lockett, A.G., & Muhlemann, A.P. (April 1971). *A scheduling problem involving sequence dependent changeover times*. Technical Notes, University of Manchester, UK.
- Low, C.Y. (1995). Job shop scheduling heuristics for sequence- dependent setups. *Comput. Ind. Eng.*, 29:279±83.
- Marsh, J.D., & Montgomery, D.C. (1973). Optimal procedures for scheduling jobs with sequence-dependent changeover times on parallel processors. *AIIE Technical Papers*, 279-286
- McLean, C. & Leong, S. (2001). The expanding role of simulation in future manufacturing. *Proceedings of the 2001 Winter Simulation Conference*, ed. B. A. Peters, J. S. Smith, D. J. Medeiros, and M. W. Rohrer.
- McNally, P. & Heavey, C. (2002) *Simulation as a desktop resource: the limitations and future requirements of simulation technologies*. Proceedings of the Nineteenth Conference of the Irish Manufacturing Committee. Queens University, Belfast.
- Miller, D.M., Chen, H.C., Matson, J., & Liu, Q. (1999). A hybrid genetic algorithm for a single machine scheduling problem. *Journal of Heuristics*, 5, 437-454

- Monma, C.L. & Potts, C.N. (1989). On the complexity of scheduling with batch setups. *Operations Research*, 37, 798-804
- O'Grady, P.J., & Harrison, C. (1988). Search based job scheduling and sequencing with setup times, *Omega International Journal of Management Science*, 16, 547-552.
- Ovacik, I.M., & Uzsoy, R. (1994). Exploiting shop floor status information to Schedule complex job shops, *Journal of Manufacturing Systems*, 13, 73-84
- Ovacik, I.M., & Uzsoy, R. (1993). Worst-case error bounds for parallel machine problems with bounded sequence dependent setup times. *Operations Research Letters*, 14, 251-256
- Ovacik, I.M., & Uzsoy, R. (1995). Rolling horizon procedures for dynamic parallel machine scheduling with sequence dependent setup times. *International Journal of Production Research*, 33, 3173-3192
- Parthasarathy, S., & Rajendran, C. (1997). A simulated annealing heuristic for scheduling to minimize weighted tardiness in a flowshop with sequence dependent setup times of jobs-a case study. *Production planning and Control*, 8, 475-483
- Parunak. (1990). *Characterizing Manufacturing Scheduling Problem*.
- Pegden, C.D., Shannon, R.E., & Sadowski R.P. (1990). *Introduction to simulation using SIMAN* (2nd Ed), New York: McGraw-Hill
- Pinedo, M. (1995). *Scheduling, Theory, Algorithms, and Systems*. New Jersey: Prentice- Hall.
- Pinedo, M., & Chao, X. (1999). *Operations scheduling with applications in manufacturing and services*. Singapore: McGraw-Hill
- Raman, N., Rachamadugu, R.V., & Talbot, F.B. (1989). Real-time scheduling on an automated machine center. *Eur J Oper Res*, 40:222 ± 42.

- Rembold, U., Nnaji, B.O., & Storr A. (1993). *Computer Integrated Manufacturing and Engineering*, Addison-Wesley
- Rios-Mercado, R.Z., & Bard, F.B. (1998). Heuristics for the flow line problem with setup costs. *European Journal of Operational Research*, 110, 76-98
- Scott, J. Mason, John W. Fowler, & W. Matthew Carlyle. (2002). A modified shifting bottleneck heuristic for minimizing total weighted tardiness in complex job shops. *Journal of Scheduling*, 5:247-262
- Sotokov, Y.N., Tautenhahn, T., & Werner, F. (1998). On the application of insertion techniques for job shop problems with batch setup times. *RAIRO Oper Res.*
- Srikar, B.N., & Ghosh, S. (1986). A MILP model for the n-job, M-stage flowshop with sequence dependent setup times. *International Journal of Production Research*, 24, 1459-1472
- Sun, X.Q., & Noble, J.S. (1999). An approach to jobshop scheduling with sequence dependent setups. *Journal of manufacturing Systems*, 18, 416-430.
- Szwark, W., & Gupta, J.W.D. (1987). A flowshop problem with sequence dependent setup times. *Naval Research Logistics Quarterly*, 34, 619-627
- Taha, H.A (1997). *Operations Research an Introduction*, 6th Ed., New Jersey: Prentice-Hall,.
- Taner, M.R. (2001). *Scheduling with Sequence Dependent Setup Times*. Phd Thesis, North Carolina State University. Raleigh
- Uskup, G., & Smith, S.B. (1975). A branch and bound for two stage production sequencing. *Operations Research*, 23, 118-136
- Uzsoy, R., Martin-Vega, L.A., Lee, C.Y., & Leonard, P.A. (1991). Production scheduling algorithms for a semiconductor test facility. *IEEE Transactions Semiconductor Manufacturing*, 4, 270-280

- Uzsoy, R., Lee, C.Y. & Martin-Vega, L.A. (1992). Scheduling semiconductor test operations: Minimizing maximum lateness and number of tardy jobs on a single machine. *Naval Research Logistics*, 39, 369-388
- Wilbrecht, J.K., & Prescott, W.B. (1969). The influence of setup time on job shop performance. *Manage Sci.*, 16:B274± B280.
- Wortman, D.B. (1992). Managing capacity: Getting the most from your firm's assets. *Ind. Eng.*, 24:47±9.
- Wu, Szu-Yung, D. (1989). Artificial Intelligence and Scheduling Applications. Artificial Intelligence. Manufacturing Theory and Practice (S. T. Kumars et al,) *Institute of Industrial Engineers*.
- Zhou, C., & Egbelu, P.J. (1989). Scheduling in a manufacturing shop with sequence dependent setups. *Robotics and Computer Integrated Manufacturing*, 5, 73-81

APPENDIX A : Input Analysis of Batch Sizes

Table A.1: Historical data of batch sizes

258	220	455	286	377	597	244	426	225	555	434	361
554	549	518	597	237	222	469	219	366	553	484	378
270	450	448	356	264	310	525	404	550	592	521	565
273	542	275	396	218	359	531	326	579	254	384	447
444	504	331	352	442	495	586	332	541	289	585	439
418	206	216	265	333	520	327	574	399	356	546	396
532	359	205	480	570	504	393	587	220	241	384	225
581	229	302	593	397	451	316	245	229	297	326	223
448	281	380	361	369	435	411	291	428	515	375	294
214	480	557	567	329	518	418	559	306	257	295	464
599	217	409	571	253	573	557	555	526	242	378	537
547	540	388	538	361	433	464	411	419	424	565	308
375	484	218	414	299	578	495	351	303	251	535	350
505	512	481	439	419	531	444	377	309	540	584	231
300	448	217	336	513	501	207	310	438	345	351	294
339	420	444	388	390	378	582	534	400	326	446	246
434	425	510	548	237	351	368	521	215	204	563	473
274	461	343	449	458	280	256	487	289	379	207	271
309	316	218	566	463	414	543	511	210	322	410	530
464	263	312	541	377	287	544	240	412	423	599	352
590	333	246	295	510	215	314	486	483	287	512	384
252	429	223	462	596	315	362	552	343	341	471	477
439	386	575	493	432	285	532	525	292	203	273	475
318	357	277	593	268	439	576	410	569	327	227	218
285	255	414	440	451	296	367	289	507	354	458	525
248	218	385	504	403	443	488	378	576	481	317	299
435	583	556	372	518	389	269	408	393	212	227	324
298	217	460	455	578	548	550	268	496	517	559	269
472	436	355	540	502	341	599	527	400	358	497	388
338	324	528	237	270	368	423	239	298	390	378	561
495	214	392	360	464	323	421	440	469	582	276	500
587	282	599	277	435	519	454	436	443	310	560	583
226	465	251	227	344	268	246	493	328	431	504	595
496	486	258	536	329	305	395	327	536	497	272	451
564	512	427	492	463	356	534	295	201	478	367	278
381	556	483	479	385	563	480	445	473	440	224	335
258	356	293	223	457	215	297	552	376	569	593	299
473	481	540	239	375	428	260	203	362	500	466	269
265	337	304	404	366	413	378	511	499	201	341	394
324	338	430	408	417	574	483	357	473	573	495	390
366	414	463	418	266	554	552	262	527	382	264	511
373	536	248	315	252	470	245	455				

