AN INTEGRATED SCHEDULING APPROACH FOR DISCRETE MANUFACTURING SYSTEMS

By

AJAY KUMAR JAIN, B. Eng., M.Sc.

A Thesis

Submitted to the School of Graduate Studies

In Partial Fulfillment of the Requirements

for the Degree

Doctor of Philosophy

McMaster University

AN INTEGRATED SCHEDULING APPROACH FOR DISCRETE MANUFACTURING SYSTEMS

DOCTOR OF PHILOSOPHY (1995) McMASTER UNIVERSITY
(Mechanical Engineering) Hamilton, Ontario

TITLE: An Integrated Scheduling Approach for Discrete Manufacturing Systems

AUTHOR: Ajay Kumar Jain
B. Eng. (Mangalore University, India)
M. Sc. (University of Regina)

SUPERVISOR: Dr. Hoda A. ElMaraghy
Professor and Director of Flexible Manufacturing Centre (till June 30 '94)
Dean of Engineering, The University of Windsor (July 1994)

NUMBER OF PAGES: xv, 230

ABSTRACT

Efficient scheduling in discrete manufacturing environment can improve productivity through reduced work–in–progress and finished good inventories. This research examined the feasibility and advantages of using genetic algorithms in manufacturing scheduling. All literature to date indicate that there is a need for better scheduling algorithms which can provide better and faster solutions. This thesis focuses on the improvement of the scheduling performance through: a) the application of genetic algorithms to the schedule optimization problem, b) the consideration of batch splitting and c) dynamic scheduling.

First, genetic algorithms are developed for two types of scheduling situations: a) scheduling in the presence of single process plans without any routing flexibility and b) scheduling where routing flexibility and multiple process plans exist. Analytical techniques and heuristics–based methods have been developed by other researchers to solve such scheduling problems which are useful for limited size problems. Since the schedule population size is fixed in genetic algorithms, there is a tremendous reduction in the memory requirements. In every generation bad schedules are replaced by better ones. Also, since genetic algorithms performs a multi–point search at a time, they are relatively faster than other techniques used for scheduling which are usually based on single point search. Genetic algorithms models were formulated and tested using nineteen example problems for both flexible and fixed routing scenarios. Numerical results show that the genetic algorithms approach perform significantly better than other approaches both in terms of finding the optimal/near optimal solution and the computation time. The saving in computation time was up to 50% in some cases. Because of the vast number of schedules available in the case of multiple process plans, dispatching rules have been employed in the genetic algorithms to obtain a satisfactory schedule. It is often observed that in the case of alternate routings and while using dispatching rules, researchers tended to use a single job in each order which is rarely the case in practice. In this thesis, both alternate routings and order sizes greater

than one are simultaneously considered along with dispatching rules while scheduling with genetic algorithms. Twelve different dispatching rules and seven performance criteria are used and the performance of each rule has been studied extensively with respect to each of the performance criteria.

Second, three new batch splitting policies were introduced in this research to split the batches into smaller sub–batches. A process plan–based method was developed, where the information available from the process plans is used for batch splitting. This is different from other batching policies where batching is done based on inventory models from queueing theory for which cost function consisting of ordering, inventory holding and work–in–process carrying costs are required. The new batch splitting policies were compared with existing ones and the results show that in several cases the newly developed policies performed significantly better than existing ones. The improvement in the performance was observed to be in the range of 35% to 45%.

Next, the dynamic scheduling aspects are considered and rescheduling algorithms were developed for four kinds of uncertainties. These include unexpected machine breakdown, rush order arrival, order cancellation and increased order priority. These algorithms reschedule only interrupted tasks, hence focusing on local rescheduling. These algorithms were tested using three different performance criteria namely, mean flow time, mean tardiness, and average machine utilization. The developed rescheduling algorithms proved successful in dealing with the shop floor uncertainties and the results indicate that the proposed algorithms are effective and could easily be used to dynamically schedule the manufacturing systems in the presence of disturbances.

Finally, as part of the system implementation, a user interface has been developed in 'C' to input required scheduling data used for schedule optimization. The scheduler outputs a list of several good schedules. These schedules are then analyzed by the output analyzer which outputs a list of different performance criteria values, machine utilizations, ready and completion times of each order and the Gantt chart of the best schedule.

ACKNOWLEDGEMENTS

The author wishes to express his sincere appreciation to his supervisor, Dr. H. A. El-Maraghy, for her constant support and guidance during the course of this research. The author wishes to acknowledge and thank Dr. A. R. Montazemi and Dr. B. Sowerby, who were members of his supervisory committee.

Special thanks are also due to the systems analyst, Mr. Todd Pfaff for his programming assistance and to the administrative assistant, Ms. Patricia Jackson, for her help with the write-up. Many thanks are also due to friends and colleagues, too numerous to list individually by name, for their invaluable help and support.

Appreciation is extended to the Department of Mechanical Engineering for providing me the opportunity to be a teaching assistant. The financial support in the form of a research scholarship from Dr. H. A. ElMaraghy is also greatly appreciated.

The author would like to convey special thanks to his wife, Ritu, for her love, patience, and moral support, and to his newborn daughter, Raveena, whose oblivion about the research made it all possible. Finally, the author would like to dedicate this thesis to his parents whose confidence and expectations have been a constant source of encouragement throughout the work.

TABLE OF CONTENTS

LIST OF FIGURES

Figures **Page**

LIST OF TABLES

NOMENCLATURE

The following notations are used in this research:

i = Job number i (i = 1, 2, ..., n)

k = Machine k (k = 1, 2, ..., m)

t = Time

J_i = Number of operations of job i

D_i = Demand of part i (i = 0 to n)

M_i = Number of machine choice for each operation of a part (sum of machine choices for each operation, (i = 0 to n)

N_i = Total number of operations in each part, (i = 0 to n)

T_i = Processing time of each part (sum of processing times of each operation), (i = 0 to n)

P_i = M_i, N_i or T_i (based on the batching policy used), (i = 0 to n)

Q_i = Sub–batch quantity

O_{ij} = Operation number j of job i (O_{i1}, O_{i2}, ..., O_{iJi})

r_{ij} = Ready time of operation j of job i

S_{ij} = Starting time of operation O_{ij}

C_{ij} = Completion time of operation O_{ij}

t_{ijk} = Processing time of operation O_{ij} on machine k

S = A set containing a partial schedule of all operations already scheduled

S' = A set containing all unscheduled operations

S'' = A set containing all schedulable operations ($r_{ij} \leq t$), where a schedulable operation is an operation whose preceding operations are completed.

a_k = Release time of machine k (time at which machine k becomes available)

Min_allowed_qty = Minimum allowed quanity in any sub–batch

CHAPTER 1

INTRODUCTION

At present, most industries are confronted with persistent customer demands for a wider variety of products, faster production rate, shorter delivery times and higher delivery reliability. Many economists, manufacturing engineers, and managers believe steps need to be taken to increase productivity through the use of automated product design and automated production facilities. The importance of product and process improvements is well recognized because they show good results in economic gains. Also, productivity is enhanced through improved machine utilization, reduced work–in–process inventory and effective use of material. Performance improvement is gained through improved procedures for shop planning and control procedures. The general shop control problem for day–to–day operations is a scheduling problem, where specified jobs are assigned to particular resources for accomplishment at a particular time. Scheduling is considered to be the most difficult problem in manufacturing because of its NP–completeness. Over the years, researchers have tried to solve this problem optimally, and have discovered that optimal analysis of scheduling is very difficult in practice. The growing interest in genetic algorithms has led some production researchers to advocate these methods to manufacturing scheduling problems. The approach presented in this research is generic and can be applied to any type of manufacturing system. Manufacturing systems can be classified according to the production volume and are described in the next section.

1.1 Manufacturing Systems

Generally speaking, a manufacturing system is a collection of material, manpower, money and management that goes into the manufacture of a product. Although there are differences among particular manufacturing systems, the basic system framework needed for effective scheduling is common to all manufacturing industries. These systems differ from each other in terms of production volume, production variety, labour skill, special tooling, etc.

1.1.1 Job Shop Manufacturing Systems

The batch sizes in these systems are small (often one of a kind), characterizing them as low volume manufacturing systems. These are usually used to meet specific customer orders, and there is a high variety in the product type. Because of the low volume and high variety, the flexibility of production equipment and labour skill is relatively high.

1.1.2 Flexible Manufacturing Systems

In above systems the flexibility to manufacture a wide range of products in short times has been achieved at the expense of manufacturing efficiency. Automated manufacturing systems, such as flexible manufacturing systems (FMSs) have been developed in recent years to provide the flexibility of job shops and nearly the efficiency of mass production systems. A Flexible manufacturing system (FMS) is one that can be defined as a group of numerically controlled machine tools or similar other automated or semi–automated workstations that are interconnected by automated material handling systems [Groover, 1987]. These systems are computer controlled and can efficiently manufacture more than one part type at low to medium volumes. They are designed to fill the gap between high production transfer lines (mass production systems) and low volume job shops, thus providing an opportunity to meet the current demands of the manufacturing systems which call for flexibility, quality of products and cost effectiveness simultaneously.

1.1.3 Transfer Lines (Mass Production Manufacturing Systems)

This is the continuous specialized production of identical products. It is characterized by very high production rates, dedicated equipment to the manufacture of a particular product, and very high demand rates for the product. The entire plant is often designed for the exclusive purpose of producing a particular product. The equipment is special purpose rather than general purpose. The skill level of labor in mass production systems tends to be lower than in flexible or job shop manufacturing systems. The production ranges of these systems overlap to some degree and it is difficult to draw a line to separate the three types. Figure 1.1 illustrates the relative position of these systems in terms of manufacturing flexibility and productivity.

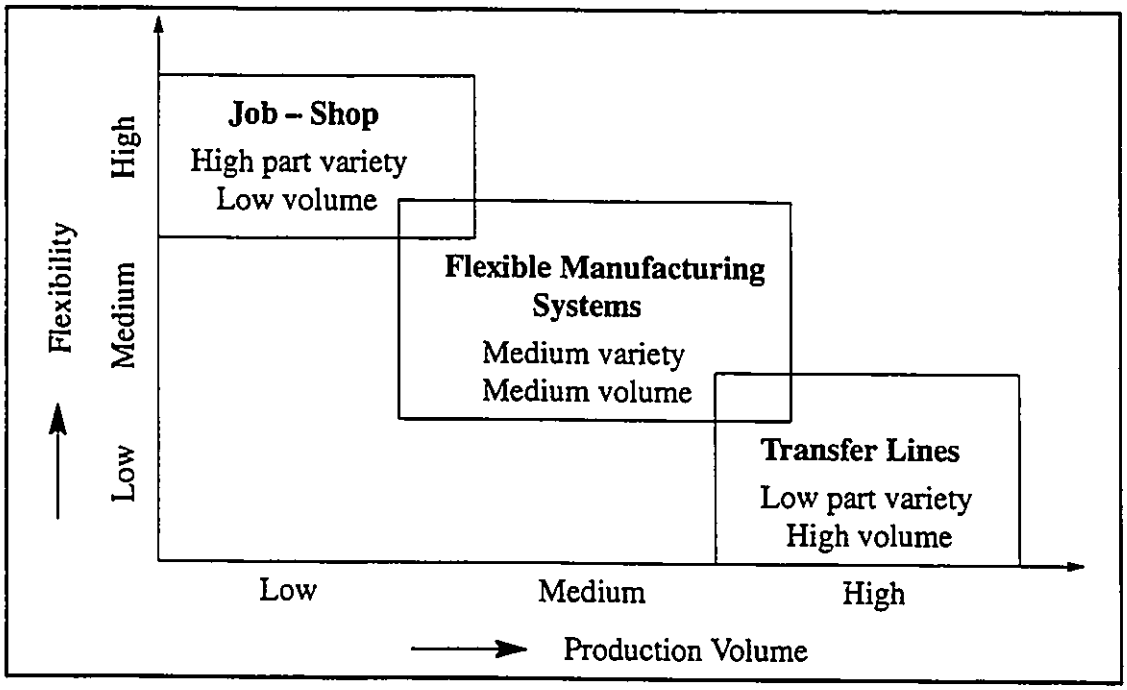


Figure 1.1 Types of manufacturing systems

1.2 Production Planning and Scheduling

The efficient operation of a manufacturing system depends upon its physical components as well as the effectiveness of the planning and scheduling problems.

Production planning problems are concerned with those decisions that have to be made before the production system can actually begin to produce parts. Production scheduling problems are concerned with the decisions regarding the running of the manufacturing system during real time, once it has been set up during the planning stage.

1.2.1 Production Planning

In general, the term 'planning' is not well defined in literature, and is being used to describe a broad range of actions. There are several classification schemes that exist to date. Verbraeck [1991] states the classification of planning as shown in Table 1.1. The classification is clearly hierarchical, and the different levels are dependent. It is important to note that upper level planning acts as a constraint on lower levels, but lower level planning has to provide the feedback which alone can keep the high level aims realistic. Also, a strategic plan is concerned with the whole organization, whereas a detailed production plan affects only a small part such as shop floor activities.

1.2.2 Scheduling

Scheduling is concerned with the lowest level of the planning hierarchy, i.e. detailed production planning. It is related to the assignment of specific resources to the jobs at specific period of times. Several researchers also called this level operations scheduling. Few authors, such as Conway et al. [1967] and Baker [1974], distinguish between allocation (deciding which resources will be allocated to perform each task) and sequencing (determining when each task will be performed) as subclasses of scheduling.

The production scheduling problem is categorized according to the following schemes:

(i) *Requirements generation,* which distinguishes the manufacturing system as an open or closed shop. In an open shop, all production orders are generated by customer requests and no inventory is stocked; in a closed shop, all customer requests are serviced from

inventory.

Table 1.1: Hierarchy of planning, [Verbraeck, 1991]

LEVEL	PERIOD	FREQUENCY	DECISIONS
Strategic Planning	5–10 years	Yearly	• Change of mission • Change of basic activities
Long–term Planning	4–10 years	Yearly	• New products • New investments • Development of financial structure
Medium–term (Tactical) Planning	1–5 years	Monthly/ Quarterly	• Production and sales per division • General coordination • Change in capacity of men and machines • Seasonal inventory levels
Short–term Production Planning	1–12 weeks	Weekly	• Production level per product • Inventory levels • General production schedule • Assignment of men and machines
Detailed Production Planning	Hours/Days	Weekly/Daily	• Assignment of work to specific machines and employees at a specific time

(ii) *Processing complexity,* which is concerned with the number of processing steps associated with each production task or item. A breakdown of this is (a) single machine, single stage, (b) parallel machines, single stage, (c) multistage, flow shop, and (d) multistage, job shop. The single machine, single stage is concerned with one operation per job which can be performed on only one machine. The parallel machine, single stage is concerned with still one operation per job which has a choice of parallel machines. The multistage problem refers to more than one operation per job, which in the flow shop case has the strict uni–directional precedence ordering and in the job shop case (where we have a choice of individual machines) the ordering of process is not uni–directional.

(iii) *Data variability,* which is concerned with the data involved i.e. whether the data

(processing times, sequences, technological constraints, etc.) are deterministic or stochastic. (iv) *Data time dependence,* whether the problem is static or dynamic. The problem is static if none of the initial data change over time (e.g., when all the jobs that are to be considered are available simultaneously at the beginning of the scheduling period). It is considered dynamic if the data change with time (e.g., machine breakdown, intermittent job arrivals etc.).

1.3 Computational Complexities in Manufacturing Scheduling

In principle, there are infinite number of feasible schedules for any job shop problem, because an arbitrary amount of idle time can be inserted at any machine between adjacent pair of operations. However, it is always possible to produce a unique schedule, in which, without changing the sequence of operations on the machines, the idle times between operations are minimized by shifting the start times of operations to the earliest point in time. Such a schedule is termed a semi–active schedule [Baker, 1974]. If optimal solutions are sought, then it is necessary only to consider semi–active schedules since they dominate all others [Wu, 1987]. The exact number of semi–active schedules are difficult to determine; however, the complete set of such schedules can be generated by considering every possible ordering of the jobs. Thus for a classical job shop problem, in which each job has exactly one operation on each machine and each machine must process n operations, if the sequences on each machine are entirely independent, there would be $(n!)^m$ semi–active schedules. However, the actual number of semi–active schedules is usually smaller because the sequence and technological constraints may sometimes make part of the schedules infeasible. Even then, the number of schedules may still be quite large. For example, if n = 4 jobs, then for m = 1, 3, and 5 machines, the number of semi–active schedules $(n!)^m$ is 24, 13,824, and 191 million, respectively. The explosion of semi–active schedules that

results with increasing problem size makes it impossible to use full enumeration procedures to generate optimal schedules, even with the aid of high speed computers [King, 1976].

For most of the combinatorial type problems such as scheduling, the fundamental question at issue is the number of iterations required to find an optimal solution. [King and Spachis 1980]. Work by Karp [1975] on the theory of computational complexity proved that it is unlikely that 'efficient' algorithms will ever be found for these discrete optimization problems. The efficiency of an algorithm can be defined by the computer time required to achieve a solution. Two types of algorithms are defined by Wu [1987]:

(i) Polynomial algorithm (P–Class problems), where the computer time of operation is a polynomial in the size of problem input (i.e., if the length of the problem input is L, the computer time is proportional to L^K, K is the power of the polynomial).

(ii) Non–deterministic polynomial algorithm (NP–class problems), in which the computer time is proportional to K^L.

In their basic form, scheduling problems belong to a special class of 'hard' problems, the so–called NP–hard or NP–complete problems [French, 1982] for which no polynomial time algorithm has been found. The algorithms in this class mostly have an exponential time behavior. No fast solution method exists for NP–hard problems. This means that for large scheduling problems, no optimal solutions can be computed in a reasonable amount of time. Therefore, we have to be satisfied with a schedule that is not optimal but the best possible for a given situation. This best possible schedule will suffice, given the time, knowledge, and experience available. This means that the problem of scheduling is 'satisficing' rather than 'optimizing'.

Because of the NP–completeness of the scheduling problem, researchers have used several different approaches such as (a) analytical techniques such as branch and bound, dynamic programming, and integer programming, (b) simulation techniques employing dispatching rules, and (c) artificial intelligence and expert systems techniques. Because of

the resulting inefficiency of the algorithms, the size of the problem always reaches a limit at which the computer time required becomes excessive. This limit is unfortunately reached very quickly with relatively small-sized problems [King and Spachis, 1980].

1.4 Research Effort and Organization of the Thesis

This research explores the feasibility and advantages of using genetic algorithms (GAs) to solve production scheduling problems. Genetic algorithms are optimization techniques that have already been applied with advantage to a variety of combinatorial problems. The main drawback of the existing solution approaches is the exponentially increasing search space in which the optimal solution is sought. Genetic algorithms alleviate this drawback and are based on using a fixed search space for finding possible optimal solutions. The major issues addressed in this research are:

(i) Scheduling when there are no alternate process plans available,

(ii) Scheduling in the presence of alternate process plans,

(iii) Batch splitting considerations, and

(iv) Dynamic scheduling

This thesis is organized as follows:

Chapter two reviews the literature on various approaches used for scheduling and lists the advantages and drawbacks of these approaches. It also briefly reviews the literature on batch splitting and dynamic scheduling. In addition, the motivation for the current research and the research objectives are described.

Chapter three introduces genetic algorithms and discusses their various components in detail. It presents the basic structure of genetic algorithms and genetic operators. It also presents and compares two types of genetic algorithms that are usually employed for scheduling.

Chapter four presents the use of genetic algorithms when there is only one process plan

per part available. First, the application of genetic algorithms in scheduling is discussed and then the use of genetic algorithms in scheduling, schedule representation, genetic operators and scheduling procedure is further explained in detail with the help of a numerical example. The proposed algorithm is tested on several examples obtained from the literature and results are discussed.

Chapter five presents the use of genetic algorithms in the case of multiple routing flexibility, i.e., scheduling in the case of multiple process plans. It discusses the genetic algorithm model used and working of the genetic operator. The scheduling procedure is illustrated with the help of a numerical example.

Chapter six presents batch splitting considerations for scheduling the orders. In this chapter, three new batch splitting policies are developed and are compared with three other existing policies. To release the manufacturing orders into the shop floor, three different order regeneration methods are used in this research. Use of genetic algorithms, consideration of batch splitting policies and order regeneration methods are illustrated in detail with the help of a numerical example.

Chapter seven considers dynamic scheduling in manufacturing. It first discusses different types of uncertainties and then presents rescheduling algorithms for these uncertainties. The application of these algorithms is discussed with the help of an example.

Chapter eight discusses the system that has been implemented to integrate different modules and to make the scheduling process easier. This chapter describes the system structure, hardware and software used, user interface, scheduler, analyzer and rescheduler.

Chapter nine presents the summary and conclusions of this research. Research contributions and achievements are also discussed in detail. Finally, the chapter discusses the limitations of this work and highlights the areas which should be addressed in future to overcome these limitations.

CHAPTER 2

LITERATURE REVIEW

Manufacturing scheduling has been an area of extensive and intense research over the past few decades. Research in manufacturing scheduling started primarily with simple analytical techniques such as integer programming and expanded to new solution approaches such as artificial intelligence and genetic algorithms. The vast amount of literature available to date on various scheduling approaches proves that there is no single approach that can be used successfully in all scheduling situations. In this chapter, various solution approaches used for scheduling in literature are reviewed and their advantages and drawbacks are discussed. In addition, motivations for the proposed research and objectives are delineated.

2.1 Production Scheduling Approaches

In this section, solution approaches to production scheduling are divided into four main categories, namely, (a) analytical, (b) simulation, (c) artificial intelligence and expert systems, and (d) genetic algorithms techniques. There are several published articles that use combinations of the above mentioned techniques.

2.1.1 Analytical Techniques

Various analytical techniques have been used in the past for manufacturing scheduling where the formulations have been restricted to simple problems with not more than ten tasks and ten machines/workcenters. Perhaps the research in this area is as old as the theory of

scheduling. One of the earliest and most popular approaches to the scheduling problem was developed by Hanry L. Gantt. This graphical approach, known as the Gantt chart, is a bar chart displaying job operations over time for each machine/workcenter. Analytical techniques including heuristic methods as well as operations research techniques such as integer linear programming, dynamic programming, branch and bound technique, etc., have been used to obtain optimal schedules.

Dynamic programming is a general purpose method for implicitly enumerating the entire search space. It operates by breaking the problem down into stages and searching for an optimal solution at each stage. Although it is an effective technique for small size problems, its computational demand grows exponentially with problem size, making it impractical for large problems. This deficiency is somewhat alleviated by an alternate operations research technique, a branch and bound technique, which is a more general–purpose search technique that has been successfully used for a variety of problems. It operates by simultaneously maintaining many partial paths down the search tree. Associated with each partial path is a lower bound 'L' on the cost of a solution through that path. The next path chosen for extension is always the one with the lowest value of 'L'. This allows us to follow those paths that have the greatest potential for giving the best solutions. Nodes in the search graph can be viewed as scheduling decisions, while paths represent partial or complete schedules. Baker [1974] notes that branch and bound is a promising strategy in many scheduling problems because it is often possible to calculate very strong lower bounds for partial paths. In short, branch and bound is one of the few effective general–purpose methods for solving large combinatorial problems such as scheduling.

The results of the research done in this area are so vast that there are several review articles of varying breadths and depths which survey the development of scheduling theory. Some of the articles are Bakhshi and Arora [1969], Panwalkar and Iskander [1977], Graves [1981], Raman [1985], Zanakis et al. [1989], Rachamadugu and Stecke [1993], and

Maccarthy and Liu [1993]. Graves [1981] provides a review of production scheduling theory and practice focusing entirely on analytical results from deterministic and static models in both open and closed shops under the various processing environments. Sen and Gupta [1984] review the research in the same category but focus entirely on static scheduling problems whose performance measures bear in some way on job due dates. Zanakis et al. [1989] provides an extensive survey of 442 articles that categorize them with respect to heuristic and operations research methods.

Most of the optimization techniques considered through analytical techniques resulted in NP–completeness [Maccarthy and Liu, 1993], i.e. for large–sized problems no optimal solution could be found in reasonable amount of time. Also, NP–completeness results have forced researchers to look for exploitable problem characteristics in order to be able to generate efficient solutions. This calls for the development of strong bounds and good heuristics. Another drawback of this approach is that because of the nature of the formulation, this approach does not contain all the details of the problem being modeled.

2.1.2 Simulation

As mentioned earlier, a main criticism of analytical scheduling theory is its inability to deal adequately with the complexities of the real problems that arise in practice. Simulation can be a powerful tool, both in analyzing a scheduling domain and testing a scheduling system. Indeed, in problems such as dynamic job–shop scheduling, simulation may be an indispensable tool. In complex environments, plant personnel may not have information that is necessary in designing a scheduling system. Depending on the amount of information that is built into a particular simulation model, simulation has the potential to be the most flexible model, allowing as much detail as desired or necessary to mimic reality [Stecke, 1986]. Since scheduling algorithms often employ heuristics, their designers cannot account for all the possible dynamics of the plant. To determine bottlenecks and other

problem areas and to understand the dynamics of the plant, it may be necessary to run many hours of plant simulation. The information gained from such a simulation can have a profound effect on the scheduling strategies subsequently employed. Such a simulation may even be useful as part of the scheduler. Most scheduling systems must make predictions about the future in order to make the best scheduling decisions. The ability to actually use the simulator to make these predictions could then result in more accurate predictions and thus better schedules. Usually, a simulation is tailor–made for the particular application at hand.

Simulation techniques are basically concerned with trying the various dispatching rules available on the manufacturing system, and selecting the most appropriate for a given situation. Dispatching rules are also referred as heuristic rules, scheduling rules, or priority rules. There are so many dispatching rules available to date that several researchers have written articles which delineate just the dispatching rules [Blackstone et al. 1982, Montazeri et al. 1990, Panwalkar et al., 1977]. Others have either used these dispatching rules or they have developed their own for the evaluation of manufacturing system. A good survey of simulation research has recently been published by Ramashesh [1990]. In this section the work done by some researchers in evaluating scheduling rules with different performance measures is reviewed.

2.1.2.1 Dispatching Rules and Their Classification

Basically, a scheduling rule is used to select the next job to be processed from a set of jobs waiting for service at the corresponding machine queue. These rules can be used to introduce workpieces into the system, to route jobs in the system and to assign jobs to facilities. Scheduling rules can be static, i.e. they can be applied at the beginning of the scheduling period and result in a fixed schedule for the period, or dynamic, i.e. changing over time. They can be very simple or extremely complex and may be classified in different

ways according to their specific attributes. The performance of the scheduling rules depends on the performance criterion chosen as well as on the configuration of the manufacturing system. The simulation results for the case study done [Montazeri et al. 1990] shows that scheduling rules can have a large impact on performance measures in automated systems, largely because all components of such systems are tightly interconnected. Hence, in order to evaluate the effect of scheduling rules on an automated manufacturing system, one has to carefully select a suitable performance measure and then simulate the effect of various scheduling rules on that criterion. It is known that scheduling is a complex problem; it is not only difficult to describe by means of analytical techniques, but there are no other practical methods for scheduling [King et al., 1988]. Since this is a fertile area of research, many dispatching rules have been proposed and compared. The first survey of scheduling rules was performed by Panwalkar and Iskander [1977] where 113 different dispatching rules were categorized and listed based on a survey of 36 publications. Baker [1974], Blackstone et al. [1982] and Wu [1987] have done an excellent classification of scheduling rules. Their efforts are summarized in Table 2.1 and Table 2.2.

Table 2.1: Dispatching rules and their classification [Wu, 1987]

CLASSIFICATION	DISPATCHING RULES
Processing–time based	SI, SPT, LI, LPT, SRPT, LRPT
Due–date based	EDD, OPNDD
Number–of–operations based	FOPNR, MOPNR
Setup–time based	NSUT, SIMSET
Machine–attribute based	NINQ, WINQ
Combination of simple rules	Truncated SPT, Truncated SI
Others	FCFS, RANDOM, SLACK, SLACK/OPN

Table 2.2: Dispatching rules definition

RULE	DEFINITION
SI	Shortest imminent task processing time
SPT	Shortest job processing time
LI	Longest imminent task processing time
LPT	Longest job processing time
SRPT	Shortest remaining job processing time
LRPT	Longest remaining job processing time
EDD	Earliest due date
OPNDD	Earliest operation due date
FOPNR	Fewest operations remaining
MOPNR	Most operations remaining
NSUT	No setup time
SIMSET	Shortest setup time
NINQ	Select the job whose next operation is on the machine with shortest queue
WINQ	Select the job whose next operation is on the machine with least work
T–SPT	Select jobs based on SPT; for jobs whose waiting time is greater than a specified value, use FIFO
T–SI	Select jobs based on SI; for jobs whose waiting time is greater than a specified value, use FIFO
FCFS	First come first served
SLACK	Least remaining slack time (Due date – present time – remaining time)
SLACK/ OPN	Smallest ratio of slack time to number of remaining operations

As mentioned earlier, an analytical approach to scheduling problems has proved to be extremely difficult, even with several limiting assumptions. In the face of the difficulties associated with analytical techniques, researchers in this area have relied on computer simulation of real or representative shops to study the scheduling problem. Simulation typically is not used as a means for generating production schedules *de novo*. Instead, simulation is applied as a vehicle for testing schedules or comparing the performance of alternative schedules, which have been developed using some other logic and based on some

simpler model of the production facility. Simulation similarly is also applied as a vehicle for testing and comparing the performance of alternative scheduling heuristics and dispatching rules. In this section, literature pertaining to simulation is reviewed.

Conway [1965a, b] is among the first researchers who analyzed a large number of dispatching rules. He considers a job shop with nine machine groups each with a single machine. His criteria for comparison are various measures of work–in–process inventory and job lateness. He measures two attributes for each job, shop residence time and lateness. He uses 16 priority rules (e.g. SI, LI, FIFO, etc.) and concludes that in both cases, i.e. in minimizing work–in–process inventory and average job lateness, SI rule gives the best performance. He also conducts a study on due–date based criteria and in this case he finds the SLACK/RO dispatching rule is superior to other rules. He also finds that in general, the performance of all the rules deteriorated under increased workloads.

Hershauer and Ebert [1975] test three due date–based rules versus three processing time–based rules and seven combined rules. They conclude that:

(i) the SI priority rule minimizes mean flowtime.

(ii) all due–date–based rules perform better than SI with respect to cost per order,

(iii) SLACK/OPN performs better than other due–date–based rules.

Dar–El and Wysk [1982] test six priority rules in a conventional job shop. A systematic analysis of shop performance versus different scheduling rules is presented using a computer simulation model of a conventional job shop. The simulation model is used to identify dependent parameters such as shop load and service distribution for a set of scheduling rules. The performance measure they consider are minimum job tardiness and root mean square (RMS) tardiness. They conclude that when overall ranking is considered, the SI priority rule performs best for average tardiness and slips into fourth place behind WINQ (work in next queue), FIFO and SLACK/RO for root mean square tardiness performance.

Ballakur and Steudal [1984] compare several dispatching rules in their state–of–the–art review of job shop control systems. This study indicates that: (i) the SI rule appears to be the best dispatching rule for relatively loose due dates and moderate machine utilization, (ii) SLACK/RO consistently outperforms other due–date–based rules, and (iii) combined rules involving SLACK and SI are most promising and merit further research.

Stecke and Solberg [1981] investigate an actual FMS, which consists of nine machines, an inspection station and a control queueing area connected by a material handling system. The operating strategies considered involve policies for loading (allocating operations and tooling to machines) and real–time flow control. The number of completed parts is used as the performance measure of the system and sixteen priority rules are tested with this measure. Six of these rules are simple priority rules obtained from literature and the others are the combination of some simple priority rules. Their results indicate that for the selected FMS, the SPT priority rule performs better than other rules. In this case, the SI rule performs below average. It was concluded that the choice of applicable loading and scheduling strategies depends on many particular system variables and the best combination of different rules is highly system dependent.

Blackstone et al. [1982] discusses the state of the art in the study of dispatching rules, which includes analytical approaches, simulation techniques, bias of estimates produced by simulation, sample size, and evaluation criteria. While comparing several dispatching rules they report that SI is the best priority rule when: (i) the shop has no control over due dates, or (ii) the shop has control over due dates and due dates are tight (are set at six or less times total processing times), or (iii) the shop has control over due dates and due dates are loose (are set at seven or more times total processing times) and there is great congestion in the shop (machine utilization approaches 95%). They conclude that it is impossible to identify any single rule as the best in all circumstances.

ElMaraghy [1982] considers a system of five machines, one load/unload station, and

two material handling devices and tests four priority rules. The author concludes that the SI rule yields the highest production rate in terms of total number of parts produced, as well as total processing time of all stations. SI also reduces average flowtime in comparison with other scheduling rules tested.

Montazeri and Wassenhove [1990] conduct an experiment on an actual FMS that had three machine families, three load/unload stations, three carriers, and eleven work–in–process buffer positions. Fourteen different scheduling rules are tested for the system chosen with the modular FMS simulator. The rules are based on shortest processing time, longest processing time, random selection, and a combination of longest processing time with total processing time. The criteria chosen in this study are average and variance of waiting time per part, average and variance of machine utilization, average buffer utilization, average shuttle utilization, average carrier utilization, and makespan. From the experimental results they conclude that no single scheduling rule is the winner on all performance measures and it is up to the user to choose one or more of the priority rules according to the performance measures prevailing in a particular application.

Chandra and Talavage [1991] develop a decision rule for real time dispatching of parts, each of which may have alternative processing capabilities. This decision rule is tested in a simulated flexible manufacturing system. In this study, they considered a general queue where the part is sent after the completion of its operation. Hence, a machine had a global option for selecting the part from this queue. Rather than using conventional dispatching rules, an intelligent part–selection strategy is developed that takes into account the current state and trends of the system. The basic information used for intelligent dispatching in real time are: (i) shop congestion level, (ii) preference for a part by machine, (iii) criticality of the parts, and (iv) current shop objective. The shop objectives considered are maximizing work progress rate and minimizing the number of tardy jobs. The dispatching rule developed in their study is called EXPERT. Expert is compared with SPT,

EDD, LSPO (least slack per operation), and LRS (least relative slack). The proposed intelligent reasoning procedure was found to achieve better shop performance than some of the popular dispatching rules, the improved performance being due to the ability to respond to changing circumstances.

Wu and Wysk [1989] propose a multi–pass scheduling algorithm for the dynamic scheduling of flexible manufacturing cells. Here, a dispatching rule is selected in each short time period by discrete event simulation. They conclude that the multi–pass scheduling algorithm performs better than the single–pass scheduling algorithm which used a single dispatching rule for the entire manufacturing period. In their paper, they indicate that the length of the scheduling interval is a significant factor for the performance of multi–pass scheduling algorithm. Although they discuss this problem from several aspects, no general procedure to define the scheduling interval is proposed. They use the multiple (i.e. one, two, three times) of average total processing time and find that triple the average total processing time is the best scheduling interval length.

Gupta et al. [1993] propose an integrated simulation based approach to the operations planning and scheduling (OPS) problem. They develop a detailed simulation model using FORTRAN and SLAM II which integrates loading, part inputting, routing and dispatching issues of OPS. An experimental framework is developed that provides statistical analysis of the simulation output by developing an experimental design. Several hypotheses concerning a number of system parameters are developed and statistical analysis are performed on a set of performance measures such as mean flowtime, mean tardiness, system utilization and maximum tardiness. They use four rules, namely SPT, EDD, WINQ and CRATIO and found that SPT and EDD rules perform better than than WINQ and CRATIO dispatching rules.