Input analysis of Batch Size

Distribution: Uniform

Expression: UNIF(201, 599)

Square Error: 0.000457

Chi Square Test results

Number of intervals = 22

Degrees of freedom = 21

Test Statistic = 5.03

Corresponding p-value > 0.75

Kolmogorov-Smirnov Test results

Test Statistic = 0.0269

Corresponding p-value > 0.15

Data Summary

Number of Data Points = 500

Min Data Value = 200

Max Data Value = 600

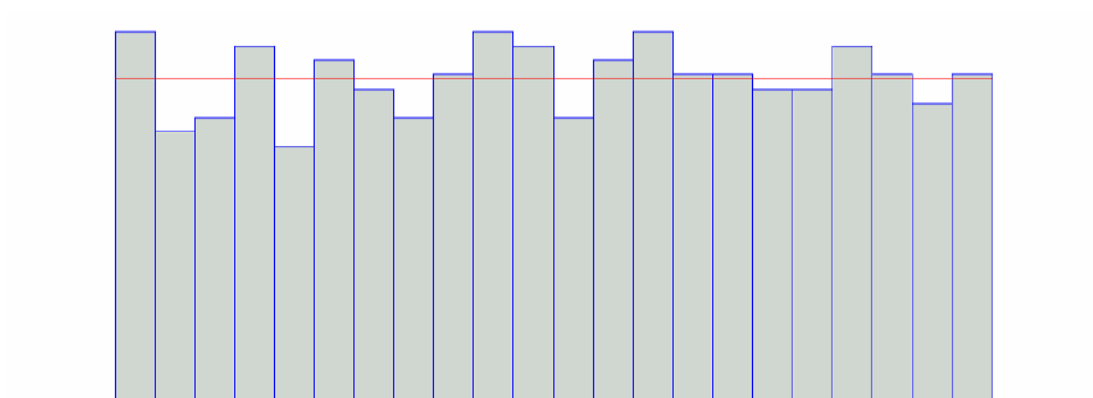
Sample Mean = 402

Sample Std Dev = 114

The square errors are calculated for each distribution and uniform distribution has the least square error among all.

Table A.2: Square Errors of different distributions considering the batch size data

Function	Sq Error
Uniform	0.000457
Beta	0.00049
Normal	0.00589
Weibull	0.00663
Gamma	0.00821
Erlang	0.00847
Exponential	0.0124
Lognormal	0.0153
Triangular	0.0158

**Figure A: Histogram of uniform distribution**

APPENDIX B : Part Types, Routes and Mean Process Times

Table B: Part types, routes and mean process times

Part type	Route	Processing Times (minutes/unit)
Part 1	Station 6	0.3
	Station 2	0.28
	Station 5	0.25
	Station 4	0.27
Part 2	Station 5	0.25
	Station 6	0.22
	Station 1	0.2
	Station 2	0.19
Part 3	Station 5	0.31
	Station 2	0.3
	Station 1	0.29
	Station 4	0.26
Part 4	Station 1	0.27
	Station 2	0.24
	Station 5	0.25
	Station 6	0.25
Part 5	Station 3	0.27
	Station 2	0.3
	Station 6	0.24
	Station 5	0.22
Part 6	Station 1	0.2
	Station 2	0.25
	Station 6	0.26
	Station 5	0.25
Part 7	Station 1	0.29
	Station 6	0.24
	Station 5	0.26
	Station 4	0.27
Part 8	Station 1	0.26
	Station 2	0.29
	Station 3	0.15
	Station 6	0.16
Part 9	Station 4	0.17
	Station 3	0.15
	Station 6	0.16
	Station 5	0.15
Part 10	Station 5	0.14
	Station 1	0.18
	Station 3	0.15
	Station 6	0.14

Part 11	Station 3	0.15
	Station 5	0.16
	Station 4	0.12
	Station 6	0.15
Part 12	Station 5	0.14
	Station 1	0.16
	Station 2	0.14
	Station 6	0.13
Part 13	Station 3	0.15
	Station 6	0.16
	Station 2	0.15
	Station 5	0.1
Part 14	Station 4	0.1
	Station 6	0.09
	Station 5	0.17
	Station 1	0.14
Part 15	Station 5	0.17
	Station 1	0.16
	Station 6	0.18
	Station 2	0.17

APPENDIX C: Statistical Analysis of Dispatching Rules for Each Performance Measure

Table C.1: Tukey test for the evaluation of policies for statistically difference considering flow time performance measure

Multiple Comparisons

Dependent Variable: FLOWTIME

			Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
(I) POLICY (J)POLICY							
Tukey HSD	1,00	2,00	,2027	1,4574	,999	-3,5413	3,9468
		3,00	5,524*	1,4574	,001	1,7804	9,2686
		4,00	10,094*	1,4574	,000	6,3502	13,8383
	2,00	1,00	-,2027	1,4574	,999	-3,9468	3,5413
		3,00	5,3218*	1,4574	,001	1,5777	9,0658
		4,00	9,8915*	1,4574	,000	6,1474	13,6356
	3,00	1,00	-5,5245*	1,4574	,001	-9,2686	-1,7804
		2,00	-5,3218*	1,4574	,001	-9,0658	-1,5777
		4,00	4,5697*	1,4574	,009	,8257	8,3138
	4,00	1,00	-10,0942*	1,4574	,000	-13,838	-6,3502
		2,00	-9,8915*	1,4574	,000	-13,635	-6,1474
		3,00	-4,5697*	1,4574	,009	-8,3138	-,8257

Based on observed means.

* The mean difference is significant at the ,05 level.

Table C.2: Tukey test for flow time

POLICY		N	Subset		
			1	2	3
Tukey	4,00	40	17,9945		
	3,00	40		22,5642	
	2,00	40			27,8860
	1,00	40			28,0888

Means for groups in homogeneous subsets are displayed. Based on Type III Sum of Squares The error term is Mean Square(Error) = 42,480. Alpha = ,05.

Table C.3: Tukey test for the evaluation of policies for statistically difference considering tardiness performance measure

Multiple Comparisons

Dependent Variable: TARDINESS

(I)POLICY (J)POLICY			Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Tukey HSD	1,00	2,00	34,5940*	8,7249	,000	12,1795	57,0085
		3,00	97,5202*	8,7249	,000	75,1058	119,9347
		4,00	99,4775*	8,7249	,000	77,0630	121,8920
	2,00	1,00	-34,5940*	8,7249	,000	-57,0085	-12,1795
		3,00	62,9263*	8,7249	,000	40,5118	85,3407
		4,00	64,8835*	8,7249	,000	42,4690	87,2980
	3,00	1,00	-97,5202*	8,7249	,000	-119,934	-75,1058
		2,00	-62,9263*	8,7249	,000	-85,3407	-40,5118
		4,00	1,9572	8,7249	,996	-20,4572	24,3717
	4,00	1,00	-99,4775*	8,7249	,000	-121,892	-77,0630
		2,00	-64,8835*	8,7249	,000	-87,2980	-42,4690
		3,00	-1,9572	8,7249	,996	-24,3717	20,4572

Based on observed means.

* The mean difference is significant at the ,05 level.

Table C.4: Tukey test for tardiness

POLICY		N	Subset		
			1	2	3
Tukey B	4,00	40	16,1315		
	3,00	40	18,0887		
	2,00	40		81,0150	
	1,00	40			115,6090

Means for groups in homogeneous subsets are displayed. Based on Type III Sum of Squares The error term is Mean Square(Error) = 1522,463. Alpha = ,05.

Table C.5: Tukey test for the evaluation of policies for statistically difference considering proportion of tardiness performance measure

Multiple Comparisons

Dependent Variable: PROPORTION OF TARDINESS

(I) POLICY (J) POLICY			Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Tukey HSD	1,00	2,00	-2,4962*	,7302	,004	-4,3720	-,6205
		3,00	1,0575	,7302	,469	-,8183	2,9333
		4,00	2,8243*	,7302	,001	,9485	4,7000
	2,00	1,00	2,4962*	,7302	,004	,6205	4,3720
		3,00	3,5538*	,7302	,000	1,6780	5,4295
		4,00	5,3205*	,7302	,000	3,4447	7,1963
	3,00	1,00	-1,0575	,7302	,469	-2,9333	,8183
		2,00	-3,5538*	,7302	,000	-5,4295	-1,6780
		4,00	1,7668	,7302	,073	-,1090	3,6425
	4,00	1,00	-2,8243*	,7302	,001	-4,7000	-,9485
		2,00	-5,3205*	,7302	,000	-7,1963	-3,4447
		3,00	-1,7668	,7302	,073	-3,6425	,1090

Based on observed means.