In short, the simulation approach has been reasonably effective in studying the dynamics of the manufacturing system. The simulation studies compare specific operating

procedures, test broad conjectures about scheduling rules and develop greater insight into the nature of a plant operation. The simulation models can be developed using some existing general purpose simulation languages such as SLAM, SIMAN, GASP IV, or SIMSCRIPT. Recently, many special purpose manufacturing system simulation languages have been developed that have various types of graphics capabilities. In addition to providing a realistic model for the evaluation of schedules and heuristics, simulation also provides a limited capability for testing modifications to an existing schedule. Since the simulation process provides dynamic output, the user can identify long queues within the model and attempt to eliminate the bottlenecks that may exist for priority jobs on the floor [White 1990].

An advantage of the simulation approach to scheduling is that it can model the effects of such factors as policy changes, which cannot be accounted for in an analytical model [Stecke and Solberg, 1981]. Another advantage is that it can provide the user with the opportunity of performing exploratory tests upon the schedules being produced. There are several drawbacks of using the simulation approach. Simulation can potentially be expensive and time consuming to develop, debug and run. The accuracy of any simulation model is limited by the judgement and skill of the programmer [Rodammer and White 1988] and even highly accurate modeling does not guarantee that optimal or even good schedules will be found experimentally. Any errors in the simulation program could lead to false conclusions. The other drawback is the large amount of time this approach takes to reach to an optimal solution because of its experimental nature. Because of the large amount of time required to find an optimal solution, this approach is good for off–line scheduling. In case of unexpected events on the shop floor, rescheduling, and therefore more computational time, is required to obtain an optimal solution. The basic problems with these dispatching rules is that they usually only handle a single measure of performance, and their locally

greedy strategies ignore the possibility for more global optimizations such as alternate routings [Fox, 1987].

2.1.3 Artificial Intelligence and Expert Systems

Most recently, artificial intelligence (AI) or knowledge–based techniques have been used (with limited success) to solve scheduling problems. AI research has been largely devoted to the development of more sophisticated and efficient search techniques and the representation and use of heuristic reasoning. They typically depict the scheduling problem as the determination and satisfaction of the large number and variety of constraints which are found in the scheduling domain. While both operations research and AI are concerned with search techniques, AI puts more of an emphasis on the knowledge that guides the search. AI is used to extend knowledge representation techniques to capture the scheduling constraints, to integrate constraints into a search process, to relax constraints when a conflict occurs, and to diagnose poor solutions to the scheduling problem. These methods employ heuristic rules to guide the search and may offer efficient search procedures for finding good solutions to computationally complex problems [Davis, 1985].

One of the most important areas of AI that has successfully been used to solve scheduling problems is expert systems. Expert systems appear to be appropriate for the scheduling problem because of their heuristic nature, i.e., they require the use of rules of thumb to achieve acceptable solutions. Expert systems are generally built for a particular production facility, and because each production facility is different, the expert systems approach may not be sufficiently robust to handle new production and scheduling situations [Kanet et al., 1987].

There are other drawbacks of using expert systems as well. First, expert systems are expensive and time–consuming to develop. The costs of developing expert schedulers tailored to specific production environments may be prohibitive. Second, expert systems for

reasonably sized problems may result in very slow computational speeds [Jackson and Jones, 1987]. Finally, expert systems strive to automate decisions that are made by human experts. To reach this goal, it is necessary to capture the high level of expertise of individuals currently involved in the solution of scheduling problems. Unfortunately, the quality of human performance in many scheduling tasks is suspect and frequently expert human schedulers may not always exist in the production scheduling environment.

There has been a large body of literature generated that employs expert systems for production scheduling. Review articles in this area are given by Steffen [1986], Kusiak and Chen [1988], Atabakhsh [1991], Smith [1991], and Noronha and Sarma [1991]. Steffen [1986] provides a survey of artificial intelligence based scheduling systems in use until 1986 and discusses his survey through historical, methodological, application and implementation perspectives. Kusiak and Chen [1988] review the research and applications of expert systems in production planning and scheduling and also explore the relationship between expert systems and operations research approaches. Atabakhsh [1991] reviews various AI techniques used in constraint–based scheduling, by comparing existing systems. Smith [1991] provides an overview of research in the field of knowledge–based production management. Finally, Noronha and Sarma [1991] give an excellent survey of articles and scheduling systems using knowledge–based approaches.

Apart from analytical, simulation and artificial intelligent approaches there has been a new area of research based on randomization procedures and selection that has gained a lot of attention recently for solving complex problems such as scheduling. Genetic algorithms (GAs) fall into this category and have been used successfully in providing optimal or near–optimal solutions. Detailed discussions about genetic algorithms is deferred until the next chapter.

2.1.4 Genetic Algorithms

Researchers are continuously making serious effort to reduce the gap between scheduling theory and practice and as a result much of the previous research effort has been spent developing more powerful algorithms and more effective heuristics for manufacturing scheduling problems. We have discussed the advantages and disadvantages of various scheduling approaches and it was observed that there is a need for better scheduling algorithms (ones that can give better and faster solutions). Genetic algorithms are optimization algorithms that have been applied successfully to many NP–complete optimization problems. They use randomized selection and recombination of solutions to evolve a good solution [Hsiao–Lan, Ross and Corne, 1993]. These algorithms are relatively faster than other optimization techniques as less time is required to search the fixed space of possible solutions. They can be used in real–time, where quick decisions are required, because of their fixed search space.

There is little literature available on the application of GAs in scheduling since it was first introduced by Davis [1985]. This paper was instructive rather than realistic, in that he showed how genetic algorithms can be used to solve a simple job shop scheduling problem. The example discussed by Davis is much simpler than those one would encounter in real life, and he suggests that the range of operations employed would have to be widely expanded if a realistic example is to be approached. The schedule is represented as a list of operations to be performed. The problem of infeasible schedules being created is reduced with the help of a decoder that always yields feasible solutions to the problem.

Liepins, Hilliard, Palmer and Morrow [1987] investigate the simplest scheduling problem, i.e. a static queue of jobs with specified due dates and run times without precedence constraints, with a single server, and which used minimal lateness as criterion. Because of the representation of the problem, a conventional crossover operator could not be used as it would create infeasible schedules. They investigate three different crossover operators for job shop scheduling: PMX, a weak greedy crossover, and a powerful greedy crossover. They

conclude that in fact a choice of crossover operators would make a difference in the scheduling results. For their experiments, strong greedy crossover operator dominated PMX, which in turn dominated the weak greedy crossover.

In another paper by Falkenauer and Bouffouix [1991], a genetic algorithm is applied and tested successfully on small, medium and large job shop problems. They introduce suitable encoding of solutions of the problem together with a new crossover operator exploiting the encoding and a cost function to estimate the quality of solutions. The two crossover operators they used are PMX (partially matched crossover) and LOX (linear order crossover). However, because of the encoding they chose, the mutation operator could not be applied. It is considered here that each operation could be done only on one machine. They compare the performance of their genetic algorithms with the two most widely used scheduling heuristics, namely LST (least slack time) and SPT (shortest processing time). Their experiments reveal the superiority of the genetic algorithms approach over the common scheduling heuristics and a slight superiority of the LOX operator over the PMX operator.

Gupta et al. [1993] address an n–job, single machine problem with an objective to minimize the flow time variance. They propose a heuristic procedure based on genetic algorithms with the potential to address more generalized objective function such as weighted flow time variance. One important issue of genetic algorithms, i.e., parameter selection problems, is addressed and some guidelines are developed using an experimental design approach. They also use PMX (partially matched crossover) operator for their experiments and mention investigating the performance of other crossover operators. They generate a data set of ten problems and find optimal solutions for all of them using genetic algorithms with very simple modification.

Nakano and Yamada [1991] use a conventional genetic algorithm to solve a job shop scheduling problem. They introduce three unique ideas in representation, evaluation, and

survival. As opposed to string representation, they use binary representation which made it possible for them to use conventional genetic algorithms. Even though a chromosome produced by conventional crossover is usually illegal, the evaluation method presented here could evaluate it by finding a similar legal chromosome. To help chromosomes survive, they introduced a new treatment, called forcing, that replaces illegal chromosome with a legal chromosome. It is shown that use of forcing improves convergence rates and solution quality. Experiments using well known job shop benchmark problems showed the solutions generated by the present approach are as good as those obtained by branch and bound methods.

All of the above studies applying genetic algorithms suffer from major drawbacks since their experiments did not consider certain important scheduling constraints such as precedence among tasks, alternative process plans, and application of dispatching rules for multiple machine assignment, which are major factors contributing to the difficulty of scheduling problems.

Recently, Uckun, Bagchi, Kawamura and Miyabe [1991] addressed the problem of alternative process plans and developed a new two chromosome representation where each chromosome represents a complete schedule. They discussed three different representational schemes, namely direct, indirect and problem specific representation. To enhance the performance of the algorithm and to expand the search space, a chromosome representation that stores problem specific information was devised and compared with other two representations. The crossover operator used was PMX (Partially matched crossover). They did not consider due dates and dispatching rules, and the system was tested using only one performance measure, namely machine utilization.

Burns [1993] used this approach to solve a real–world scheduling problem and used the same chromosome representation introduced by Uckun et al. He developed advanced knowledge augmented crossover and mutation operators and schedule evaluation was

performed based on the sum of square lateness. He concluded that the performance of the genetic algorithms for scheduling problems could be improved by the integration of problem–specific knowledge of the application domain in the chromosome structure and the recombination operators. None of the above articles addressed the issue of using different scheduling rules and more than one performance measure for their scheduling system.

The results obtained by the above authors indicate that genetic algorithms present a good scheduling alternative: they are reasonably fast, gradual (i.e., some solution is available immediately) and provide better results than the heuristics (based on the objective function of the problem). This research considers some of the important issues that have not been considered yet, such as use of dispatching rules, while using genetic algorithms in scheduling.

2.2 Batch Splitting Approaches

Batch splitting is the partitioning of the selected part types into batches for simultaneous processing on a machine with a single setup. Larger batches reduce the number of setups and changeovers required during the scheduling horizon, while smaller batches decrease in–process inventories and increase the number of scheduling options. Minimum batch sizes typically are established by management decisions based on the products, technologies, and resources involved. Maximum batch sizes are governed by performance needs and the amount of work to be done. Batch sizing decisions are frequently required for many different products and demand quantities, which creates the need for a practical, efficient decision rule [Sepehri et al., 1986]. Finding the optimal batch size that minimizes the total of production, setup, holding, shortage, and scrap costs, however, is not currently feasible [White, 1990]. Most techniques either cannot guarantee the generation of a feasible solution or are computationally prohibitive [Mclain et al., 1985].

Usually the batch scheduling problem is approached as two independent subproblems

of batch–sizing and scheduling. In the first case, the problem is assumed to be entirely a matter of determining an economic batch size. No regard is given for the possible machine interference that might result when the economic order quantities for each product are derived independently. This research direction is based on inventory models and batch sizes are determined based on the cost function consisting of ordering costs, inventory holding costs and work–in–process carrying costs. In the second case, it is assumed that the batch sizes are given, and the problem is concerned entirely with the scheduling aspects. The importance of meeting due dates in automated manufacturing systems is well recognized by practicing managers and academic researchers. Completion of jobs ahead of due dates would result in storage costs; on the other hand, if the jobs are completed after the due dates there will be tangible costs such as overtime as well as intangible costs such as loss of goodwill, unsatisfied customers etc. This problem has received considerable attention in the literature, but most of the effort has been directed toward the development and evaluation of dispatching rules intended to improve the due date performance. There has been considerable success in solving the economic batch scheduling problem for the single machine case, but the multiple machine scheduling problem remains to be solved because of its NP–completeness.

It has been proved that batch sizing decisions have a significant impact on time–related performance measures such as flow–times and makespan [Dobson et al., 1987]. In many situations, rather than manufacturing only the current requirement for a particular part, it is thought to be economically desirable to group the parts in small batches to achieve high efficiency. A number of models have been developed which directly consider the relationship between batch sizing, flow times and in–process inventory. Effect of batching on due–date–based performance measures is still scarce and there is a need for determining the performance of manufacturing systems when due dates are important.

The effect of batch sizes on processing–time–based performance measures has been

considered by many researchers. Bertrand [1985] studies the effects of batch sizes on the batch flow times in the manufacturing system. He assumes that for each product the processing time of a batch at a workcenter is proportional to the batch size. He considers the manufacturing shop as the queueing system and uses results from queueing theory to show that batch flow times increase as the batch size increases. Karmarkar, Kekre and Kekre [1985] represent the relationship between batch–sizing and shop performance for multi–item, multi–center shops using a queuing network model embedded in an optimization routine that searches for optimal batch sizes. Both of the above batch sizing models assume a FCFS queue discipline and the impact of shop scheduling is ignored, even though shop scheduling is known to have a considerable impact on shop performance.

Karmarkar [1987a] examines the delays incurred in deterministic sequencing problems and qualitatively studies the impact of changes in task length on measures such as makespan and flow times. He examines the deterministic case, and his results suggest that, with job routing and sequencing policies in effect, there is a tradeoff between flow time reduction and throughput in making batch size choices. Dobson et al. [1987] use an integrated formulation for batching and sequencing. The studied formulations are for single machines, which is not representative of real manufacturing systems that typically involve dynamic, multi–machine problems with complicated factors. Afentakis [1985] presents a model that integrates inventory and scheduling aspects of production planning in a multistage environment by transforming a batch–sizing problem to an equivalent job–shop scheduling program. Karmarkar [1987b] explores the impact of batch sizes on manufacturing lead times. The relationships between batch sizing, manufacturing lead times and in–process inventories are formulated by standard queuing models to investigate congestion phenomenon and the resulting effect on waiting times. All of the above research shows that the batching policy has an impact on performance measures such as flow times, waiting times and resource utilizations. Another similar area where batching decisions are

required is scheduling orders in automated storage/retrieval systems. Elsayed and Unal [1989] present four heuristics and analytical model for the order batching problem. The criterion they use is to minimize the total travel time. The algorithms they developed are called EQUAL, SL, MAXSAV and CWright algorithm. All heuristics are based on the time–saving criterion of combining two or more orders in a single tour rather than processing them one order at a time. Among the heuristics presented, the SL algorithm gave the best performance.

Most of the available studies considered batch–sizing and scheduling together with the limitation of one operation per part. The relationships between batch–sizing and due–date–based performance measures were not fully explored in the cases of multiple operations per part, precedence constraints among operations, alternate routings, and variable task processing times. In this thesis, we use a batching policy quite different from the models normally used in queueing theory. The input required in most of the batch sizing models involved information that was not readily available (for example, storage costs and holding costs). In the proposed research, the batch sizing is performed based on the process plans of the part types to be processed. All the information required for the batching decisions can be extracted from the process plans of the parts.

2.3 Dynamic Scheduling

Dynamic scheduling implies that a given schedule is to be revised at specific times due to certain significant changes in operating conditions. The rescheduling procedure is never planned in advance but is brought into action when certain unavoidable circumstances occur. Because of the simplicity of the algorithms proposed in this thesis, it is expected that they can be used for the real–time control of any manufacturing systems, as opposed to rescheduling in production and inventory control, where scheduling is considered over a period of weeks or days.

Rescheduling can be carried out both manually or through an application software. The manual method involves editing the existing schedules which are normally in the form of a Gantt chart. This method is tedious and time–consuming. Researchers have generally used simulation approaches for rescheduling purposes. A good survey of articles involving dynamic job shops scheduling is presented in [Ramashesh 1990]. Yamamoto and Nof [1985] use a 'regeneration' method in developing their scheduling systems. This method involve rescheduling the entire set of operations (or jobs), including those unaffected by the change in conditions, demands, and/or constraints. This is time–consuming, and often results in response times unacceptable to the user. They compare three scheduling procedures to deal with machine breakdown. However, they did not address the problems of other uncertainties such as rush orders, increased priority and order cancellations. Matsuura et al. [1993] use simulation to investigate rescheduling, and approach the problem with a perspective of selection between sequencing and dispatching in case of uncertainties. A schedule is first determined by using a branch and bound technique. This approach is switched to dispatching, which uses either the FCFS or SPT rule, when there is a change in production conditions. Li et al. [1993] propose a rescheduling algorithm based on the construction of a scheduling binary tree and a net change concept adopted from MRP systems. A limitation of the algorithm is that it could only deal with a rescheduling situation that assumes no change in the existing operation sequence of each machine. They did not consider the alternate routings for rescheduling. Dutta [1990] proposes a knowledge–based system to help automate the control activity at the scheduling level in FMS environments. The author assumes a batch size of one, which is rarely the case in actual production systems.

2.4 Motivation for the Proposed Research

The manufacturing scheduling problem has attracted a great deal of research and numerous approaches have been proposed as seen earlier. The development of good

solutions to the manufacturing scheduling problem is confounded by two realities: the combinatorial complexity of the problem and the executional uncertainty of factory operations [Fox and Kempf, 1985]. The problem complexity derives from the need to coordinate the simultaneous execution of many production processes, each a specific temporally ordered set of process steps that require use of shared resources in a manner that tends to optimize the overall performance of the factory. Performance concerns argue strongly for developing schedules in advance, providing the opportunity to anticipate important process interactions and impose execution constraints that minimize their degrading effects on performance objectives. At the same time, the unpredictability of factory operations (e.g. machine breakdown, process step variability, etc.) inevitably forces deviations from prescribed actions, and tends to work against any attempt to exploit pre–optimized schedules. An advance schedule will be of limited use unless the guidance it contains can be continually adapted to current circumstances on the factory floor.

From the past scheduling research it is observed that manufacturing scheduling is focused mainly either on the optimization or on the control part of the problem. Graves [1981] gives an extensive survey of the papers basically concerned with producing optimal solutions under various problem assumptions. These schedules are built before execution and are purely predictive. Once an operation starts (i.e., the schedule been put into effect) no changes can be done in the schedule. If an unpredictable event occurs, then the problem requires a new complete resolution at that time. In this case, it may take a little time to schedule the operations to machines, but frequently rescheduling the whole problem will take a long time and eventually results in lower system performance.

Control–oriented research [Panwalker and Iskander 1977] alternatively focuses on the development of local dispatch priority rules. These approaches can be seen as purely reactive. They are relatively insensitive to surprises, since commitments are only made as necessary to continue execution, and typically require only modest computational resources.

However, overall factory performance is very much a function of the sensitivity of the local dispatching rules to the structure and operating conditions of the factory. Unless the dynamics of the factory are well understood and stable over time, such strict reliance on local decision–making leads to otherwise avoidable congestion and suboptimal performance.

This research explores the feasibility and advantages of using genetic algorithms in manufacturing scheduling. Most manufacturing industries rely on simple facts based on supervisory personnel experiences or use heuristic rules to schedule the automated manufacturing systems. Several studies indicate that genetic algorithms can effectively solve the scheduling problems too complicated to be solved by other methods. Genetic algorithms are optimization techniques that have been applied successfully to many NP–complete optimization problems. As seen earlier, the current solution approaches to scheduling are mainly limited to mathematical, simulation and artificial intelligence techniques. All of these techniques are time–consuming. In this research, a new approach; the genetic algorithms approach, is presented for solving such scheduling problems which is based on using a fixed search space to find efficient schedules for the manufacturing orders. It is expected that by using this approach, space and time complexity can be reduced greatly. The survey of literature reveals the following facts:

(i) Mathematical programming or analytical techniques are good for small–sized problems. For large–sized scheduling problems, this approach resulted in NP–completeness and hence no optimal solution could be obtained for them. By using this approach, substantial reprogramming effort is required in case of rescheduling.

(ii) Simulation approaches primarily use dispatching rules for scheduling manufacturing systems. These models are costly in terms of computational time and the required human modeling effort. In the case of rescheduling, the simulation model has to be modified, resulting in the repetition of schedule generation.

(iii) Artificial intelligence or expert systems techniques utilize heuristics for scheduling manufacturing systems. They always create a feasible schedule provided they have a well organized set of rules. The main drawback to this approach is that only a few scheduling experts are available to provide the knowledge able to be put into rules. Secondly, substantial amount of computer time is required to obtain a feasible schedule.

(iv) In any of the above approaches alternate routing flexibility is not considered in conjunction with dispatching rules and order size greater than one. It is worthwhile to consider these issues together and use genetic algorithms approach to solve scheduling problems.

Batch splitting is the partitioning of the selected part types into small batches in order to satisfy due–date requirements and improve resource utilization. The batch splitting models attempt to trade off between the productivity losses from making too many small batches and the opportunity cost of tying up capital in inventory as large batches. The literature survey indicates the following facts:

(i) The batch scheduling problem is usually considered as two independent subproblems of batch splitting and scheduling.

(ii) Batch splitting models are usually based on inventory models and cost functions such as ordering, inventory holding, WIP carrying, etc., which are not always easy to obtain.

(iii) Batch scheduling for multi–machine problems remains unsolved.

(iv) Current batch scheduling models assume a FCFS queue.

(v) The relationship between batching and performance measure is not fully explored in the case of precedence and alternate routings.

The performance of a production system depends greatly on good and proper rescheduling with the uncertainties present in the production environment. These

uncertainties often result in orders following a route through the shop floor different than that originally developed. In such cases, previously generated schedules become invalid and have to be regenerated. The literature review gives the following picture in this area:

(a) Rescheduling is done on entire set of operations resulting in time complexity.

(b) No change in existing operation sequence is considered, i.e., the operation assignment to machine remains the same.

2.5 Objectives of the Research

The overall objective of this research is to develop an integrated scheduler that includes development of efficient algorithms for manufacturing scheduling, consideration of batch splitting and dynamic scheduling. The following steps were identified to achieve this objective:

(i) *Application of genetic algorithms to the schedule optimization problem.* The research in this area progressed in two stages, (a) when there is only one process plan available (i.e., no routing flexibility) and (b) when there are multiple process plans available (with routing flexibility), to process the job orders in the manufacturing systems. Because of the vast number of schedules available in the case of multiple process plans, dispatching rules have been employed in the genetic algorithms to obtain a satisfactory schedule.

(ii) *Introduction of batch–splitting policies and order regeneration methods to improve the performance of the manufacturing system.* Six different batch splitting policies are used in this research to split the batches based on the process plan of the job orders. The different batch splitting policies that are employed in this research are:

   (a) No batch splitting

   (b) Splitting the batch in half

   (c) Splitting the batch in equal sizes (all sub–batch sizes are equal)

(d) Splitting the batch according to processing time

(e) Splitting the batch according to number of operations

(f) Splitting the batch according to number of machines

(iii) *Dynamic Scheduling*, i.e., development of rescheduling algorithms to reduce the proposed schedule disruption. Four different types of uncertainties that normally cause discrepancies between the actual output and the planned output were considered in this research. These uncertainties are unexpected machine breakdowns, increased order priorities, rush order arrival and order cancellations. The proposed rescheduling algorithms revise only those tasks that need to be rescheduled. The objective of the rescheduling algorithms is to minimize global changes and use local alternative resources to suggest simplest needed re-routing. The current status of the shop is considered, i.e. system status is updated whenever any change in the schedule takes place.

(iv) *System Implementation.* As part of the system implementation, a user interface has been developed in 'C' to input required scheduling data used for schedule optimization. The scheduler outputs a list of several good schedules. These schedules are then analyzed by the output analyzer which makes use of weighted primary and secondary performance criteria to select the best schedule. The final output from the analyzer is a list of different performance criteria values, machine utilizations, ready and completion times of each order, and the Gantt chart of the best schedule.

CHAPTER 3

GENETIC ALGORITHMS

3.1 Introduction

There is a large class of interesting problems for which no reasonably fast algorithms have been developed. Many of these are optimization problems that arise frequently in applications. However, with difficult optimization problems, it is often possible to find an efficient algorithm whose solution is approximately optimal. For these problems we can use probabilistic algorithms as well; these algorithms do not guarantee the optimal value, but by randomly choosing sufficiently many 'witnesses', the probability of error may be made sufficiently small [Michalewicz, 1992]. Genetic algorithms (GAs) belong to the class of probabilistic algorithms, yet they differ greatly from random algorithms as they combine elements of directed and stochastic searches. Because of this, genetic algorithms are also more robust than existing directed search methods. Another important property of such genetic–based search methods is that they maintain a population of potential solutions – all other methods process a single point of the search space.

Most of the techniques used for optimization are iterative improvement techniques, where the technique is applied to a single point (the current point) in the search space. During a single iteration, a new point is selected from the neighborhood of the current point. If the new point provides a better value of the objective function, the new point becomes the

current point; otherwise, some other neighbor is selected and tested against the current point. The method terminates if no further improvement is possible.

The genetic algorithms perform a multi–dimentional search by maintaining a population of potential solutions and encourages information creation and exchange between these directions. Using past information, they direct the search with expected improved performance and achieve fairly consistent and reliable results [Whitley 1989]. The population undergoes a simulation evolution, i.e. at each generation the relatively 'good' solutions reproduce, while the relatively 'bad' solutions die. To distinguish between different solutions, an objective function (also called evaluation function, or fitness function) is used which measures the quality of the solution. Since their introduction by Holland [1975], they have been extensively studied and applied to a variety of problems, including machine learning and NP–hard optimization problems [Goldberg, 1989].

Genetic algorithms were invented to mimic some of the processes observed in natural evolution. They use a vocabulary borrowed from natural genetics. Evolution takes place in the *population* consisting of *individuals* (also called *chromosomes, strings, structures, or genotypes*). Each chromosome would represent a potential solution to a problem. An evolution process runs on a population of chromosomes and corresponds to a search through a space of potential solutions. These chromosomes are made of *units* (also called *genes, features, characters, or decoders*) arranged in a linear succession. Every gene controls the inheritance of one or several characters. Genes are located at certain places on the chromosome called *loci* (or string position). Any character of individuals can manifest itself differently; the gene is said to be in several states, called *alleles* (feature values) [Goldberg, 1989].

3.2 Types of Genetic Algorithms

Genetic algorithms vary from practitioner to practitioner, but the basic building block

for each genetic algorithm is the same. There are two types of genetic algorithms currently in practice, namely standard genetic algorithms (also known as traditional genetic algorithms) and steady state genetic algorithms. The basic structure of and differences between these two types of algorithms are explained in the following sections.

3.2.1 Standard Genetic Algorithms (SGAs)

These algorithms are also regarded as the traditional genetic algorithms in that much of the early work was carried out using these types of algorithms. Since their introduction by Holland [1975], significant amount of work regarding theory and practice of genetic algorithms has been carried out. These algorithms normally employ binary encodings, generational reproduction, and simple one–point crossover and mutation operators. The simulation of genetic evolution here is contrived so in that the population size is forced to remain unchanged during the evolution process; two mating parents create only two children but are themselves eliminated. The basic characteristics of these algorithms are outlined below:

(i) Reproduction is based on fitness value and crossover probability, and is carried out in two steps. First, strings are reproduced according to their fitness values (the more fit the strings, the more chances of getting a copy in the next generation) and second, parent string pairs are selected according to the crossover probability p_c for offspring production.

(ii) Multiple reproduction and recombination occurs at one time. In each new population, a fixed number of strings (equal to the p_c x N) undergo crossover, where N is the population size. Here the crossover rate controls the frequency with which the crossover operator is applied. For example, if N is 100 and p_c is 0.6, then in the next generation 60 strings will undergo crossover.

(iii) Parents die and are replaced by their children. This is the main drawback of these algorithms: after the crossover application, parent strings on which the crossover was applied are lost. All the parent sets are replaced by their children.

(iv) The best solution found may be lost. Because all the parents are replaced by their children, chances of loosing the best solution are greatly increased.

(v) The feedback is slower because of multiple interactions between generations. Since the operators are applied on more than one pair of strings, the feedback rate is usually slower.

(vi) Because of the nature of genetic operators, chances of duplicate strings being created are greatly increased.

The basic structure of standard genetic algorithms is given in Figure 3.1. First, an appropriate chromosome representation should be defined to represent the problem that corresponds to the fitness or objective function values. Next, the population size and maximum number of generations should be specified. The probabilities of crossover and mutation are also specified. A set of initial population P(t) is then generated. The fitness value (objective function value) is next evaluated for each individual in the current population. The genetic operators are then applied on the old population to generate the new population for the next generation. Individuals in the new population are then evaluated and the procedure is repeated until the maximum number of generations is achieved. Two generations of SGAs is shown in Figure 3.2. In the figure, individuals 1 and 2, 4 and 5, 7 and 8, 10 and 11, and 13 and 14 are selected for the genetic operator application. The children produced as a result are 1' and 2', 4' and 5' and so on respectively. Individuals 1' and 2' are the children of parents 1 and 2. These children replace their parents and the procedure is continued until the specified number of generations are reached.

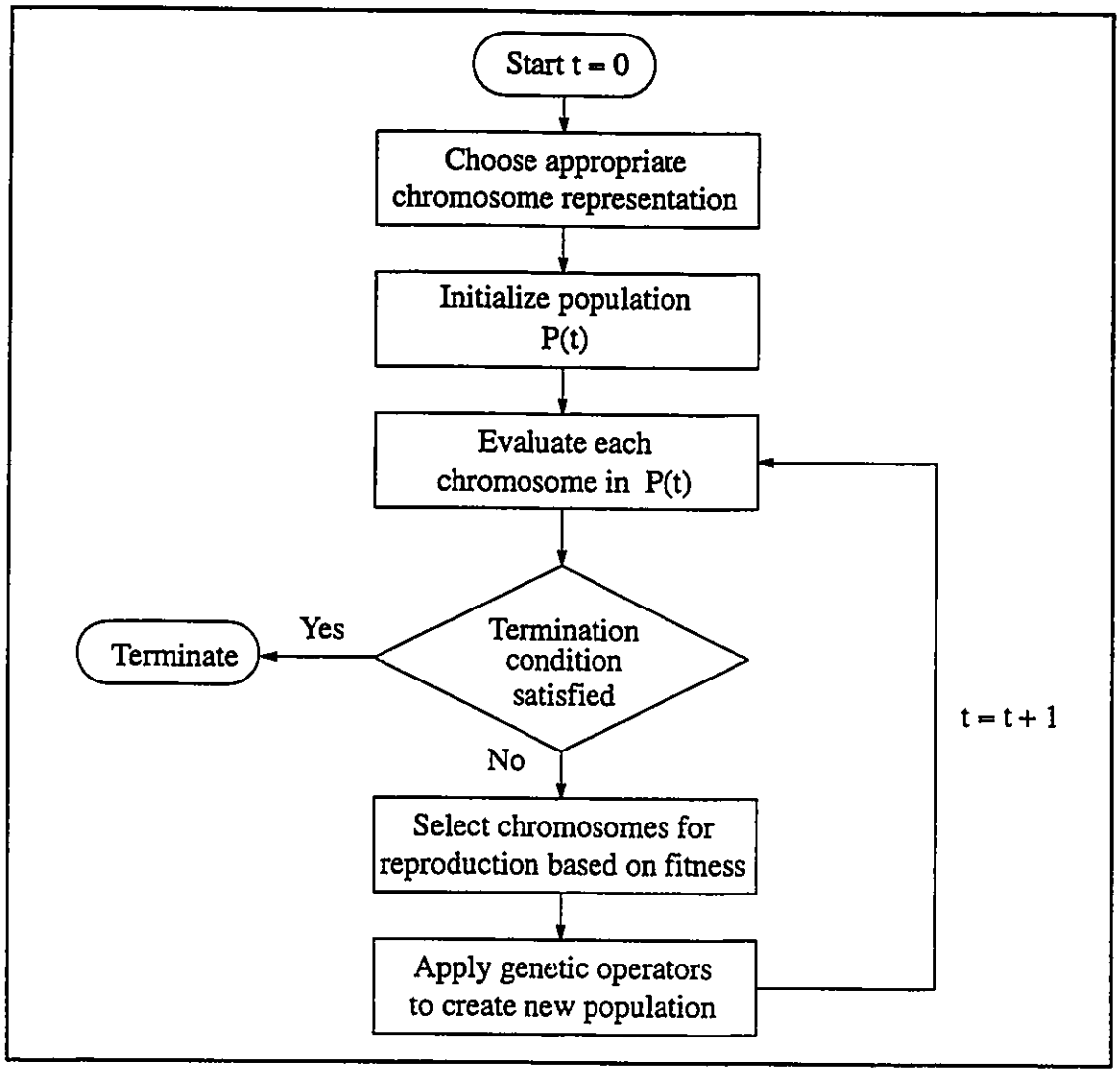


Figure 3.1: Basic structure of standard genetic algorithms

3.2.2 Steady–State Genetic Algorithms (SSGAs)

These types of algorithms were introduced by Whitley and Kauth [1988] in their genetic algorithms software package which they later called 'GENITOR'. These algorithms are also termed as Genitor–style algorithms. The main characteristics of these algorithms are:

(i) Reproduction is based on the ranking of individuals. All the individuals in the population are ranked according to their fitness values. Based on this ranking, two

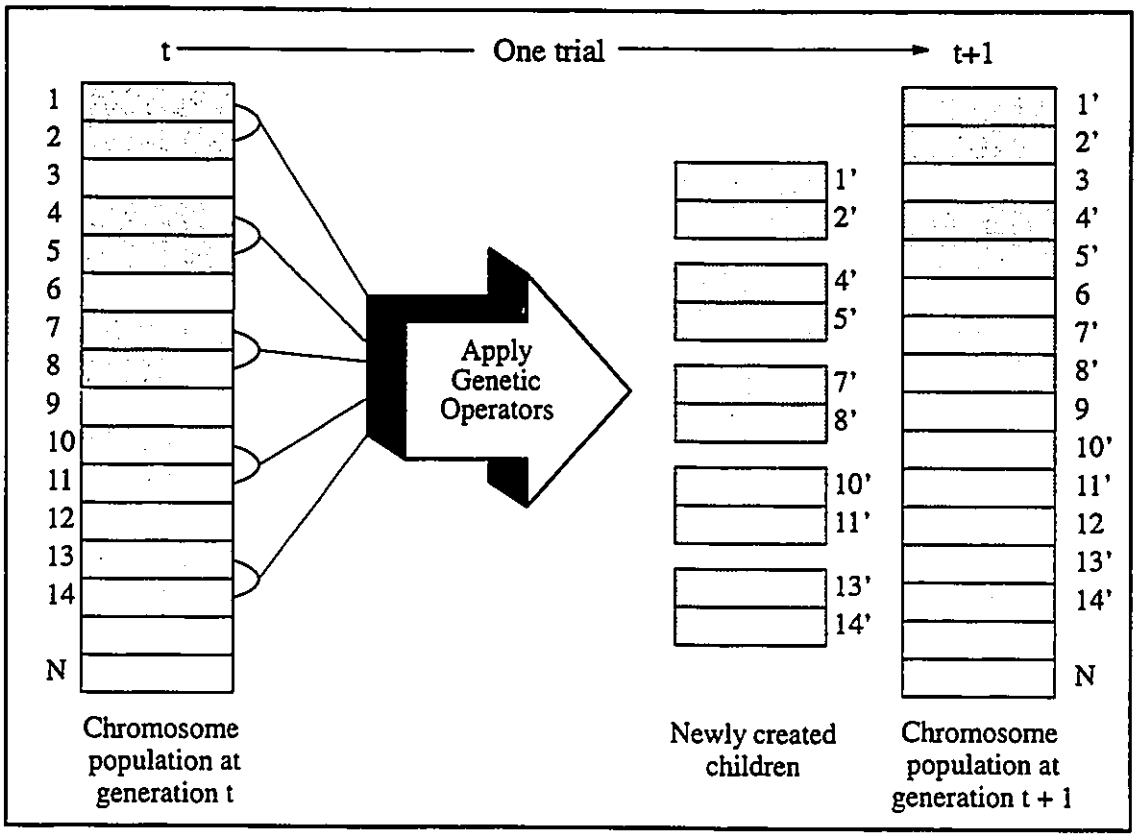


Figure 3.2: Two generations of SGAs

parents are selected for reproduction and produce an offspring that is immediately placed back into the population.