* The mean difference is significant at the ,05 level.

Table C.6: Tukey test for proportion of tardiness

POLICY		N	Subset		
			1	2	3
Tukey B	4,00	40	4,4082		
	3,00	40		6,1750	
	1,00	40		7,2325	
	2,00	40			9,7288

Means for groups in homogeneous subsets are displayed. Based on Type III Sum of Squares The error term is Mean Square(Error) = 1522,463. Alpha = ,05.

Table C.7: Tukey test for the evaluation of policies for statistically difference considering WIP performance measure

Multiple Comparisons
Dependent Variable: WIP

(I) POLICY (J) POLICY			Mean Difference (I-J)	Std. Error	Sig.	95% Confidence Interval	
						Lower Bound	Upper Bound
Tukey HSD	1,00	2,00	43,7045*	5,9361	,000	28,4545	58,9545
		3,00	61,5175*	5,9361	,000	46,2675	76,7675
		4,00	69,1401*	5,9361	,000	53,8901	84,3901
	2,00	1,00	-43,7045*	5,9361	,000	-58,9545	-28,454
		3,00	17,8130*	5,9361	,014	2,5630	33,0630
		4,00	25,4356*	5,9361	,000	10,1855	40,6856
	3,00	1,00	-61,5175*	5,9361	,000	-76,7675	-46,267
		2,00	-17,8130*	5,9361	,014	-33,0630	-2,5630
		4,00	7,6225	5,9361	,573	-7,6275	22,8726
	4,00	1,00	-69,1401*	5,9361	,000	-84,3901	-53,890
		2,00	-25,4356*	5,9361	,000	-40,6856	-10,185
		3,00	-7,6225	5,9361	,573	-22,8726	7,6275

Based on observed means.

* The mean difference is significant at the ,05 level.

Table C.8: Tukey test for WIP

POLICY		N	Subset		
			1	2	3
Tukey B	4,00	40	27,3558		
	3,00	40	34,9784		
	2,00	40		52,7914	
	1,00	40			96,4959

Means for groups in homogeneous subsets are displayed. Based on Type III Sum of Squares The error term is Mean Square(Error) = 704,744. Alpha = ,05.

APPENDIX D : SIMAN Codes of Policy 1

POLICY 1

```

57$      CREATE,      1,0:expo(37,seed1);
58$      ASSIGN:      part=

disc(0.066,1,0.132,2,0.198,3,0.204,4,0.270,5,0.336,6,0.402,7,0.468,8,0.534,9,0.6,10,0.666,11,0.732,1
2,0.798,13,0.864,14,1,15):
        amount=unif(200,600,seed2):
        NS=part:
        picture=part:
        entersystem=TNOW:
        product=type:
        type=type+1;
231$     BRANCH,:     If,part==1,at1,Yes:
        If,part==2,at2,Yes:
        If,part==3,at3,Yes:
        If,part==4,at4,Yes:
        If,part==5,at5,Yes:
        If,part==6,at6,Yes:
        If,part==7,at7,Yes:
        If,part==8,at8,Yes:
        If,part==9,at9,Yes:
        If,part==10,at10,Yes:
        If,part==11,at11,Yes:
        If,part==12,at12,Yes:
        If,part==13,at13,Yes:
        If,part==14,at14,Yes:
        If,part==15,at15,Yes;
at1      ASSIGN:      totalprocesstime=amount*(r(part,2)+r(part,5)+r(part,6)+r(part,4));
232$     BRANCH,:     If,TNOW>=coefficient*480,233$,Yes:
        If,TNOW<coefficient*480,234$,Yes;
233$     ASSIGN:      coefficient=coefficient+1:
        nextshift=coefficient*480:
        duedate=nextshift+4.7*(totalprocesstime+4*27);
26$      ROUTE:      0.0,SEQ;

234$     ASSIGN:      nextshift=coefficient*480:
        duedate=nextshift+4.7*(totalprocesstime+4*27):NEXT(26$);

at2      ASSIGN:
totalprocesstime=amount*(r(part,2)+r(part,6)+r(part,1)+r(part,5)):NEXT(232$);

at3      ASSIGN:
totalprocesstime=amount*(r(part,5)+r(part,2)+r(part,4)+r(part,1)):NEXT(232$);

at4      ASSIGN:
totalprocesstime=amount*(r(part,1)+r(part,2)+r(part,5)+r(part,6)):NEXT(232$);

at5      ASSIGN:
totalprocesstime=amount*(r(part,2)+r(part,3)+r(part,5)+r(part,6)):NEXT(232$);

```

```

at6      ASSIGN:
totalprocesstime=amount*(r(part,1)+r(part,2)+r(part,6)+r(part,5)):NEXT(232$);

at7      ASSIGN:
totalprocesstime=amount*(r(part,1)+r(part,4)+r(part,5)+r(part,6)):NEXT(232$);

at8      ASSIGN:
totalprocesstime=amount*(r(part,2)+r(part,1)+r(part,6)):NEXT(232$);

at9      ASSIGN:
totalprocesstime=amount*(r(part,4)+r(part,3)+r(part,6)+r(part,5)):NEXT(232$);

at10     ASSIGN:
totalprocesstime=amount*(r(part,5)+r(part,1)+r(part,3)+r(part,6)):NEXT(232$);

at11     ASSIGN:
totalprocesstime=amount*(r(part,5)+r(part,4)+r(part,3)+r(part,6)):NEXT(232$);

at12     ASSIGN:
totalprocesstime=amount*(r(part,5)+r(part,1)+r(part,2)+r(part,6)):NEXT(232$);

at13     ASSIGN:
totalprocesstime=amount*(r(part,5)+r(part,2)+r(part,3)+r(part,6)):NEXT(232$);

at14     ASSIGN:
totalprocesstime=amount*(r(part,5)+r(part,1)+r(part,4)+r(part,6)):NEXT(232$);

at15     ASSIGN:
totalprocesstime=amount*(r(part,5)+r(part,1)+r(part,2)+r(part,6)):NEXT(232$);

24$      STATION,    s1;
55$      ASSIGN:    ara=part;
0$       BRANCH,,:  If,aaa==1,275$,Yes:
                If,aaa>1,1$,Yes;
275$     ASSIGN:    time11(aaa)=TNOW;
2$       BRANCH,,:  If,aab==1,277$,Yes:
                If,aab>1,3$,Yes;
277$     ASSIGN:    time12(aab)=TNOW;
61$     BRANCH,,:  If,aac==1,276$,Yes:
                If,aac>1,62$,Yes;
276$     ASSIGN:    time13(aac)=TNOW;
4$       BRANCH,,:  If,part==res1,b1,Yes:
                If,part==res2,b2,Yes:
                If,part==res3,b3,Yes:
                If,part<>res1.and.part<>res2.and.part<>res3,5$,Yes;
b1       BRANCH,,:  If,aaa==1,278$,Yes:
                If,aaa>1,279$,Yes;
278$     ASSIGN:    processtime=amount*r(part,1):
                top1=top1+processtime+27:
                aaa=aaa+1;
10$     QUEUE,      q1;
11$     SEIZE,      1:
                r1,1:
                operation1,1;
13$     BRANCH,,:  If,res1==part,12$,Yes:
                If,res1<>part,14$,Yes;
12$     ASSIGN:    preparation=5:
                res1=part:

```

```

top1=top1-22;
15$  DELAY:    processtime;
16$  RELEASE:   r1,1;
367$  WRITE,    fileofres1,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;
329$  BRANCH,:   If,NQ(q1)==0.and.STATE(r1)==IDLE_RES,330$,Yes:
        If,NQ(q1)<>0.or.STATE(r1)      '_RES,235$,Yes;
330$  ASSIGN:    aaa=1:
        top1=0;
235$  DELAY:    preparation;
257$  RELEASE:   operation1,1;
25$   ROUTE:    0.0,SEQ;

14$   ASSIGN:    preparation=27:
        res1=part: NEXT(15$);

279$   ASSIGN:    processtime=amount*r(part,1):
        top1=top1+processtime+27:NEXT(10$);

b2     BRANCH,:   If,aab==1,280$,Yes:
        If,aab>1,281$,Yes;
280$   ASSIGN:    processtime=amount*r(part,1):
        top2=top2+processtime+27:
        aab=aab+1;
17$   QUEUE,     q2;
18$   SEIZE,      1:
        r2,1:
        operation2,1;
20$   BRANCH,:   If,res2==part,19$,Yes:
        If,res2<>part,21$,Yes;
19$   ASSIGN:    preparation=5:
        res2=part:
        top2=top2-22;
22$   DELAY:    processtime;
23$   RELEASE:   r2,1;
368$   WRITE,    fileofres2,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;
331$   BRANCH,:   If,NQ(q2)==0.and.STATE(r2)==IDLE_RES,332$,Yes:
        If,NQ(q2)<>0.or.STATE(r2)==BUSY_RES,236$,Yes;
332$   ASSIGN:    aab=1:
        top2=0;
236$   DELAY:    preparation;
258$   RELEASE:   operation2,1:NEXT(25$);

21$   ASSIGN:    preparation=27:
        res2=part: NEXT(22$);

281$   ASSIGN:    processtime=amount*r(part,1):
        top2=top2+processtime+27:NEXT(17$);

b3     BRANCH,:   If,aac==1,282$,Yes:
        If,aac>1,283$,Yes;
282$   ASSIGN:    processtime=amount*r(part,1):
        top3=top3+processtime+27:
        aac=aac+1;
65$   QUEUE,     q3;
66$   SEIZE,      1:
        r3,1:
        operation3,1;
68$   BRANCH,:   If,res3==part,67$,Yes:

```

```

        If,res3<>part,69$,Yes;
67$  ASSIGN:    preparation=5:
        res3=part:
        top3=top3-22;
70$  DELAY:    processtime;
71$  RELEASE:   r3,1;
369$  WRITE,    fileofres3,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;
333$  BRANCH,:   If,NQ(q3)==0.and.STATE(r3)==IDLE_RES,334$,Yes:
        If,NQ(q3)<>0.or.STATE(r3)==BUSY_RES,237$,Yes;
334$  ASSIGN:    aac=1:
        top3=0;
237$  DELAY:    preparation;
259$  RELEASE:   operation3,1:NEXT(25$);

69$  ASSIGN:    preparation=27:
        res3=part:NEXT(70$);

283$  ASSIGN:    processtime=amount*r(part,1):
        top3=top3+processtime+27:NEXT(65$);

5$    SEARCH,   q1,1,NQ(q1):part==ara;
6$    BRANCH,:   If,J>0,b1,Yes:
        If,J==0,8$,Yes;
8$    SEARCH,   q2,1,NQ(q2):part==ara;
7$    BRANCH,:   If,J>0,b2,Yes:
        If,J==0,64$,Yes;
64$   SEARCH,   q3,1,NQ(q3):part==ara;
63$   BRANCH,:   If,J>0,b3,Yes:
        If,J==0,9$,Yes;
9$    BRANCH,    1:
        If,top1==MN(top1,top2,top3),b1,Yes:
        If,top2==MN(top1,top2,top3),b2,Yes:
        If,top3==MN(top1,top2,top3),b3,Yes;
62$   ASSIGN:    time13(aac)=TNOW:
        top3=top3-time13(aac)+time13(aac-1):
        aac=aac+1:NEXT(4$);

3$    ASSIGN:    time12(aab)=TNOW:
        top2=top2-time12(aab)+time12(aab-1):
        aab=aab+1:NEXT(61$);

1$    ASSIGN:    time11(aaa)=TNOW:
        top1=top1-time11(aaa)+time11(aaa-1):
        aaa=aaa+1:NEXT(2$);

51$   STATION,   s2;
56$   ASSIGN:    ara=part;
27$   BRANCH,:   If,bba==1,284$,Yes:
        If,bba>1,28$,Yes;
284$   ASSIGN:    time21(bba)=TNOW;
29$   BRANCH,:   If,bbb==1,286$,Yes:
        If,bbb>1,30$,Yes;
286$   ASSIGN:    time22(bbb)=TNOW;
72$   BRANCH,:   If,bbc==1,285$,Yes:
        If,bbc>1,73$,Yes;
285$   ASSIGN:    time23(bbc)=TNOW;
31$   BRANCH,:   If,part==res4,b4,Yes:

```

```

        If,part==res5,b5,Yes:
        If,part==res6,b6,Yes:
        If,part<res4.and.part<res5.and.part<res6,32$,Yes;
b4    BRANCH,:    If,bba==1,287$,Yes:
        If,bba>1,288$,Yes;
287$    ASSIGN:    processtime=amount*r(part,2):
        top4=top4+processtime+27:
        bba=bba+1;
37$    QUEUE,      q4;
38$    SEIZE,       1:
        r4,1:
        operation4,1;
40$    BRANCH,:    If,res4==part,39$,Yes:
        If,res4<part,41$,Yes;
39$    ASSIGN:    preparation=5:
        res4=part:
        top4=top4-22;
42$    DELAY:      processtime;
43$    RELEASE:    r4,1;
370$    WRITE,      fileofres4,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
335$    BRANCH,      1:
        If,NQ(q4)==0.and.STATE(r4)==IDLE_RES,336$,Yes:
        If,NQ(q4)<0.or.STATE(r4)==BUSY_RES,240$,Yes;
336$    ASSIGN:    bba=1:
        top4=0;
240$    DELAY:      preparation;
260$    RELEASE:    operation4,1;
52$    ROUTE:      0.0,SEQ;

41$    ASSIGN:    preparation=27:
        res4=part: NEXT(42$);

288$    ASSIGN:    processtime=amount*r(part,2):
        top4=top4+processtime+27:NEXT(37$);

b5    BRANCH,:    If,bbb==1,289$,Yes:
        If,bbb>1,290$,Yes;
289$    ASSIGN:    processtime=amount*r(part,2):
        top5=top5+processtime+27:
        bbb=bbb+1;
44$    QUEUE,      q5;
45$    SEIZE,       1:
        r5,1:
        operation5,1;
47$    BRANCH,:    If,res5==part,46$,Yes:
        If,res5<part,48$,Yes;
46$    ASSIGN:    preparation=5:
        res5=part:
        top5=top5-22;
49$    DELAY:      processtime;
50$    RELEASE:    r5,1;
371$    WRITE,      fileofres5,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
337$    BRANCH,      1:
        If,NQ(q5)==0.and.STATE(r5)==IDLE_RES,338$,Yes:
        If,NQ(q5)<0.or.STATE(r5)==BUSY_RES,239$,Yes;
338$    ASSIGN:    bbb=1:
        top5=0;
239$    DELAY:      preparation;

```

```

261$    RELEASE:    operation5,1:NEXT(52$);

48$    ASSIGN:    preparation=27:
           res5=part: NEXT(49$);

290$    ASSIGN:    processtime=amount*r(part,2):
           top5=top5+processtime+27:NEXT(44$);

b6      BRANCH,:    If,bbc==1,291$,Yes:
           If,bbc>1,292$,Yes;
291$    ASSIGN:    processtime=amount*r(part,2):
           top6=top6+processtime+27:
           bbc=bbc+1;
76$    QUEUE,    q6;
77$    SEIZE,    1:
           r6,1:
           operation6,1;
79$    BRANCH,:    If,res6==part,78$,Yes:
           If,res6<>part,80$,Yes;
78$    ASSIGN:    preparation=5:
           res6=part:
           top6=top6-22;
81$    DELAY:    processtime;
82$    RELEASE:    r6,1;
372$    WRITE,    fileofres6,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;
339$    BRANCH,    1:
           If,NQ(q6)==0.and.STATE(r6)==IDLE_RES,340$,Yes:
           If,NQ(q6)<>0.or.STATE(r6)==BUSY_RES,238$,Yes;
340$    ASSIGN:    bbc=1:
           top6=0;
238$    DELAY:    preparation;
262$    RELEASE:    operation6,1:NEXT(52$);

80$    ASSIGN:    preparation=27:
           res6=part:NEXT(81$);

292$    ASSIGN:    processtime=amount*r(part,2):
           top6=top6+processtime+27:NEXT(76$);

32$    SEARCH,    q4,1,NQ(q4):part==ara;
33$    BRANCH,:    If,J>0,b4,Yes:
           If,J==0,35$,Yes;
35$    SEARCH,    q5,1,NQ(q5):part==ara;
34$    BRANCH,:    If,J>0,b5,Yes:
           If,J==0,75$,Yes;
75$    SEARCH,    q6,1,NQ(q6):part==ara;
74$    BRANCH,:    If,J>0,b6,Yes:
           If,J==0,36$,Yes;
36$    BRANCH,    1:
           If,top4==MN(top5,top4,top6),b4,Yes:
           If,top5==MN(top5,top4,top6),b5,Yes:
           If,top6==MN(top5,top4,top6),b6,Yes;
73$    ASSIGN:    time23(bbc)=TNOW:
           top6=top6-time23(bbc)+time23(bbc-1):
           bbc=bbc+1:NEXT(31$);

30$    ASSIGN:    time22(bbb)=TNOW:
           top5=top5-time22(bbb)+time22(bbb-1):