(ii) One–at–a–time recombination and replacement occurs in a generation. Unlike standard genetic algorithms only one recombination in a population takes place in one generation.

(iii) Parents and children both live and the worst solution in the population dies. The major difference of steady–state algorithms is that how the newly created offspring is placed into the population. When ranking is used, the worst individual in the population is replaced by newly created offspring. In other words, parents and children both stays in the population after recombination phase.

(iv) Best found solutions are maintained in the population. Since the worst solution is replaced from the population, the best solution found in previous generations is always maintained in the current population until displaced by a better solution. This can result in more aggressive form of search and is often quite effective.

(v) The feedback is faster because of one–at–a–time recombination. The feedback rate is faster than for standard genetic algorithms because of the single interaction between generations.

(vi) Parents are selected according to a selection bias. Since there is only one interaction at a time, it is very important to select the individuals from the population that will undergo recombination. Instead of crossover probability, a different parameter, known as selection bias, is used in these algorithms. Selection bias is a floating point number that specifies the amount of preference to be given to the superior individuals in a genetic population.

(vii) Duplicate strings are never created. Steady–state without duplicates is a reproduction technique that discards children who are duplicates of current chromosomes in the population rather than inserting them into the population. This ensures that every member of the population is different. This involves some overhead, in that it requires potentially a large number of equality tests whenever a child is created. An important fact is that the additional execution time spent by a real–world genetic algorithm using steady–state without duplicates is negligible with regard to the total time spent in optimization [Davis, 1991].

Whitley [1989] used GENITOR to test several problems and presented the evidence and arguments for the superiority of steady–state algorithms over standard genetic algorithms. The basic structure of steady–state genetic algorithms is the same as standard genetic algorithms except that in SSGAs ranking is employed and recombinations take place only once in one generation. This is shown in Figure 3.3. Allocating reproductive trials

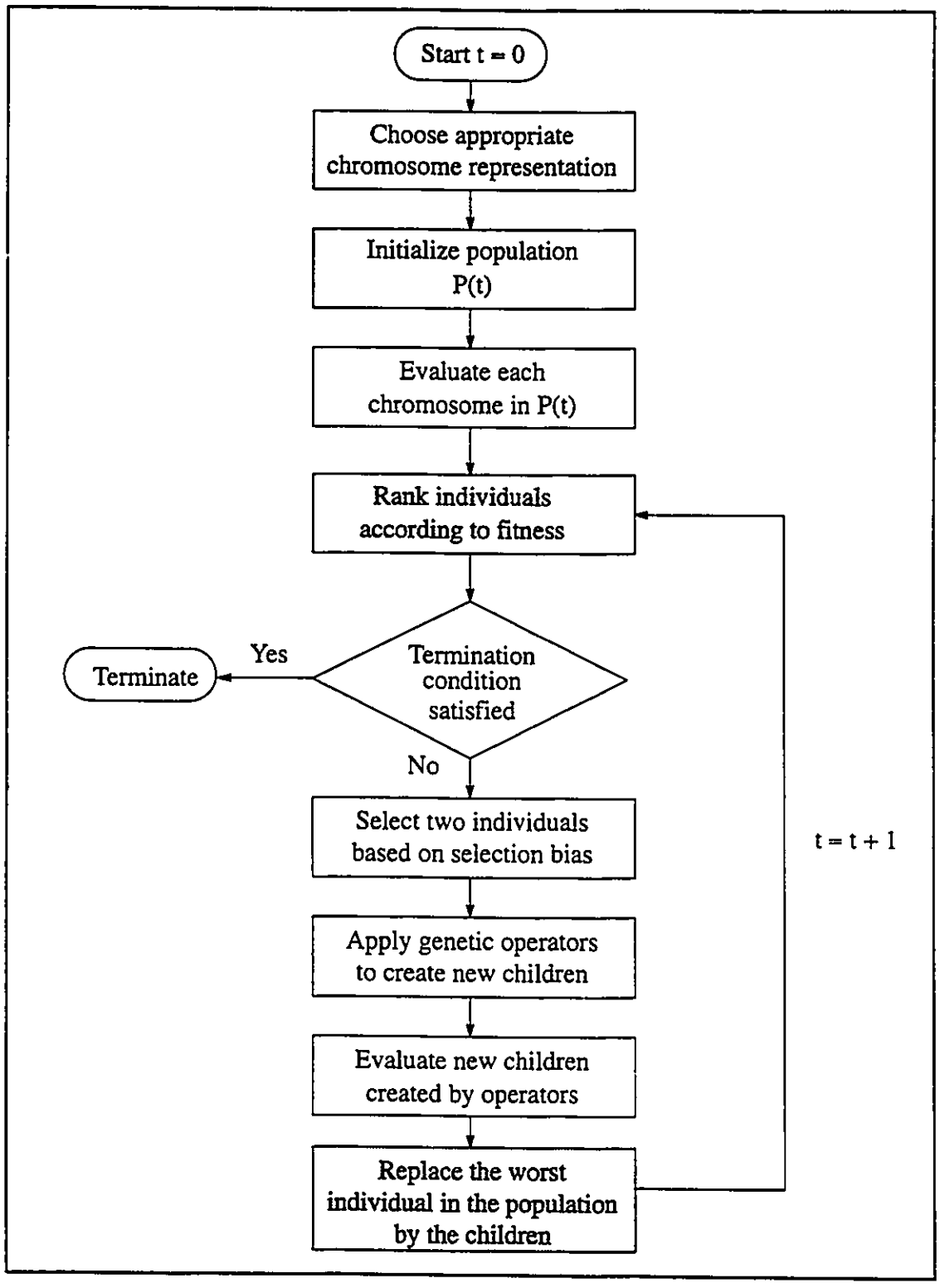


Figure 3.3: Basic Structure of steady state genetic algorithms

according to rank provides a means for directly controlling selective pressure [Whitley, 1989]. Other parameters such as those used by Grefenstette [1986] affect selective pressure indirectly and imprecisely. Figure 3.4 shows two generations of SSGAs. As explained earlier, the individuals are first ranked according to their fitness values. In the figure, F1, F2, etc. are the fitness values of the individuals. Only two individuals are selected (1 and 4) for the genetic operator application, the result of which are two children, 1' and 4'. The child with the higher fitness value enters the population, replacing the worst individual in the population. The child with the lower fitness value is simply discarded.

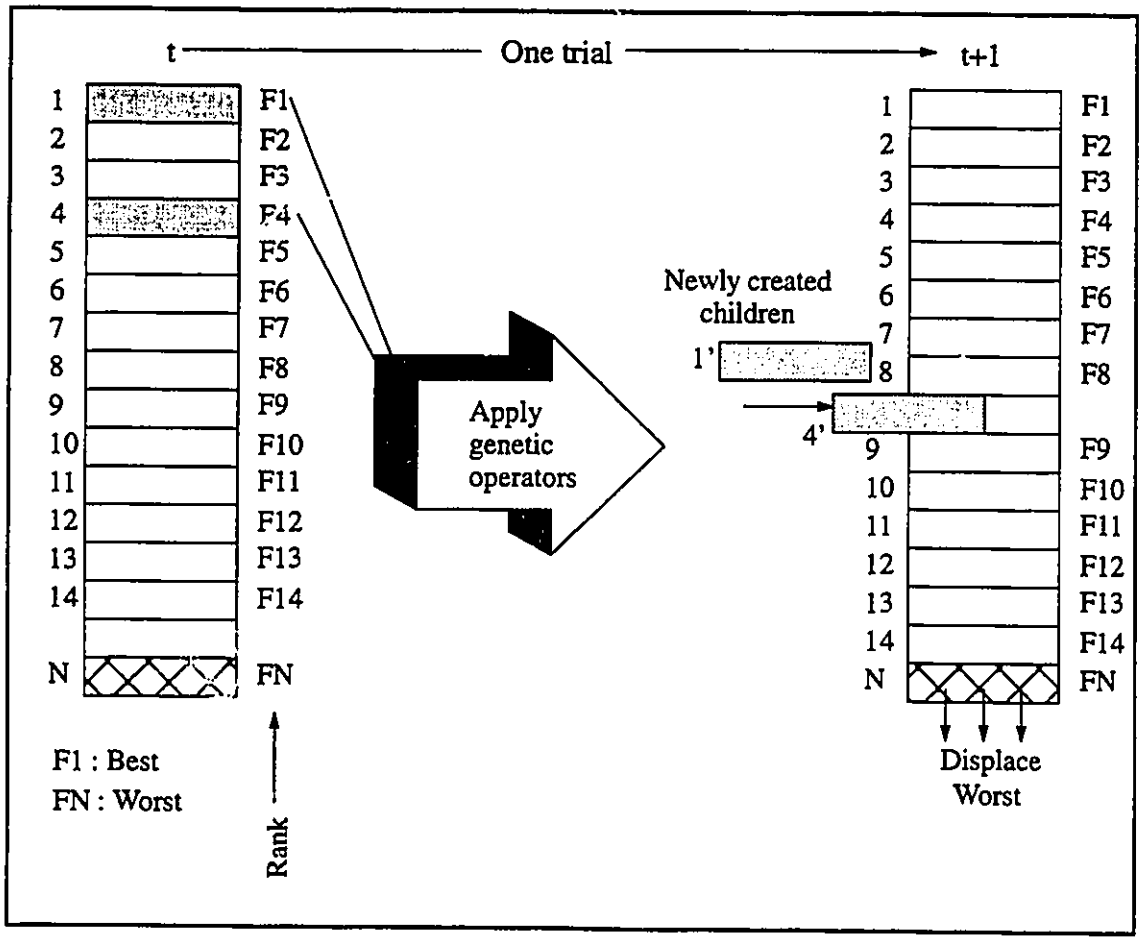


Figure 3.4: Two generations of SSGAs

3.3 Components of Genetic Algorithms

For any particular problem, the genetic algorithms must have the following five components:

(i) a genetic representation for potential solutions to the problem

(ii) a way to create an initial population of potential solutions

(iii) an evaluation function for rating solutions in terms of their "fitness"

(iv) genetic operators that alter the composition of children during reproduction

(v) values of various parameters that the genetic algorithms uses (population size, probabilities of applying genetic operators, etc.)

A great deal of study has been devoted to these components; however, the three issues that are frequently addressed in literature are problem representation, genetic operators, and selection of parameter values. The following section provides a general discussion about these components.

3.3.1 Representation

In all of Holland's work [1975], and in the work of many of his students, chromosomes are bit strings – lists of 0's and 1's. Bit strings have been shown to be capable of usefully encoding a wide variety of information, and they have been shown to be effective representation mechanisms in unexpected domains (such as function optimization). The properties of bit string representations for genetic algorithms have been extensively studied, and a good deal is known about the genetic operators and parameter values that work well with them.

However, there is a large class of problems that cannot be encoded as a bit string because of the ordering dependencies among genes of the chromosomes. The example of such optimization problems include the traveling salesman problem (TSP), bin packing

problem, and job shop scheduling problems. The representation used for such problems are called list representations.

3.3.2 Genetic Operators

In order to ensure that the problem space is comprehensively searched, a means must be introduced in the genetic algorithms. This population variation is developed by using genetic operators. The classic genetic operators are:

3.3.2.1 Reproduction

The reproduction process is nothing more than a selection of the more fit members of the population, from which members are selected for crossover and mutation transformations. There are many different ways to implement the reproduction operator; almost any method that biases selection toward fitness seems to work well. One approach to selecting members from the initialized population is to assign each member a probability of selection $f_i / \sum_m f_i$, where m is the total population size. A mating pool can then be created, of the same size as the initial population, but with a higher average fitness function value.

3.3.2.2 Crossover

While reproduction duplicates the most fit members of a population for mating, it does not in any way improve any single structure in the population. It is the crossover operator that allows the characteristics of the chromosomes in the population pool to be altered, with the intent of representing the best characteristics in the next generation. This is similar to the transfer of genetic material from parents to children in biological processes. The crossover is executed by selecting two mating parents, randomly choosing a site on the genetic chromosome, then swapping strings of two parents on either side of the crossover site. A probability of crossover p_c is defined by the user to determine if crossover should be implemented.

A good deal of research in the genetic algorithm field has been devoted to

investigating the effects of variations on the crossover operator, as well as to developing new kinds of crossover operators tailored to different encoding schemes and specific problem types. A simple crossover operator, as mentioned earlier, was devised for binary representations only. Scheduling problems fall into the category of ordering problems where binary chromosome representations are not easy to use. Scheduling problems usually employ list representations where the chromosome is represented by the value of the gene itself (for example, operation, machine, city etc.) making chromosome a sequence of operations, machines or the cities. These alternative representations tend to be coupled with genetic operators different from the ones originally proposed by Holland. Use of a simple crossover operator on such problems would produce illegal schedules and for this reason, a variety of crossover operators have been developed for such ordering problems [Potts et al., 1994]. Some of the commonly used operators are a two–point crossover, cyclic crossover, order crossover, partially matched crossover and edge recombination operator. It has been shown [Lee et al. 1993, Whitley et al. 1991] that a two–point crossover and edge recombination operator consistently performed better than other crossover operators. In this research, a two–point crossover operator is used in the case of multiple process plan scheduling and an edge recombination operator is used for a single process plan.

3.3.2.3 Mutation

Mutation safeguards the genetic search process from a premature loss of valuable genetic material due to reproduction and crossover. Mutation is not a primary genetic operator. The process of mutation is simply to choose a few members from the population pool according to the probability of mutation p_m, and alter the feature value of a randomly selected gene on each chosen string. These operators are shown in Figure 3.5.

3.3.3 Parameters of Genetic Algorithms

The task of optimizing a complex system presents at least two levels of problems for

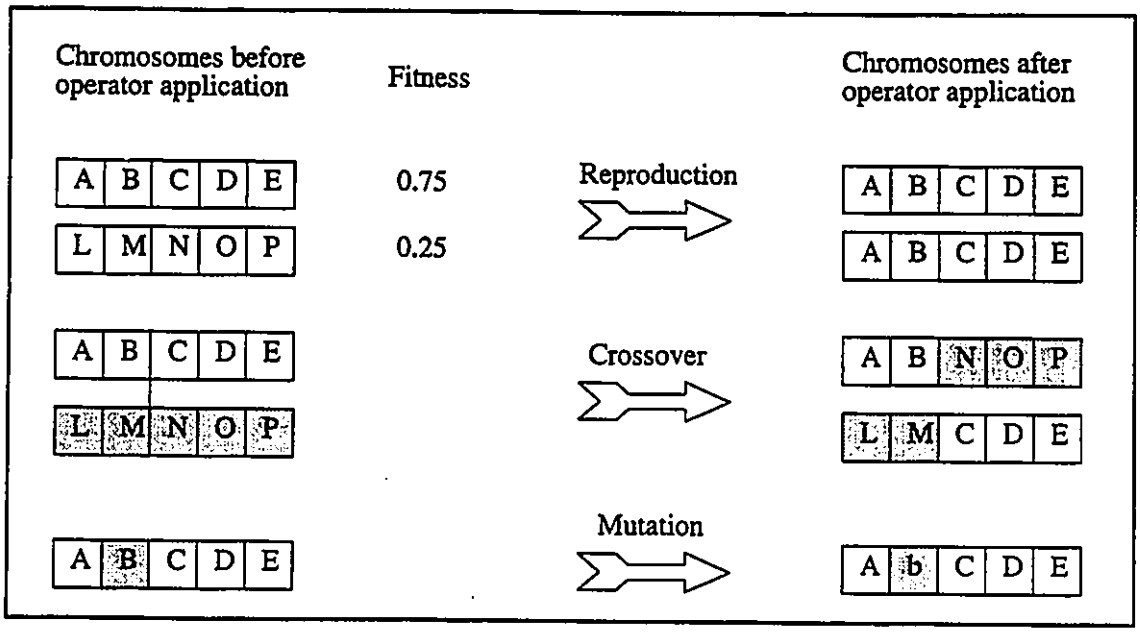


Figure 3.5: Genetic operators

the system designer. First, a class of optimization algorithms must be chosen that is suitable for application to the system. Second, various parameters of the optimization algorithm need to be tuned for efficiency [Grefenstette, 1986]. Various parameters that play a crucial role for the successful implementation of genetic algorithm are the crossover rate, mutation rate, and population size. The settings of these parameter values significantly affects the performance of genetic algorithms. The problem of setting the parameter values has been studied extensively for bit string representations [De Jong 1975, Grefenstette 1986, Schaffer et al. 1989]. Very little work has been reported for other types of representations. Since this research uses steady–state genetic algorithms, new genetic operators for recombining individuals are used instead of regular crossover and mutation operators. These are reduced surrogate crossover, edge recombination and adaptive mutation. In this research, extensive experiments have been conducted to obtain the best combination of these parameters.

3.4 Selection of Genetic Algorithms Parameters

Various parameters studied in this research are selection bias, adaptive mutation and population size. These are explained in the following section.

3.4.1 Selection Bias

In SGAs, the crossover operator is used to select and change the strings in the population. The frequency with which the crossover operator is applied is controlled by the crossover rate. In each new population, N x p_c strings undergo crossover. The higher the crossover rate, the more quickly new strings are introduced into the population. If the crossover rate is too high, high performance strings are discarded faster than selection can produce improvements. If the crossover rate is too low, the search may stagnate due to a lower exploration rate. Hence, it is very important to find a crossover rate in order to successfully implement the genetic algorithm. Apart from crossover, there are other parameters in SGAs such as generation gaps, selection strategy and scaling factors that are used to control the selective pressure. Generation gap (G) controls the percentage of the population to be replaced during each generation. N x G strings of the population $P(t)$ are chosen to be replaced in $P(t+1)$. A value of G = 1.0 means that the entire population is replaced during each generation. A selection strategy is used to select the strings from each population that are supposed to be used for recombination. In SGAs two selection strategies are normally used. The first is a pure selection strategy where each string in the current population is reproduced in proportion to the string's fitness. The second is an elitist strategy: pure selection is performed first and the structures with the best performance are chosen to survive to the next generation, avoiding removal by crossover, mutation, or sampling error. A scaling factor is used to maintain the population diversity [Grefenstette, 1986].

By using selection bias instead of crossover rate, the problem of selecting a good

crossover rate is greatly reduced. Selection bias is a floating point number which specifies the amount of preference to be given to superior individuals in a genetic population. In SSGAs only one pair of strings are allowed to undergo recombination. The selection of 'selection bias' plays an important role in SSGAs. Various parameters of SGAs such as crossover rate, selection strategies, scaling windows and generation gaps are replaced by the selection bias in SSGAs [Whitley, 1989].

3.4.2 Adaptive Mutation

In SGAs, to increase the variability of the population, a secondary search operator, mutation, is normally used. Random mutation provides background variation and occasionally introduces beneficial material into the strings. After selection, each bit position (gene position) of each string in the new population undergoes a random change with a probability equal to the mutation rate p_m. Consequently, approximately $p_m \times N \times L$ mutations occur per generation, where N is the population size and L is the string length. A low level of mutation serves to prevent any given bit position from freezing at a single value in the population. A high level of mutation yields an essentially random search.

Adaptive mutation [Whitley, 1988] has been used in this research rather than normal mutation. Adaptive mutation bases the amount of disruption to a given string on two factors: the relative similarity of its two parent strings and the actual mutation rate. The more similar the two parent strings, the more likely mutation is to occur. The actual mutation that occurs is the product of this similarity and a fractional mutation rate. Thus, if the mutation rate is 0.20 and the parents of a particular string are identical, approximately 20% of the string value will be changed.

3.4.3 Population Size

Selecting an appropriate population size is crucial in SGAs. Population size affects both global performance and the genetic algorithms' efficiency. Genetic algorithms with

small populations usually perform poorly because the population provides an insufficient coverage of the problem space. A large population is more likely to be representative of the entire problem domain. Furthermore, a large population prevents premature convergence to local instead of global solutions. A drawback of the large population is that it requires more evaluations per generation, possibly resulting in an unacceptably slow rate of convergence.

In SSGAs, population size is not that important as only two evaluations per generation occur because of one–at–a–time recombination. It is only in the first generation when all the strings in the population are evaluated after which there is only one evaluation per generation. This accounts for the faster speed of SSGAs compared to SGAs. However, in order to examine the effect of population size on steady–state genetic algorithms, this parameter is also taken into consideration in this research.

CHAPTER 4

SINGLE PROCESS PLAN SCHEDULING

In this chapter, a genetic algorithm model is developed for the case where there is only one process plan available per job, i.e., when there is no routing flexibility. The objectives considered for scheduling in this chapter are minimizing the makespan or mean flow time. Genetic algorithms design issues are discussed and the working of genetic operators employed is explained in detail. Parameter settings for the genetic algorithms used for the single process plan scheduling (SPPS) problem are done through extensive experimentation. Finally, the genetic algorithms approach is compared to several other approaches in terms of optimality of solution and time complexity.

4.1 Problem Description

The detailed scheduling of the various elements of a production system is crucial to the efficiency and control of tasks. Orders are released into the system where each order may contain a set of jobs of a particular job type. Each order has a due date associated with it and in all cases it is the desire of the manufacturer to comply with these dates. Each job consists of a number of tasks to be performed in a given sequence on the available machines. In exceptional cases, such as machine breakdowns, preemption of ongoing jobs may occur where high priority jobs arrive at busy machines and must be processed immediately. Figure 4.1 illustrates the relationship among orders, job types and tasks.

The single process plan scheduling (SPPS) problem may be stated as follows: 'Given

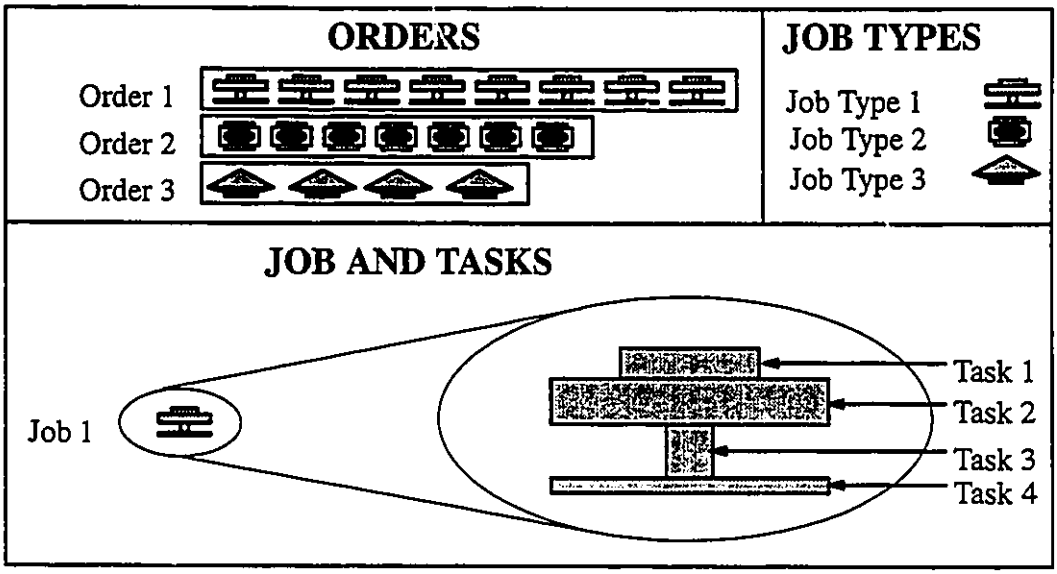


Figure 4.1: Relationship among orders, job types and tasks

process plans for each part, the objective is to find a feasible schedule for a given set of jobs such that some performance criteria is optimized'. The performance criteria chosen for the SPPS algorithms here are minimizing the makespan or the mean flow time. As seen earlier in the literature review, several approaches have been used to solve the scheduling problem, including analytical, simulation and experts systems approach. In this research, the genetic algorithms approach is used to generate and evaluate feasible schedules. The prerequisite to this stage includes a process plan of each job and genetic algorithms parameter values. The process plan of a job includes:

i) Number of jobs. This means the production quantity of each job and variety of different jobs that are to be scheduled should be known in advance.

ii) Number of operations in each job. This means the number of different operations of each job should be known in advance. Each operation is completely performed on one machine, i.e. it is assumed that an operation does not need more than one machine for its processing.

iii) Number of each resource (machine) type. This means that the capacity of each resource and its availability are known before scheduling.

iv) Processing and setup time of each operation on each machine. This means that the processing and setup time for each operation is known a priori before scheduling.

v) Precedences among operations within a job. Certain operations often cannot be performed until all its preceding operations are completed. For instance, it is necessary to make a hole before it can be threaded.

vi) Due date of each job type. This is the time at which each job is supposed to leave the system, and is the time by which the processing of the last operation should be completed.

vii) Ready time, release time or arrival time. This is when a job is released to the shop floor by some external job generating process. This is significant because it is the earliest time when the processing of the first operation of a job can begin.

In this chapter we are concerned with non–preemptive scheduling problems where n jobs are available to be processed on m machines at time t = 0. Associated with each job j is a processing time t_j. The following performance measures are employed in this research:

$$\text{Mean Flow Time (MFT)} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=J_i} C_{ij} \tag{4.1}$$

$$\text{Makespan (MS)} = \max(C_{ij}), \forall i, \text{ and } j = J_i \tag{4.2}$$

The scheduling decision involved here is the order in which jobs are to be scheduled on individual machines, which enables us to denote any schedule by a permutation of job indices and to find a permutation that minimizes the objective function.

4.1.1 Modeling Assumptions

In order to be able to develop a tractable scheduling model, several assumptions are

usually made. Some of the assumptions made at this initial stage of the research while scheduling with genetic algorithms are:

(i) Pre–emption is not allowed (i.e., operations that begin processing are completed without interruption)

(ii) Machines never break down.

(iii) Operators, tools, and material are always available.

(iv) Due dates, if they exist, are fixed.

(v) Processing times are deterministic and known in advance.

(vi) Each machine can perform only one operation at a time.

(vii) Setup times are machine dependent.

(viii) Each job, once started, must be performed to completion (i.e., no order cancellations).

(ix) Each job can be processed by only one machine at a time (i.e., jobs are assumed to be indivisible).

(x) Transportation time between facilities is negligible or included in processing time.

(xi) Jobs are not recycled due to fabrication errors, engineering changes, etc.

Some of the above assumptions are relaxed in a later chapter (Chapter 7: Dynamic scheduling) to achieve a better representation of reality. These are:

(i) Machines break down is considered.

(ii) Pre–emption of jobs is allowed in the case of high priority jobs.

(iii) Due dates are not fixed.

(iv) Order cancellations are permitted.

4.2 Genetic Algorithm Design Issues

Genetic algorithms have been applied to a variety of function optimization problems, and have been shown to be highly effective in searching large, complex response surfaces even in the presence of difficulties such as high dimensionality, multi–modality,

discontinuity and noise (Dejong, 1975). Genetic algorithms are a class of optimization algorithms that seek improved performance by sampling areas of the parameter space that are more likely to lead to good solutions. In this chapter, these algorithms are examined as a possible method for solving combinatorial–type scheduling problems efficiently. Their chief advantage lies in their ability to jump randomly from schedule to schedule, allowing them to escape from local optima in which other algorithms often get trapped. Stochastic processes generate an initial population of schedules and then apply principles of natural selection/survival of the fittest to improve the schedules. The achievement of the optimum schedule that conforms to the stated 'objective' is naturally an attractive aim of the scheduler. It was seen from the literature review that substantial effort has been devoted to the associated mathematical and computational background and the methods explored have been wide–ranging and often mathematically complex. The process of schedule generation is generally characterized by finite, often large, number of variables of the discrete type. The main ingredients of genetic algorithms and how they relate to manufacturing scheduling are explained in Figure 4.2. In this section, various issues in the design of genetic algorithms

Natural	Genetic Algorithm	Scheduling
Chromosome	String	List of operations/machines
Gene	Feature	Single machine
Allele	Feature value	Machine type
Fitness value	Cost function	Objective function

Figure 4.2: Natural and genetic algorithms terminology

for combinatorial–type optimization problems are discussed and the approach used in the development of the proposed genetic algorithm for SPPS problem is described. All these design issues are explained here using an example problem obtained from Sridhar and

Rajendran [1993]. The processing time matrix for this example is given in Figure 4.3. The

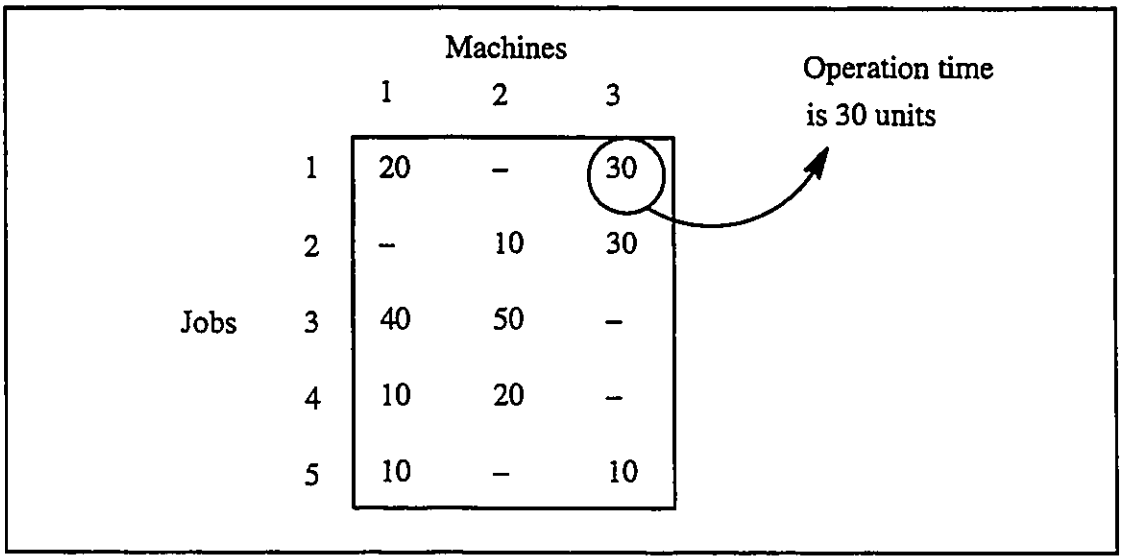


Figure 4.3: Processing time matrix for the chosen problem

objective here is to minimize the mean flow time of the jobs. There are ten operations in total that are to be performed on three different machines. The sequence of operations within a job are fixed whereas the sequence of jobs has to be optimized.

4.2.1 Schedule Representation

Choosing an appropriate representation is the first step in applying genetic algorithms to an optimization problem. A number of representations such as adjacency representation [Grefenstette et al. 1985], permutation representation [Goldberg and Lingle 1985], ordinal representation [Grefenstette et al. 1985] have been explored for such combinatorial–type scheduling problems. The genetic algorithms for SPPS problems uses a permutation type representation where the list of operations is used as the chromosome (schedule) as shown in Figure 4.4. Each operation is mapped onto a digit which is then used as the gene. The precedence among operations is taken care of by the evaluation function. Permutations of different operations constitute the search space denoted by n!.

Operation	11	13	22	23	31	32	41	42	51	53
Mapping	1	2	3	4	5	6	7	8	9	10

Figure 4.4: Schedule Representation for SPPS GAs

4.2.2 Initialization

The next important step is population initialization, i.e. the creation of an initial schedule population. Population initialization can be done in two ways. It can be either generated randomly or a well adapted (seeded) population can be used as an initial population. For the SPPS problem, an initial population of desired size has been generated randomly using a recursive procedure which lists all possible permutations randomly. An example of initial population thus generated by this procedure is shown in Figure 4.5. The

Chromosomes	Population
Schedule 1	3 2 9 7 10 4 8 5 6 1
Schedule 2	4 7 8 3 2 9 5 6 10 1
Schedule 3	3 8 4 7 9 1 2 6 5 10
Schedule 4	9 7 4 6 5 2 10 8 1 3
Schedule 5	2 3 7 9 10 6 1 5 8 4
Schedule 6	8 4 7 9 3 1 5 10 6 2
⋮	⋮
Schedule N	2 1 10 5 6 3 4 9 7 8

Figure 4.5: Initial Population of GAs

number of random schedules (N) to be generated in the population is specified by the user.

4.2.3 Evaluation Function

The objective function chosen here is to minimize the mean flow time. In each generation, the value of the fitness function or the objective function for a particular schedule is calculated using the following scheduling algorithm (evaluation function) [Nasr and Elsayed 1990]:

i) Calculate ready time for all operations

Let $t = 0$, and $a_k = 0$

Beginning with S as an empty set, while S' includes all the operations.

Set $r_{i1} = 0$, for all i

$r_{ij} = r_{i(j-1)} + t_{i(j-1)k}$, for all $i = 1, 2, ..., n$, and $j = 2, 3, ..., Ji$

ii) If there is any $[O_{ij} \in S' \mid r_{ij} \leq t]$ go to step iii).

Otherwise: Set $t = \min (r_{ij})$

iii) Place all $[O_{ij} \in S' \mid r_{ij} \leq t]$ in a set S''.

iv) Assign the operations to respective machines (if available) and calculate the completion time of each operation, i.e.,

$C_{ij} = \max (r_{ij}, a_k) + t_{ijk}$

v) For each operation in the set S'':

(a) Remove the operations, O_{ij}, from the set S' and the set S''.

(b) Place O_{ij} in the set S.

(c) Determine:

$S_{ij} = \max \{r_{ij}, a_k\}$

$C_{ij} = S_{ij} + t_{ijk}$

$r_{i(j+h)} = r_{i(j+h)} + [C_{ij} - r_{i(j+1)}]$ for $h = 1, 2, ..., (J_i - j)$

(d) Set $a_k = C_{ij}$.

vi) Check if S' is empty; if it is not, then go to (ii) and repeat the procedure until the schedule is complete.

In every generation, this algorithm is applied on the selected chromosome schedule to assign the operations to the machines. The fitness values thus obtained for the schedules are shown in Figure 4.6. The genetic algorithm proceeds from generation to generation,

Chromosomes	Population										Objective Function
Schedule 1	3	2	9	7	10	4	8	5	6	1	66.0
Schedule 2	4	7	8	3	2	9	5	6	10	1	70.0
Schedule 3	3	8	4	7	9	1	2	6	5	10	72.0
Schedule 4	9	7	4	6	5	2	10	8	1	3	72.0
Schedule 5	2	3	7	9	10	6	1	5	8	4	78.0
Schedule 6	8	4	7	9	3	1	5	10	6	2	96.0
⋮					⋮						⋮
Schedule N	2	1	10	5	6	3	4	9	7	8	154.0

Figure 4.6: Initial Population of GAs with fitness values

saving the best schedules in each generation and getting rid of those schedules whose objective function values are low compared to the saved ones. In each generation the schedule population is always kept fixed. Genetic algorithms are popular for searching complex surfaces since they use limited search space.

4.2.4 Recombination Operator – Edge Recombination Operator

A number of recombination operators has been investigated for scheduling problems such as a partially matched crossover (PMX), order crossover (OX), cyclic crossover (CX), and edge recombination operator. A comparison of these sequencing operators has been conducted by several authors [Starkweather et al. 1991, Lee et al. 1993]. It was observed

that, in most of the cases, edge recombination operator performed better than other sequencing operators. The edge recombination operator was developed by D. Whitley [1989] and is used in this research for the genetic algorithms employed for the SPPS problem. This operator uses an edge table to construct an offspring that inherits as much information as possible from the parent structures. This edge table stores all the connections from the two parents that lead into and out of the node (an operation). An example of an edge table and the operation of the edge–recombination operator are given in Figure 4.7.