```

```

bbb=bbb+1:NEXT(72$);

28$    ASSIGN:    time21(bba)=TNOW:
        top4=top4-time21(bba)+time21(bba-1):
        bba=bba+1:NEXT(29$);

103$   STATION,    s3;
105$   ASSIGN:    ara=part;
83$    BRANCH,:    If,gga==1,293$,Yes:
        If,gga>1,84$,Yes;
293$   ASSIGN:    time31(gga)=TNOW;
85$    BRANCH,:    If,ggb==1,294$,Yes:
        If,ggb>1,86$,Yes;
294$   ASSIGN:    time32(ggb)=TNOW;
87$    BRANCH,:    If,part==res7,b7,Yes:
        If,part==res8,b8,Yes:
        If,part<>res7.and.part<>res8,88$,Yes;
b7     BRANCH,:    If,gga==1,295$,Yes:
        If,gga>1,296$,Yes;
295$   ASSIGN:    processtime=amount*r(part,3):
        top7=top7+processtime+27:
        gga=gga+1;
93$    QUEUE,      q7;
94$    SEIZE,       1:
        r7,1:
        operation7,1;
95$    BRANCH,:    If,res7==part,245$,Yes:
        If,res7<>part,246$,Yes;
245$   ASSIGN:    preparation=5:
        res7=part:
        top7=top7-22;
96$    DELAY:      processtime;
97$    RELEASE:    r7,1;
373$   WRITE,      fileofres7,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;
349$   BRANCH,     1:
        If,NQ(q7)==0.and.STATE(r7)==IDLE_RES,350$,Yes:
        If,NQ(q7)<>0.or.STATE(r7)==BUSY_RES,247$,Yes;
350$   ASSIGN:    gga=1:
        top7=0;
247$   DELAY:      preparation;
267$   RELEASE:    operation7,1;
104$   ROUTE:      0.0,SEQ;

246$   ASSIGN:    preparation=27:
        res7=part: NEXT(96$);

296$   ASSIGN:    processtime=amount*r(part,3):
        top7=top7+processtime+27:NEXT(93$);

b8     BRANCH,:    If,ggb==1,297$,Yes:
        If,ggb>1,298$,Yes;
297$   ASSIGN:    processtime=amount*r(part,3):
        top8=top8+processtime+27:
        ggb=ggb+1;
98$    QUEUE,      q8;
99$    SEIZE,       1:
        r8,1:
        operation8,1;

```

```

100$   BRANCH,:   If,res8==part,248$,Yes:
        If,res8<>part,249$,Yes;
248$   ASSIGN:    preparation=5:
        res8=part:
        top8=top8-22;
101$   DELAY:     processtime;
102$   RELEASE:   r8,1;
374$   WRITE,     fileofres8,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;
351$   BRANCH,    1:
        If,NQ(q8)==0.and.STATE(r8)==IDLE_RES,352$,Yes:
        If,NQ(q8)<>0.or.STATE(r8)==BUSY_RES,250$,Yes;
352$   ASSIGN:    ggb=1:
        top8=0;
250$   DELAY:     preparation;
268$   RELEASE:   operation8,1:NEXT(104$);

249$   ASSIGN:    preparation=27:
        res8=part: NEXT(101$);

298$   ASSIGN:    processtime=amount*r(part,3):
        top8=top8+processtime+27:NEXT(98$);

88$   SEARCH,     q7,1,NQ(q7):part==ara;
89$   BRANCH,:    If,J>0,b7,Yes:
        If,J==0,91$,Yes;
91$   SEARCH,     q8,1,NQ(q8):part==ara;
90$   BRANCH,:    If,J>0,b8,Yes:
        If,J==0,92$,Yes;
92$   BRANCH,    1:
        If,top7==MN(top7,top8),b7,Yes:
        If,top8==MN(top7,top8),b8,Yes;
86$   ASSIGN:     time32(ggb)=TNOW:
        top8=top8-time32(ggb)+time32(ggb-1):
        ggb=ggb+1:NEXT(87$);

84$   ASSIGN:     time31(gga)=TNOW:
        top7=top7-time31(gga)+time31(gga-1):
        gga=gga+1:NEXT(85$);

130$   STATION,    s4;
132$   ASSIGN:     ara=part;
106$   BRANCH,:    If,hha==1,299$,Yes:
        If,hha>1,107$,Yes;
299$   ASSIGN:     time41(hha)=TNOW;
108$   BRANCH,:    If,hhb==1,300$,Yes:
        If,hhb>1,109$,Yes;
300$   ASSIGN:     time42(hhb)=TNOW;
110$   BRANCH,:    If,part==res9,b9,Yes:
        If,part==res10,b10,Yes:
        If,part<>res9.and.part<>res10,111$,Yes;
b9     BRANCH,:    If,hha==1,301$,Yes:
        If,hha>1,302$,Yes;
301$   ASSIGN:     processtime=amount*r(part,4):
        top9=top9+processtime+27:
        hha=hha+1;
116$   QUEUE,     q9;
117$   SEIZE,      1:

```

```

        r9,1:
        operation9,1;
119$  BRANCH,:    If,res9==part,118$,Yes:
        If,res9<>part,120$,Yes;
118$  ASSIGN:    preparation=5:
        res9=part:
        top9=top9-22;
121$  DELAY:    processtime;
122$  RELEASE:   r9,1;
375$  WRITE,    fileofres9,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
353$  BRANCH,    1:
        If,NQ(q9)==0.and.STATE(r9)==IDLE_RES,354$,Yes:
        If,NQ(q9)<>0.or.STATE(r9)==BUSY_RES,252$,Yes;
354$  ASSIGN:    hha=1:
        top9=0;
252$  DELAY:    preparation;
269$  RELEASE:   operation9,1;
131$  ROUTE:    0.0,SEQ;

120$  ASSIGN:    preparation=27:
        res9=part: NEXT(121$);

302$  ASSIGN:    processtime=amount*r(part,4):
        top9=top9+processtime+27:NEXT(116$);

b10   BRANCH,:    If,hhb==1,303$,Yes:
        If,hhb>1,304$,Yes;
303$  ASSIGN:    processtime=amount*r(part,4):
        top10=top10+processtime+27:
        hhb=hb+1;
123$  QUEUE,    q10;
124$  SEIZE,    1:
        r10,1:
        operation10,1;
126$  BRANCH,:    If,res10==part,125$,Yes:
        If,res10<>part,127$,Yes;
125$  ASSIGN:    preparation=5:
        res10=part:
        top10=top10-22;
128$  DELAY:    processtime;
129$  RELEASE:   r10,1;
376$  WRITE,    fileofres10,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
355$  BRANCH,    1:
        If,NQ(q10)==0.and.STATE(r10)==IDLE_RES,356$,Yes:
        If,NQ(q10)<>0.or.STATE(r10)==BUSY_RES,251$,Yes;
356$  ASSIGN:    hhb=1:
        top10=0;
251$  DELAY:    preparation;
270$  RELEASE:   operation10,1:NEXT(131$);

127$  ASSIGN:    preparation=27:
        res10=part: NEXT(128$);

304$  ASSIGN:    processtime=amount*r(part,4):
        top10=top10+processtime+27:NEXT(123$);

111$  SEARCH,    q9,1,NQ(q9):part==ara;
112$  BRANCH,:    If,J>0,b9,Yes:

```

```

        If,J==0,114$,Yes;
114$  SEARCH,    q10,1,NQ(q10):part==ara;
113$  BRANCH,,:  If,J>0,b10,Yes:
        If,J==0,115$,Yes;
115$  BRANCH,    1:
        If,top9==MN(top9,top10),b9,Yes:
        If,top10==MN(top9,top10),b10,Yes;
109$  ASSIGN:    time42(hhb)=TNOW:
        top10=top10-time42(hhb)+time42(hhb-1):
        hhb=hhb+1:NEXT(110$);

107$  ASSIGN:    time41(hha)=TNOW:
        top9=top9-time41(hha)+time41(hha-1):
        hha=hha+1:NEXT(108$);


157$  STATION,    s5;
159$  ASSIGN:    ara=part;
133$  BRANCH,,:  If,iia==1,305$,Yes:
        If,iia>1,134$,Yes;
305$  ASSIGN:    time51(iia)=TNOW;
135$  BRANCH,,:  If,iib==1,306$,Yes:
        If,iib>1,136$,Yes;
306$  ASSIGN:    time52(iib)=TNOW;
160$  BRANCH,,:  If,iic==1,307$,Yes:
        If,iic>1,161$,Yes;
307$  ASSIGN:    time53(iic)=TNOW;
162$  BRANCH,,:  If,iid==1,308$,Yes:
        If,iid>1,163$,Yes;
308$  ASSIGN:    time54(iid)=TNOW;
137$  BRANCH,,:  If,part==res11,b11,Yes:
        If,part==res12,b12,Yes:
        If,part==res13,b13,Yes:
        If,part==res14,b14,Yes:
        If,part<>res11.and.part<>res12.and.part<>res13.and.part<>res14,138$,Yes;
b11  BRANCH,,:  If,iia==1,309$,Yes:
        If,iia>1,310$,Yes;
309$  ASSIGN:    processtime=amount*r(part,5):
        top11=top11+processtime+27:
        iia=iia+1;
143$  QUEUE,    q11;
144$  SEIZE,    1:
        r11,1:
        operation11,1;
146$  BRANCH,,:  If,res11==part,145$,Yes:
        If,res11<>part,147$,Yes;
145$  ASSIGN:    preparation=5:
        res11=part:
        top11=top11-22;
148$  DELAY:    processtime;
149$  RELEASE:    r11,1;
377$  WRITE,    fileofres11,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
357$  BRANCH,    1:
        If,NQ(q11)==0.and.STATE(r11)==IDLE_RES,358$,Yes:
        If,NQ(q11)<>0.or.STATE(r11)==BUSY_RES,253$,Yes;
358$  ASSIGN:    iia=1:
        top11=0;
253$  DELAY:    preparation;

```

```

271$    RELEASE:    operation11,1;
158$    ROUTE:      0.0,SEQ;

147$    ASSIGN:     preparation=27:
                res11=part: NEXT(148$);

310$    ASSIGN:     processtime=amount*r(part,5):
                top11=top11+processtime+27:NEXT(143$);

b12     BRANCH,:    If,iib==1,311$,Yes:
                If,iib>1,312$,Yes;
311$    ASSIGN:     processtime=amount*r(part,5):
                top12=top12+processtime+27:
                iib=iib+1;
150$    QUEUE,      q12;
151$    SEIZE,       1:
                r12,1:
                operation12,1;
153$    BRANCH,:    If,res12==part,152$,Yes:
                If,res12<>part,154$,Yes;
152$    ASSIGN:     preparation=5:
                res12=part:
                top12=top12-22;
155$    DELAY:      processtime;
156$    RELEASE:    r12,1;
378$    WRITE,      fileofres12,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
359$    BRANCH,     1:
                If,NQ(q12)==0.and.STATE(r12)==IDLE_RES,360$,Yes:
                If,NQ(q12)<>0.or.STATE(r12)==BUSY_RES,254$,Yes;
360$    ASSIGN:     iib=1:
                top12=0;
254$    DELAY:      preparation;
272$    RELEASE:    operation12,1:NEXT(158$);

154$    ASSIGN:     preparation=27:
                res12=part: NEXT(155$);

312$    ASSIGN:     processtime=amount*r(part,5):
                top12=top12+processtime+27:NEXT(150$);

b13     BRANCH,:    If,iic==1,313$,Yes:
                If,iic>1,314$,Yes;
313$    ASSIGN:     processtime=amount*r(part,5):
                top13=top13+processtime+27:
                iic=iic+1;
168$    QUEUE,      q13;
169$    SEIZE,       1:
                r13,1:
                operation13,1;
171$    BRANCH,:    If,res13==part,170$,Yes:
                If,res13<>part,172$,Yes;
170$    ASSIGN:     preparation=5:
                res13=part:
                top13=top13-22;
173$    DELAY:      processtime;
174$    RELEASE:    r13,1;
379$    WRITE,      fileofres13,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
363$    BRANCH,     1:

```

```

        If,NQ(q13)==0.and.STATE(r13)==IDLE_RES,364$,Yes:
        If,NQ(q13)<>0.or.STATE(r13)==BUSY_RES,255$,Yes;
364$  ASSIGN:    iic=1:
        top13=0;
255$  DELAY:    preparation;
273$  RELEASE:  operation13,1:NEXT(158$);

172$  ASSIGN:    preparation=27:
        res13=part: NEXT(173$);

314$  ASSIGN:    processtime=amount*r(part,5):
        top13=top13+processtime+27:NEXT(168$);

b14   BRANCH,:  If,iid==1,315$,Yes:
        If,iid>1,316$,Yes;
315$  ASSIGN:    processtime=amount*r(part,5):
        top14=top14+processtime+27:
        iid=iid+1;
175$  QUEUE,    q14;
176$  SEIZE,    1:
        r14,1:
        operation14,1;
178$  BRANCH,:  If,res14==part,177$,Yes:
        If,res14<>part,179$,Yes;
177$  ASSIGN:    preparation=5:
        res14=part:
        top14=top14-22;
180$  DELAY:    processtime;
181$  RELEASE:  r14,1;
380$  WRITE,    fileofres14,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
361$  BRANCH,    1:
        If,NQ(q14)==0.and.STATE(r14)==IDLE_RES,362$,Yes:
        If,NQ(q14)<>0.or.STATE(r14)==BUSY_RES,256$,Yes;
362$  ASSIGN:    iid=1:
        top14=0;
256$  DELAY:    preparation;
274$  RELEASE:  operation14,1:NEXT(158$);

179$  ASSIGN:    preparation=27:
        res14=part: NEXT(180$);

316$  ASSIGN:    processtime=amount*r(part,5):
        top14=top14+processtime+27:NEXT(175$);

138$  SEARCH,    q11,1,NQ(q11):part==ara;
139$  BRANCH,:  If,J>0,b11,Yes:
        If,J==0,141$,Yes;
141$  SEARCH,    q12,1,NQ(q12):part==ara;
140$  BRANCH,:  If,J>0,b12,Yes:
        If,J==0,165$,Yes;
165$  SEARCH,    q13,1,NQ(q13):part==ara;
164$  BRANCH,:  If,J>0,b13,Yes:
        If,J==0,167$,Yes;
167$  SEARCH,    q14,1,NQ(q14):part==ara;
166$  BRANCH,:  If,J>0,b14,Yes:
        If,J==0,142$,Yes;
142$  BRANCH,    1:
        If,top11==MN(top11,top12,top13,top14),b11,Yes:

```

```

If,top12==MN(top11,top12,top13,top14),b12,Yes:
If,top13==MN(top11,top12,top13,top14),b13,Yes:
If,top14==MN(top11,top12,top13,top14),b14,Yes;
163$  ASSIGN:    time54(iid)=TNOW:
        top14=top14-time54(iid)+time54(iid-1):
        iid=iid+1:NEXT(137$);

161$  ASSIGN:    time53(iic)=TNOW:
        top13=top13-time53(iic)+time53(iic-1):
        iic=iic+1:NEXT(162$);

136$  ASSIGN:    time52(iib)=TNOW:
        top12=top12-time52(iib)+time52(iib-1):
        iib=iib+1:NEXT(160$);

134$  ASSIGN:    time51(iaa)=TNOW:
        top11=top11-time51(iaa)+time51(iaa-1):
        iia=iaa+1:NEXT(135$);

206$  STATION,    s6;
208$  ASSIGN:    ara=part;
182$  BRANCH,:    If,kka==1,317$,Yes:
        If,kka>1,183$,Yes;
317$  ASSIGN:    time61(kka)=TNOW;
184$  BRANCH,:    If,kkb==1,319$,Yes:
        If,kkb>1,185$,Yes;
319$  ASSIGN:    time62(kkb)=TNOW;
209$  BRANCH,:    If,kkc==1,318$,Yes:
        If,kkc>1,210$,Yes;
318$  ASSIGN:    time63(kkc)=TNOW;
211$  BRANCH,:    If,kkd==1,320$,Yes:
        If,kkd>1,212$,Yes;
320$  ASSIGN:    time64(kkd)=TNOW;
186$  BRANCH,:    If,part==res15,b15,Yes:
        If,part==res16,b16,Yes:
        If,part==res17,b17,Yes:
        If,part==res18,b18,Yes:
        If,part<>res15.and.part<>res16.and.part<>res17.and.part<>res18,187$,Yes;
b15   BRANCH,:    If,kka==1,321$,Yes:
        If,kka>1,322$,Yes;
321$  ASSIGN:    processtime=amount*r(part,6):
        top15=top15+processtime+27:
        kka=kka+1;
192$  QUEUE,      q15;
193$  SEIZE,       1:
        r15,1:
        operation15,1;
195$  BRANCH,:    If,res15==part,194$,Yes:
        If,res15<>part,196$,Yes;
194$  ASSIGN:    preparation=5:
        res15=part:
        top15=top15-22;
197$  DELAY:      processtime;
198$  RELEASE:    r15,1;
381$  WRITE,      fileofres15,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
341$  BRANCH,     1:
        If,NQ(q15)==0.and.STATE(r15)==IDLE_RES,342$,Yes:

```

```

                                If,NQ(q15)<>0.or.STATE(r15)==BUSY_RES,241$,Yes;
342$    ASSIGN:    kka=1:
                                top15=0;
241$    DELAY:    preparation;
263$    RELEASE:    operation15,1;
207$    ROUTE:    0.0,SEQ;

196$    ASSIGN:    preparation=27:
                                res15=part: NEXT(197$);

322$    ASSIGN:    processtime=amount*r(part,6):
                                top15=top15+processtime+27:NEXT(192$);

b16    BRANCH,:    If,kkb==1,323$,Yes:
                                If,kkb>1,324$,Yes;
323$    ASSIGN:    processtime=amount*r(part,6):
                                top16=top16+processtime+27:
                                kkb=kkb+1;
199$    QUEUE,    q16;
200$    SEIZE,    1:
                                r16,1:
                                operation16,1;
202$    BRANCH,:    If,res16==part,201$,Yes:
                                If,res16<>part,203$,Yes;
201$    ASSIGN:    preparation=5:
                                res16=part:
                                top16=top16-22;
204$    DELAY:    processtime;
205$    RELEASE:    r16,1;
382$    WRITE,    fileofres16,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;;
343$    BRANCH,    1:
                                If,NQ(q16)==0.and.STATE(r16)==IDLE_RES,344$,Yes:
                                If,NQ(q16)<>0.or.STATE(r16)==BUSY_RES,242$,Yes;
344$    ASSIGN:    kkb=1:
                                top16=0;
242$    DELAY:    preparation;
264$    RELEASE:    operation16,1:NEXT(207$);

203$    ASSIGN:    preparation=27:
                                res16=part: NEXT(204$);

324$    ASSIGN:    processtime=amount*r(part,6):
                                top16=top16+processtime+27:NEXT(199$);

b17    BRANCH,:    If,kkc==1,325$,Yes:
                                If,kkc>1,326$,Yes;
325$    ASSIGN:    processtime=amount*r(part,6):
                                top17=top17+processtime+27:
                                kkc=kkc+1;
217$    QUEUE,    q17;
218$    SEIZE,    1:
                                r17,1:
                                operation17,1;
220$    BRANCH,:    If,res17==part,219$,Yes:
                                If,res17<>part,221$,Yes;
219$    ASSIGN:    preparation=5:
                                res17=part:
                                top17=top17-22;

```

```

222$   DELAY:      processtime;
223$   RELEASE:    r17,1;
383$   WRITE,      fileofres17,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;
345$   BRANCH,      1:
          If,NQ(q17)==0.and.STATE(r17)==IDLE_RES,346$,Yes:
          If,NQ(q17)<>0.or.STATE(r17)==BUSY_RES,243$,Yes;
346$   ASSIGN:      kkc=1:
          top17=0;
243$   DELAY:      preparation;
265$   RELEASE:    operation17,1:NEXT(207$);

221$   ASSIGN:      preparation=27:
          res17=part: NEXT(222$);

326$   ASSIGN:      processtime=amount*r(part,6):
          top17=top17+processtime+27:NEXT(217$);

b18    BRANCH,:    If,kkd==1,327$,Yes:
          If,kkd>1,328$,Yes;
327$   ASSIGN:      processtime=amount*r(part,6):
          top18=top18+processtime+27:
          kkd=kkd+1;
224$   QUEUE,      q18;
225$   SEIZE,      1:
          r18,1:
          operation18,1;
227$   BRANCH,:    If,res18==part,226$,Yes:
          If,res18<>part,228$,Yes;
226$   ASSIGN:      preparation=5:
          res18=part:
          top18=top18-22;
229$   DELAY:      processtime;
230$   RELEASE:    r18,1;
384$   WRITE,      fileofres18,"(17x,'.....product#:',3x,F5.0,3x,'start:',F7.1)":product,TNOW;;
347$   BRANCH,      1:
          If,NQ(q18)==0.and.STATE(r18)==IDLE_RES,348$,Yes:
          If,NQ(q18)<>0.or.STATE(r18)==BUSY_RES,244$,Yes;
348$   ASSIGN:      kkd=1:
          top18=0;
244$   DELAY:      preparation;
266$   RELEASE:    operation18,1:NEXT(207$);

228$   ASSIGN:      preparation=27:
          res18=part: NEXT(229$);

328$   ASSIGN:      processtime=amount*r(part,6):
          top18=top18+processtime+27:NEXT(224$);

187$   SEARCH,      q15,1,NQ(q15):part==ara;
188$   BRANCH,:    If,J>0,b15,Yes:
          If,J==0,190$,Yes;
190$   SEARCH,      q16,1,NQ(q16):part==ara;
189$   BRANCH,:    If,J>0,b16,Yes:
          If,J==0,214$,Yes;
214$   SEARCH,      q17,1,NQ(q17):part==ara;
213$   BRANCH,:    If,J>0,b17,Yes:
          If,J==0,216$,Yes;
216$   SEARCH,      q18,1,NQ(q18):part==ara;

```

```

215$    BRANCH,:    If,J>0,b18,Yes:
                If,J==0,191$,Yes;
191$    BRANCH,    1:
                If,top15==MN(top15,top16,top17,top18),b15,Yes:
                If,top16==MN(top15,top16,top17,top18),b16,Yes:
                If,top17==MN(top15,top16,top17,top18),b17,Yes:
                If,top18==MN(top15,top16,top17,top18),b18,Yes;
212$    ASSIGN:    time64(kkd)=TNOW:
                top18=top18-time64(kkd)+time64(kkd-1):
                kkd=kkd+1:NEXT(186$);

210$    ASSIGN:    time63(kkc)=TNOW:
                top17=top17-time63(kkc)+time63(kkc-1):
                kkc=kkc+1:NEXT(211$);

185$    ASSIGN:    time62(kkb)=TNOW:
                top16=top16-time62(kkb)+time62(kkb-1):
                kkb=kkb+1:NEXT(209$);

183$    ASSIGN:    time61(kka)=TNOW:
                top15=top15-time61(kka)+time61(kka-1):
                kka=kka+1:NEXT(184$);

53$    STATION,    s7;
365$    BRANCH,:    If,TNOW<12000,54$,Yes:
                If,TNOW>=12000,366$,Yes;
54$    DISPOSE;

366$    ASSIGN:    exitsystem=TNOW:
                timeinsystem=exitsystem-entersystem:
                numberofparts=numberofparts+1:
                totaltime=totaltime+timeinsystem:
                meancompletiontime=totaltime/numberofparts;
59$    BRANCH,:    If,TNOW>duedate,60$,Yes:
                If,TNOW<=duedate,54$,Yes;
60$    ASSIGN:    tardiness=TNOW-duedate:
                totaltardiness=totaltardiness+tardiness:
                latejobs=latejobs+1:
                meantardiness=totaltardiness/latejobs:NEXT(54$);

```

```

ATTRIBUTES: product:
            totalprocesstime:
            part:
            entersystem:
            tardiness:
            amount:
            preparation:
            processtime:
            timeinsystem:
            duedate:
            exitsystem:

```

```

FILES: fileofres17,"c:\arena3\file(res17).dat",Sequential(),Free Format,Error,No,Close:
fileofres18,"c:\arena3\file(res18).dat",Sequential(),Free Format,Error,No,Close:
fileofres1,"c:\arena3\file(res1).dat",Sequential(),Free Format,Error,No,Close:
fileofres2,"c:\arena3\file(res2).dat",Sequential(),Free Format,Error,No,Close:

```

```

fileofres3,"c:\arena3\file(res3).dat",Sequential(),Free Format>Error,No,Close:
fileofres10,"c:\arena3\file(res10).dat",Sequential(),Free Format>Error,No,Close:
fileofres4,"c:\arena3\file(res4).dat",Sequential(),Free Format>Error,No,Close:
fileofres11,"c:\arena3\file(res11).dat",Sequential(),Free Format>Error,No,Close:
fileofres5,"c:\arena3\file(res5).dat",Sequential(),Free Format>Error,No,Close:
fileofres12,"c:\arena3\file(res12).dat",Sequential(),Free Format>Error,No,Close:
fileofres6,"c:\arena3\file(res6).dat",Sequential(),Free Format>Error,No,Close:
fileofres13,"c:\arena3\file(res13).dat",Sequential(),Free Format>Error,No,Close:
fileofres7,"c:\arena3\file(res7).dat",Sequential(),Free Format>Error,No,Close:
fileofres14,"c:\arena3\file(res14).dat",Sequential(),Free Format>Error,No,Close:
fileofres8,"c:\arena3\file(res8).dat",Sequential(),Free Format>Error,No,Close:
fileofres15,"c:\arena3\file(res15).dat",Sequential(),Free Format>Error,No,Close:
fileofres9,"c:\arena3\file(res9).dat",Sequential(),Free Format>Error,No,Close:
fileofres16,"c:\arena3\file(res16).dat",Sequential(),Free Format>Error,No,Close:
fileofexit,"c:\arena3\file(exit).dat",Sequential(),Free Format>Error,No,Close;

```

VARIABLES: top3,0:

```

iid,1:
kkd,1:
time13(4600):
res17:
top11,0:
top4,0:
coefficient,1:
res18:
top12,0:
totaltime,0:
meancompletiontime,0:
top5,0:
top13,0:
top6,0:
top14,0:
res1,:
latejobs,0:
nextshift,:
top7,0:
top15,0:
res2,:
top8,0:
top16,0:
res3,:
meantardiness,0:
top9,0:
top17,0:
res4,:
top18,0:
totalardiness,0:
res5:
numberofparts,0:
res6,:
res10:
res7:
time61(4600):
type,1:
res11:
res8:
time51(4600):
time62(4600):

```

```

res9:
res12:
time41(4600):
time52(4600):
time63(4600):
res13:
time31(4600):
time42(4600):
time53(4600):
time64(4600):

```

```

r(15,6),0.3,0.28,0.25,0.27,0.25,0.22,0.2,0.19,0.31,0.3,0.29,0.26,0.27,0.24,0.25,0.25,0.27,0.3,0.24,0.22,
0.2,0.25,

```

```

0.26,0.25,0.29,0.24,0.26,0.27,0.26,0.29,0.15,0.16,0.17,0.15,0.16,0.15,0.14,0.18,0.15,0.14,0.15,0.16,0.
12,0.15,

```

```

0.14,0.16,0.14,0.13,0.15,0.16,0.15,0.1,0.1,0.09,0.17,0.14,0.17,0.16,0.18,0.17,0.32,0.35,0.36,0.38,0.4,0.
35,0.36,

```

```

0.38,0.35,0.37,0.33,0.31,0.35,0.32,0.35,0.35,0.33,0.34,0.36,0.35,0.34,0.35,0.33,0.37,0.39,0.34,0.36,0.
3,0.34,0.3:

```

```

ara,0:
aaa,1:
bba,1:
hha,1:
iia,1:
kka,1:
gga,1:
time21(4600):
res14:
time32(4600):
time54(4600):
top1,0:
aab,1:
bbb,1:
hhb,1:
iib,1:
kkb,1:
ggb,1:
time11(4600):
time22(4600):
res15:
top2,0:
aac,1:
bbc,1:
iic,1:
kkc,1:
time12(4600):
time23(4600):
res16:
top10,0;

```

```

SEEDS:      seed1,30000,Common:
           seed2,40000,Common;

```

```

QUEUES:     q15,LVF(processtime):
           q16,LVF(processtime):

```

q1,LVF(processtime):
 q17,LVF(processtime):
 q2,LVF(processtime):
 q18,LVF(processtime):
 q3,LVF(processtime):
 q4,LVF(processtime):
 q5,LVF(processtime):
 q6,LVF(processtime):
 q7,LVF(processtime):
 q8,LVF(processtime):
 q9,LVF(processtime):
 q10,LVF(processtime):
 q11,LVF(processtime):
 q12,LVF(processtime):
 q13,LVF(processtime):
 q14,LVF(processtime);

RESOURCES: r14,Capacity(1),-,Stationary:
 operation10,Capacity(1),-,Stationary:
 operation6,Capacity(1),-,Stationary:
 r1,Capacity(1),-,Stationary:
 r15,Capacity(1),-,Stationary:
 operation11,Capacity(1),-,Stationary:
 operation7,Capacity(1),-,Stationary:
 r2,Capacity(1),-,Stationary:
 r16,Capacity(1),-,Stationary:
 operation12,Capacity(1),-,Stationary:
 operation8,Capacity(1),-,Stationary:
 r3,Capacity(1),-,Stationary:
 r17,Capacity(1),-,Stationary:
 operation13,Capacity(1),-,Stationary:
 operation9,Capacity(1),-,Stationary:
 r4,Capacity(1),-,Stationary:
 r18,Capacity(1),-,Stationary:
 operation14,Capacity(1),-,Stationary:
 operation15,Capacity(1),-,Stationary:
 r5,Capacity(1),-,Stationary:
 operation16,Capacity(1),-,Stationary:
 r6,Capacity(1),-,Stationary:
 operation17,Capacity(1),-,Stationary:
 r7,Capacity(1),-,Stationary:
 operation18,Capacity(1),-,Stationary:
 r8,Capacity(1),-,Stationary:
 r9,Capacity(1),-,Stationary:
 operation1,Capacity(1),-,Stationary:
 operation2,Capacity(1),-,Stationary:
 r10,Capacity(1),-,Stationary:
 operation3,Capacity(1),-,Stationary:
 r11,Capacity(1),-,Stationary:
 operation4,Capacity(1),-,Stationary:
 r12,Capacity(1),-,Stationary:
 r13,Capacity(1),-,Stationary:
 operation5,Capacity(1),-,Stationary;

STATIONS: s1:
 s2:
 s3:
 s4:

s5:
s6:
s7;

SEQUENCES: 1,seq1,s6&s2&s5&s4&s7:
2,seq2,s5&s6&s1&s2&s7:
3,seq3,s5&s2&s4&s1&s7:
4,seq4,s1&s2&s5&s6&s7:
5,seq5,s3&s2&s6&s5&s7:
6,seq6,s1&s2&s6&s5&s7:
7,seq7,s1&s6&s5&s4&s7:
8,seq8,s1&s2&s3&s6&s7:
9,seq9,s4&s3&s6&s5&s7:
10,seq10,s5&s1&s3&s6&s7:
11,seq11,s3&s5&s4&s6&s7:
12,seq12,s5&s1&s2&s6&s7:
13,seq13,s3&s6&s2&s5&s7:
14,seq14,s4&s6&s5&s1&s7:
15,seq15,s5&s1&s6&s2&s7;

DSTATS: nq(q3):
nq(q18):
nr(r16):
nr(r5):
nq(q2):
nq(q17):
nr(r4):
nr(r15):
nq(q1):
nq(q16):
nr(r3):
nr(r14):
nq(q15):
nr(r2):
nr(r13):
nq(q14):
nr(r1):
nr(r12):
nq(q13):
nr(r11):
nq(q12):
nr(r10):
nq(q11):
nq(q10):
nq(q9):
nq(q8):
nq(q7):
nr(r9):
nq(q6):
nr(r8):
nr(r18):
nq(q5):
nr(r7):
nq(q4):
nr(r17):
nr(r6);

REPLICATE, 20,0.0,132000,Yes,Yes,12000;