4.2.5 Parameter Values

A variety of parameters such as selection bias (SBIAS), adaptive mutation (AMUT), population size (PSIZE), and number of generations (NGEN) play a crucial role in the successful implementation of genetic algorithms. The settings of these parameter values significantly affect the performance of genetic algorithms. The problem of setting the parameter values has been extensively studied for bit string representation (for example, Grefenstette [1986]). Very little work has been reported on other types of representations. In this research, an extensive computational experimentation has been performed for the selection of parameters and is given in Section 4.4.

4.3 Genetic Algorithms Model

The genetic algorithm for the SPPS problem is developed incorporating the design issues discussed above. These steps are now discussed in detail.

(i) (**Chromosome representation**) A schedule is represented here as a list of operations.

(ii) (**Initialization**) Select the initial parameters and create an initial diversified population of schedules.

   (a) Set the values for PSIZE, SBIAS, AMUT and NGEN. The length of the chromosome is set equal to the total number of operations to be scheduled.

- *Two parents selected are Schedule 2 and Schedule 5 from Figure 4.6*

Parent 1:	4	7	8	3	2	9	5	6	10	1	70.000
Parent 2:	2	3	7	9	10	6	1	5	8	4	78.000

- *Create edge table, which has edge connection information from both the parents*

Edge Table			
Operation	Edges/Links	Operation	Edges/Links
1	10, 4, 6, 5	6	5, 10, 1
2	3, 9, 4	7	4, 8, 3, 9
3	8, 2, 7	8	7, 3, 5, 4
4	1, 7, 8, 2	9	2, 5, 7, 10
5	9, 6, 1, 8	10	6, 1, 9

- *The recombination algorithm:*

1. Choose an initial operation from one of the two parents
2. Remove all occurrences of the 'current operation' from the edge table.
3. If the current operation has entries in its edgelist, go to 4; otherwise, go to 5.
4. Determine which of the operations in the edgelist of the current operation has the fewest entries in its own edgelist. The operation with the fewest entries becomes the current operation. Ties are broken randomly. Go to step 2.
5. If there are no remaining operations 'unvisited', then stop. Otherwise, randomly choose an "unvisited" operation and go to step 2.

- *By applying the edge recombination operator, the following offspring is obtained*

Offspring:	9	2	3	7	8	4	1	5	6	10	68.000

Figure 4.7: Working of edge recombination operator

(b) Read the processing times, setup times, due dates, and ready times for all the jobs.

(c) Create an initial population of schedules of size PSIZE and call it 'oldpop'.

(d) Calculate the objective function values for all the schedules in the population using an evaluation function described earlier.

(e) Sort the population in the increasing order of objective function value.

(f) Set NGEN = 1 (i.e., current generation = 1)

(iii) **(Recombination)** Apply recombination operator to the 'oldpop' to form a new population.

(a) Two parents are selected from the oldpop based on the specified selection bias. They are called 'mom' and 'dad'. Parents are selected based on the following algorithm [Whitley, 1989]:

$$\text{index} = \frac{\text{PSIZE} \times (\text{SBIAS} - \text{sqrt}(\text{BIAS}^2 - 4.0 \times (\text{SBIAS} - 1) \times \text{random}()))}{2.0 / (\text{SBIAS} - 1)}$$

where index is the schedule number to be selected from the sorted population

(b) The edge recombination operator is applied to the two selected parents to form a new child.

(c) Calculate the objective function value for this child.

(d) If the objective function value of the child is better than any of the schedules in the population, then insert the child in the population at the appropriate place, removing the worst schedule from the population.

(iv) **(New Generation)** Evaluate the current generation number to determine the next step.

(a) If GEN < NGEN, then GEN is incremented by one and the current population becomes oldpop. GO TO Step (iii).

(b) If GEN = NGEN, then STOP.

The best schedule would be the schedule in the current population with the highest objective function value.

4.4 Genetic Algorithms Parameter Setting for SPPS Problem

The most difficult and time–intensive issue in the successful implementation of the genetic algorithms is finding good parameter settings. A number of approaches have been suggested to derive robust parameter setting for traditional GAs, including carrying out brute force searches, using an adaptive operator fitness technique [Davis, 1991]. In one of the most extensive studies for determining the optimal parameter values, Schaffer et al. [1989] concluded that the optimal parameter settings vary from problem to problem. However, very little work has been reported regarding setting the parameters for steady state genetic algorithms used for the combinatorial type scheduling problems. The main parameters required for the genetic algorithms used for SPPS problem are population size (PSIZE) and the selection bias (SBIAS). Selection bias is a floating point number used in the selection of two schedules for genetic reproduction and can take a value between 1.0 and 2.0. This number specifies the amount of preference to be given to the superior individuals in a genetic population. For example, a bias of 2.0 indicates that the best schedule has twice the chance of being chosen compared with the average schedule.

In this research, extensive experiments have been conducted to obtain good values of genetic algorithms parameters. The genetic algorithms are implemented in 'C' on SUN computers. The tests were run on two well known examples obtained from the published literature for which the optimal values were known a priori. The first test problem (referred as A1 from now on) was obtained from Gupta [1972] and consists of six jobs to be scheduled on three machines. This example was subsequently used by Chan [1989] for testing his heuristic algorithms. In this research, this problem was solved using the genetic algorithms

approach for minimizing the makespan and mean flow time, and in both cases better results were obtained.

The second test problem (referred as A2 from now on) is a bench mark six–job and six–machine problem normally used by authors to test the efficiency of their solution approaches. The approaches used to solve this problem includes branch and bound [Barker et. al. 1985, Carlier and Pinson, 1989], heuristics [Adams et al. 1988, Yang et al. 1989, Vancheeswaran et al. 1993, Hatzikonstantis, 1992], simulated annealing [Laarhoven et al., 1992], and genetic algorithms [Nakano et al., 1991]. The data for both these problem sets is given in Appendix A.

The performance of genetic algorithms for the SPPS problem has been investigated for determining the optimal values of population size (PSIZE) and selection bias (SBIAS) using the edge recombination operator. For the experiments, eight levels of population size and five levels of selection bias were chosen as shown in Figure 4.8 The population size

Parameters	Levels							
Population Size (PSIZE)	25	50	75	100	125	150	200	400
Selection Bias (SBIAS)	1.1	1.3	1.5	1.7	1.9			

Figure 4.8: Experimental design parameters for SPPS GAs

was varied from small (size 25) to large (size 400) and selection bias was incremented in the steps of 0.2 from 1.1 to 1.9. The number of generations was fixed to 10,000. At each generation, the objective function values of the best and the worst schedule were compared. The genetic algorithms proceeds with optimization until the difference between them is zero. Both examples were run using different combinations of these parameters, and the results are tabulated in Table 4.1 and Table 4.2. The performance of genetic algorithms and their convergence trend for the fixed selection bias of 1.7 and 1.9 and variable population

Table 4.1: Results of Problem A1

PSIZE	SBIAS	Value	GEN	PSIZE	SBIAS	Value	GEN
25	1.1	67	100	125	1.1	66	5200
	1.3	67	100		1.3	**64***	4500
	1.5	66	100		1.5	66	2600
	1.7	67	300		1.7	66	3000
	1.9	66	400		1.9	67	900
50	1.1	66	700	150	1.1	66	4200
	1.3	67	400		1.3	**64***	4100
	1.5	66	1300		1.5	66	6000
	1.7	67	100		1.7	**64***	3500
	1.9	67	100		1.9	**64***	3400
75	1.1	66	2500	200	1.1	65	5300
	1.3	67	300		1.3	67	5100
	1.5	**64** *	4500		1.5	66	8100
	1.7	66	1100		1.7	66	7300
	1.9	66	500		1.9	66	8800
100	1.1	66	3700	400	1.1	66	7300
	1.3	66	2100		1.3	65	6000
	1.5	66	2600		1.5	66	5000
	1.7	66	2300		1.7	66	9800
	1.9	**64** *	3700		1.9	66	6100

sizes for problems A1 and A2 are plotted in Figures 4.9a and 4.9b. These figures show the convergence of average population mean with respect to the number of generations.

4.4.1 General Guidelines for Parameter Settings

From the results, it was observed that genetic algorithms generally perform poorly with small population sizes as they provide an insufficient sample size for the experimentation. Intuitively, it seems that increasing the population size should improve the performance of the genetic algorithms and, similarly, increasing the number of generations should also improve the genetic algorithms performance. However, from the results of problem A1 and A2, it was observed that a medium population size, i.e. a population size

Table 4.2: Results of Problem A2

PSIZE	SBIAS	Value	GEN	PSIZE	SBIAS	Value	GEN
25	1.1	59	100	125	1.1	59	900
	1.3	59	300		1.3	59	1300
	1.5	59	600		1.5	57	1700
	1.7	59	300		1.7	58	2500
	1.9	59	200		1.9	57	1900
50	1.1	59	400	150	1.1	59	2700
	1.3	59	300		1.3	57	3400
	1.5	59	500		1.5	58	4600
	1.7	59	1100		1.7	59	4300
	1.9	59	300		1.9	58	3400
75	1.1	59	600	200	1.1	59	5300
	1.3	58	1600		1.3	58	8500
	1.5	59	500		1.5	59	5800
	1.7	57	100		1.7	59	7700
	1.9	58	200		1.9	59	9800
100	1.1	59	500	400	1.1	59	8700
	1.3	59	200		1.3	59	7600
	1.5	58	900		1.5	59	9500
	1.7	55 *	3600		1.7	59	8900
	1.9	55 *	1000		1.9	59	7600

between 75 and 150, offered the best performance. Increasing the population size further did not result in any significant performance variation. Also, a large population requires more evaluations, resulting in an unacceptably slow rate of convergence. Hence for small scheduling problems a population size of 75 is suggested as a guideline principle; for large problems a population size of 150 is recommended.

For the experiments conducted, a selection bias value of 1.5 and greater yielded good results. Values lower than 1.5 are not recommended as these values are not sufficient to drive the search hard enough to achieve similar results. The results reported here are best

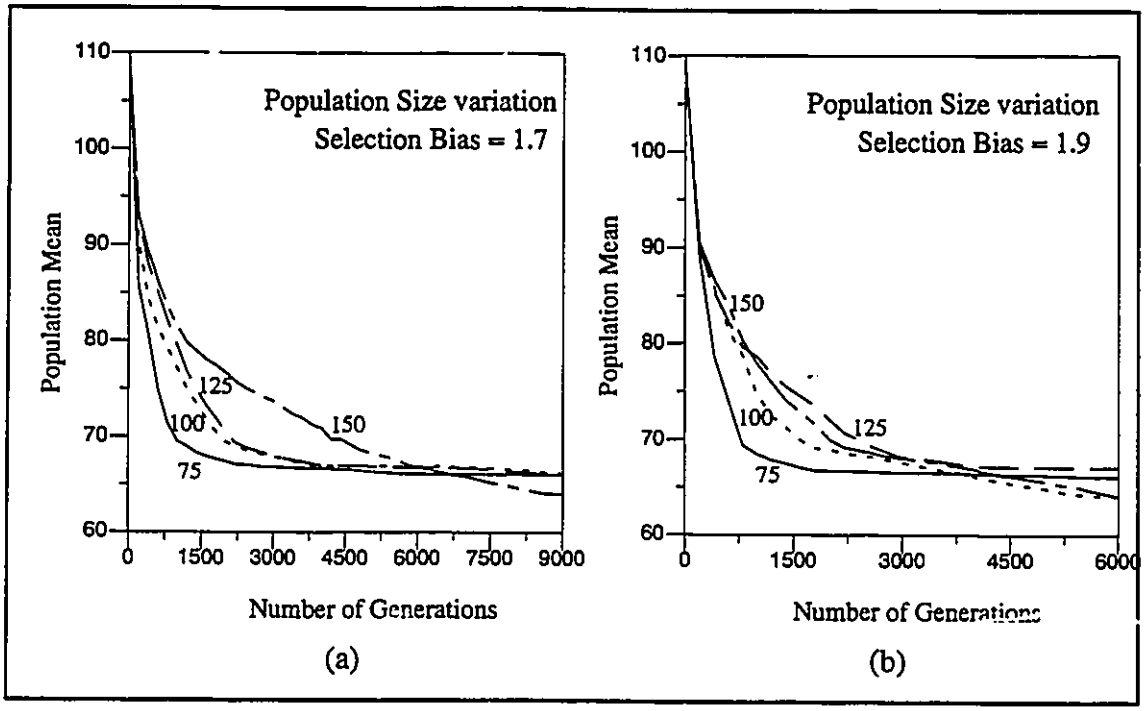


Figure 4.9a: Population mean variation for Problem A1

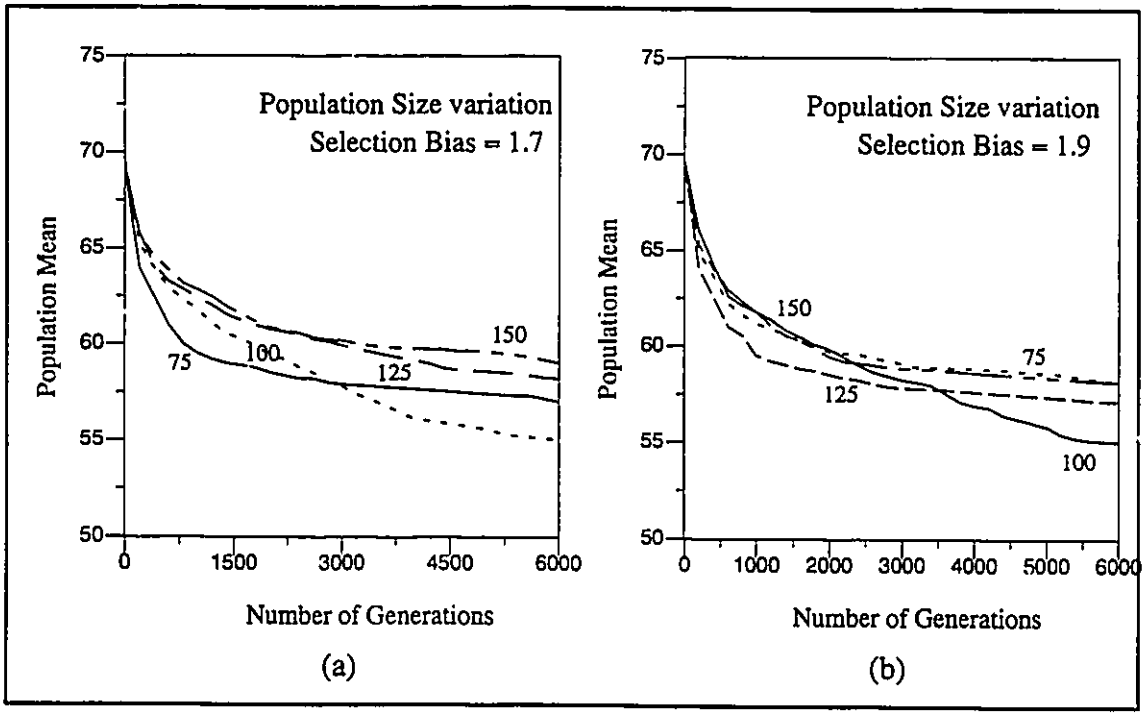


Figure 4.9b: Population mean variation for Problem A2

for the chosen problems. However, depending on the problem, multiple runs may be a good strategy for obtaining the best parameter values.

The Gantt charts of the best schedules obtained through genetic algorithms for problems A1 and A2 are shown in Figure 4.10 and 4.11. It was also noticed from the results

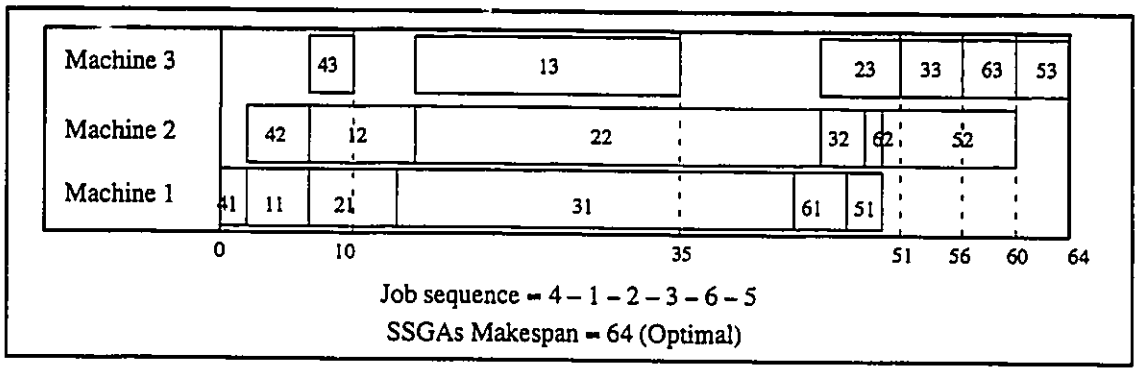


Figure 4.10: Gantt Chart for problem A1 using SPPS GAs

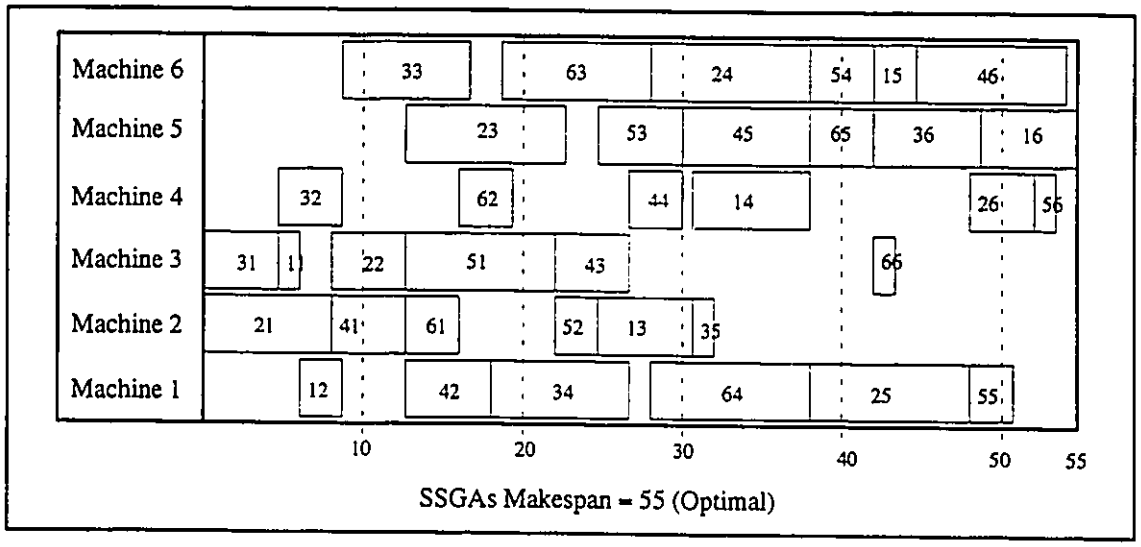


Figure 4.11: Gantt Chart for problem A2 using SPPS GAs

that for problem A1 Gupta [1972] and Chan [1989] both reported an optimal value of 68.0 whereas by using a genetic algorithms approach, a value of less than 68.0 is always obtained no matter what the parameter value settings were.

4.5 Computational Experience

In this section, the performance of steady–state genetic algorithms is evaluated against a number of existing procedures. In order to test the validity of the results obtained by genetic algorithms, seventeen examples of various sizes were chosen from the published literature and run using genetic algorithms. This section provides the comprehensive results of the experimentation in terms of nearness to the desired solution and time it takes to reach this solution. Details of all these problems can be found in Appendix A. The computational experience for all these problems are also discussed.

4.5.1 Validation of Results

A 'C' implementation of steady–state genetic algorithms was tested on all the problems using the following genetic algorithms parameters:

(i) Population Size (PSIZE): 100

(ii) Selection Bias (SBIAS): 1.9

(iii) Number of Generations (NGEN): 10,000

The results are summarized in Table 4.3 and Table 4.4. For each problem chosen, the table shows the number of jobs, number of machines, performance criterion used, reported objective function value, value obtained using genetic algorithms, percentage deviation between these values, percentage deviation between the CPU time and the reference number from which the example was obtained.

For some of the problems the CPU time for the algorithms was not reported. Problems P1, P2 and P3 were simple two–machine problems for which optimal solutions were known. These problems were used to test the working of SSGAs and for all three problems optimal solutions were obtained. Problem P4 was acquired from Sridhar and Rajendran [1993] where a simulated annealing approach was used to schedule five parts on three machines. They used minimization of total flow time as the criterion. The solution found by using

genetic algorithms was 320, which was the reported optimal. Problem P5 was first used by Gupta [1972] to test the heuristics he developed for the scheduling problems. He used minimization of makespan as the criterion. However, the value obtained by the algorithm was only 68.0 (reported optimal 64.0). Subsequently, the same example was used by Chan [1989] to test the heuristics. Chan reported a makespan value of 68.0 and a mean flow time value of 37.0 whereas from the provided Gantt chart, the calculated value of makespan was 70.0 and the mean flow time value of 38.6. However, in both cases the genetic algorithms approach fared better and was able to obtain an optimal makespan value of 64.0 and a slightly better value (38.5) in the case of mean flow time.

Problem P6 was obtained from Chan [1989] where mean flow time was used as the performance criterion and the relative percentage deviation found was 2.2%. Problem P8 was taken from Rajendran and Chaudhuri [1991] where the authors used minimization of total flow time as the performance criterion. The heuristics they provided were able to obtain a value of 384.0 for the total flow time. By using the GAs approach, a value of 370.0 was obtained resulting in 3.6% relative gain in the performance. Problems P7, P9, P10 and P11 were all taken from Lee [1987] where an expert systems approach was used for scheduling the manufacturing systems. It was observed that for the examples used, the GAs approach was able to achieve better or same performance values reported by the author. In all of these examples except P7, Lee used four dispatching rules for scheduling the parts. For problem P9, Lee reported the best mean flow time value of 34.0, which was obtained using LST (least slack time) dispatching rule. The other rules could not obtain as good a result. By using the genetic algorithms approach, the best value of 34.0 was obtained without considering any dispatching rule. Naturally, in these cases, the proven optimality of the solution has eliminated the need to apply dispatching rules. In problem P10, a modified version of genetic algorithms was used where dispatching rules were incorporated in the evaluation function and was tested on two different performance criteria, namely

minimizing the mean flow time and mean tardiness. It was observed that in all cases, the genetic algorithms approach fared better than the expert systems approach.

Problems P12 and P13 were obtained from Chan [1989] and were solved by heuristic algorithms and evaluated using simulation programs. For both problems, a better value for the objective function was achieved using the proposed genetic algorithms approach. Problem P15 was a real industrial example obtained from Logendran and Nudtapon [1991] in which the genetic algorithms approach was successfully applied to achieve optimal results.

In all of the above approaches, the CPU time actually used to solve the problems was not reported. Traditionally, the two main criteria by which computational results are compared are [Vancheeswaran and Townsend 1992]: (a) the best value determined by the algorithm for one or more benchmark problems, and (b) the run time required to arrive at the best solution on the computer employed, or the number of iterations required to compute the best value. The genetic algorithms approach was tested on two benchmark problems (six–jobs/six–machines and ten–jobs/ten–machines [Muth and Thompson 1963]) that are usually used to test the performance of the scheduling algorithms. Scheduling data for these problems are given in Appendix A. These problems were solved by various researchers using different computers. Table 4.4 lists the computer employed and results achieved for these problems. Best value results for these problems have been reported for heuristic methods and optimization methods, and these will provide our main basis for comparison. Comparing the efficiency of the algorithms in terms of the time it takes to achieve the desired solution is based on the computer employed. In this thesis, SUN computer was used to solve these two benchmark scheduling problems. The CPU seconds required by each computer is shown in Table 4.4.

Problem P16 is a classical six–jobs/six–machines benchmark problem, where, minimizing the makespan is the performance criterion and for which the optimal solution

is already known to be 55.0. As can be seen from the table, the genetic algorithms approach was able to obtain this optimal value in less time than was actually reported by others. The type of computers used in various applications is shown, however, it is recognized that several factors such as memory requirements, hardware configuration, disk size, loading point, etc. can affect the CPU time. Nevertheless, the information provided in Table 4.4 regarding the CPU time can be helpful as a basis for comparison. It was observed that CPU time of GAs approach was better than other reported CPU times. The percent saving in computation time was observed to be more than 50% in some cases.

The other benchmark problem is a notorious ten–jobs/ten–machines scheduling problem [Barker et. al. 1985] that has defied solution for many years, despite the fact that every available algorithm has been tried on it. Over the years, better and better solutions were discovered, usually in computer runs that took several hours and generated ten of thousands of search tree nodes. The best found reported optimal for this problem is known to be 930, reported by Adams et al. [1988] and by Carlier and Pinson [1989]. Adams et al. used a shifting bottleneck procedure to solve this problem and their algorithm took just over five minutes of VAX 780/11 time to find the optimal solution. Carlier and Pinson reported that after a long run that generated 22,000 nodes in five hours on a Prime 2655 computer, this solution proved to be optimal. This was the only problem for which the GAs approach was not able to achieve the optimal results. A makespan value of 995.0 was achieved by using this approach, losing only 6.9% of the optimal value. On the other hand, the saving in computation time is greater than 50.0% over several other reported results.

4.6 Summary and Conclusions

In this chapter, a genetic algorithm model was developed and tested for scheduling problems when there is no routing flexibility in the system. The effectiveness of the GAs approach was proved by comparing the results obtained by the proposed approach with other

Table 4.3: Experimental results of the test problems with SPPS GAs

P	N	M	Method	Performance Measure	Value		D (%)	CPU Seconds		Reference No.
					Reported	GA		Reported	GA	
1	2	2	MIP Formulation	Makespan	8	8 †	0.0%	16	0.02	[42]
	3	2	MIP Formulation	Makespan	8	8 †	0.0%	143	0.03	
2	7	2	Johnson's Algorithm (special case)	Makespan	36	36 †	0.0%	N/A	0.03	[645
3	9	2	Johnson's Algorithm	Makespan	44	44 †	0.0%	N/A	0.04	[76]
	14	2	Johnson's Algorithm	Makespan	115	115 †	0.0%	N/A	0.06	
4	5	3	Simulated Annealing	Flow Time	320	320	0.0%	N/A	0.02	[99]
5	6	3	Heuristics	Makespan	68.0	64.0 †	+5.9%	N/A	0.62	[47]
			Heuristics	Makespan	68.0	64.0 †	+5.9%	N/A	0.62	[17]
				Mean Flow Time	37.0	38.5	+0.4%	N/A	1.62	
6	4	4	Heuristics	Mean Flow Time	69.75	68.25	+2.2%	N/A	0.05	[17]
7	4	4	Artificial Intel./ Expert Systems	Makespan	37	37	0.0%	N/A	0.8	[69]
8	5	4	Heuristics	Total Flow Time	384	370	+3.6%	N/A	3.2	[86]

COMMENTS:

% deviation = (New Value – Old Value) x 100 / Old Value

LEGEND:

P – Problem Number
N – Number of Jobs
M – Number of Machines
N/A – Not Available
† – Reported Optimal
D – Percent Deviation (Positive deviation shows that values obtained through GAs are better)

Table 4.3: Experimental results of the test problems with SPPS GAs (Cont'd.)

P	N	M	Method	Performance Measure	Value		D (%)	CPU Seconds		Reference No.
					Reported	GA		Reported	GA	
9	5	4	Artificial Intel./ Expert Systems	Mean Flow Time	LST 34	34	0.0%	N/A	0.44	[69]
					EDD 38		+10.5%			
					SPT 42		+19.0%			
					LPT 40		+15.0%			
10	5	4	Artificial Intel./ Expert Systems	Mean Flow Time	LST 34	32	+5.9%	N/A	0.20	
					EDD 38	32	+15.8%			
					SPT 42	32	+23.8%			
					LPT 40	40	0.0%			
				Mean Tardiness	LST 4.0	0	+100.0%	N/A	0.20	
					EDD 2.0	0	+100.0%			
					SPT 3.2	0	+100.0%			
					LPT 9.6	0.8	+91.6%			
11	9	4	Artificial Intel./ Expert Systems	Mean Flow Time	LST 21	17	+19.0%	N/A	0.21	
					EDD 23	20	+13.0%			
					SPT 23	20	+13.0%			
					LPT 21	17	+19.0%			
12	4	5	Heuristics	Mean Flow Time	112.0	105.0	+6.25%	N/A	0.11	[17]
				Makespan	42.0	37.0	+11.9%	N/A	0.10	
13	4	6	Heuristics	Mean Flow Time	130.75	129.0	+1.3%	N/A	0.25	[17]
14	5	7	Heuristics	Makespan	569.58	562.31	+1.3%	N/A	3.10	[73]
15	8	7	Heuristics	Makespan	51330	51330	0.0%	N/A	0.03	

Table 4.4: Experimental results of two benchmark test problems with SPPS GAs

Performance measure: Makespan

Method	Computer Type	Value (V)		D (%)	CPU Seconds (T)		G	Reference No.
		V_R	V_{GA}		T_R	T_{GA}		
Problem P16: six–jobs/six–machines								
Branch and Bound	Pascal on Cyber 171	55.0	55.0 †	0.0%	1.50	0.45	3.33	[9]
Heuristics	Fortran on VAX 780/11	55.0	55.0 †	0.0%	1.50	0.45	3.33	[1]
Heuristics	Fortran on VAX 8550	60.0	55.0 †	+8.3%	0.80	0.45	1.78	[117]
Branch and Bound	Fortran on PRIME 2655	55.0	55.0 †	0.0%	1.00	0.45	2.22	[16]
Genetic Algorithms	N/A	55.0	55.0 †	0.0%	N/A	0.45	–	[81]
Heuristics	N/A	57.0	55.0 †	+3.5%	N/A	0.45	–	[108]
Simulated Annealing	VAX 785	55.0	55.0 †	0.0%	52.0	0.45	115.56	[68]
Heuristics	Modula2 on IBM PS/2	58.0	55.0 †	+5.2%	11.03	0.45	24.51	[51]
Problem P17: ten–jobs/ten–machines								
Branch and Bound	Pascal on Cyber 171	960	995	–3.6%	193.0	30.0	6.43	[9]
Heuristics	Fortran on VAX 780/11	930 †	995	–6.9%	851.0	30.0	28.37	[1]
Heuristics	Fortran on VAX 8550	1092	995	+8.8%	23.5	30.0	0.78	[117]
Branch and Bound	Fortran on PRIME 2655	930 †	995	–6.9%	3305.0	30.0	110.17	[16]
Genetic Algorithms	N/A	965	995	–3.1%	N/A	30.0	–	[81
Heuristics	N/A	1159	995	+14.2%	N/A	30.0	–	[108]
Simulated Annealing	VAX 785	933	995	–6.6%	57,772	30.0	1925.73	[68]
Heuristics	Modula2 on IBM PS/2	988	995	–0.7%	90.78	30.0	3.03	[51]

V_R = Reported makespan
V_{GA} = Makespan with GA
† = Optimal makespan

T_R = Reported CPU time
T_{GA} = CPU time on SUN Computers

D = % gain (+) or loss (–) in makespan value
G = CPU time improvement factor (T_R/T_{GA})

approaches used for scheduling. The results show that by using the GAs approach the saving in computation time is in the order of 50% or more. It was observed that the degree of difficulty in solving a scheduling problem by GAs increases with the number of machines. However, for a given number of machines, an increase in the number of jobs does not seem to increase the scheduling difficulty. On the contrary, the computational effort as well as the quality of solutions both showed improvements. For all the example problems except the notorious ten–jobs/ten–machines case, better or optimum results were obtained. For the ten–jobs/ten–machines case, the solution found was not far from optimality. The solution achieved for this problem was within 7 percent of the best known solution and was obtained in significantly reduced CPU time.

As far as GAs parameters are concerned, experiments showed that high selection bias (> 1.5) and a medium population size (= 100) sufficed in all cases. However, as pointed out by Grensfette [1986] and Schaffer [1989], multiple runs for different combinations of selection bias and population size are the best strategy for fine tuning the parameter selection.

CHAPTER 5

MULTIPLE PROCESS PLAN SCHEDULING

In this chapter, a genetic algorithm model is developed for the case when there are alternative process plans available for each part. Genetic algorithm parameters are selected by extensive experimentation, as in the previous chapter. The task of schedule optimization becomes more tedious when there are multiple process plans available for a given job. The multiple plans give rise to routing flexibility, because of which the possible number of feasible schedules increases. Dispatching rules are employed for the multiple process plan scheduling (MPPS) problem to alleviate the problem of finding an optimal schedule. Consideration of dispatching rules, order sizes greater than one, and alternate routings during scheduling with genetic algorithms constitute the major focus of this chapter. Finally, a detailed example is given where the performance of various scheduling rules with respect to different performance measures are presented.

5.1 Routing Flexibility

In many cases, it is possible to manufacture a product on more than one machine. These products have multiple process plans and since there is a choice of an alternate machine, this results in routing flexibility. This has the potential of increasing the output by eliminating bottlenecks which are usually present when there are no alternate routings. Because of the routing flexibility, the feasible number of schedules grows exponentially and finding a satisfactory schedule among these schedules becomes a formidable task. This is

shown in Figure 5.1., which shows two parts each having three operations. Operation 1 of

	Part1			Part2		
Operations	O11	O12	O13	O21	O22	O23
	M1	M2	M3	M2	M3	M1
Machines	M3	M4	M1	M4	M1	M3
	M4				M4	

Figure 5.1: An example of flexible routing

part 1 can be performed on machine 1, 3 or 4. Operation 2 can be performed on machine 2 or 4, and so on. In this case it is assumed that the operations must be performed in a predetermined linear sequence, i.e., the sequence of operations remains fixed while the sequence of machines on which these operations are performed varies. One possible sequence of machines in this case would be M1 – M2 – M3 – M2 – M3 – M1. Another possible machine sequence would be M3 – M4 – M3 – M2 – M1 – M3. The total number of possible schedules in this case would be 144 (3 x 2 x 2 x 3 x 2 x 2).

5.2 Problem Description

The MPPS scheduling problem may be stated as: "Given alternative process plans (routing flexibilities) for each part, the objective is to find a feasible schedule for a given set of jobs such that some given performance criterion is optimized". An important issue addressed in this chapter is the use of dispatching rules with genetic algorithms to schedule the jobs on to the machines in the presence of multiple routing flexibility. The prerequisites of this stage will be the process plan for each job, which includes: the number and type of jobs, number of tasks in each job, number and types of machines available, processing and setup times of tasks on machines, order due dates, release time of jobs into the shop floor

and performance criteria to be chosen. These characteristics were discussed in Section 4.1. An important prerequisite for multiple process plan scheduling is the selection of dispatching rules. In this research, twelve different dispatching rules are employed for the MPPS problem and performance of which are judged against various performance measures. The performance measures and scheduling rules considered in this stage are given in the following subsection.

5.2.1 Performance Measures and Scheduling Rules

The most often used measurement criteria for selecting scheduling rules are classified according to the processing time, waiting time and due dates. Typical measures chosen to evaluate scheduling rules in our research are as follows:

i) Minimize mean and maximum completion time

$$\text{Mean Completion Time} = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=J_i} C_{ij} \tag{5.1}$$

$$\text{Maximum Completion Time} = \max\ (C_{ij}),\ \forall \quad \text{i, and j} = J_i \tag{5.2}$$

ii) Minimize mean and maximum waiting time

$$\text{Waiting Time} = W_{ij} = \sum_{j=J_i} C_{ij} - \sum_{j=0} r_{ij} - \sum_{j=0}^{J_i} t_{ij}\ , \quad \forall\ i \tag{5.3}$$

$$\text{Mean Waiting Time} = \frac{1}{n}\sum_{i=1}^{n}\ [\ W_{ij}\ ] \tag{5.4}$$

$$\text{Maximum Waiting Time} = \max\ (W_{ij}),\ \forall \quad \text{i, and j} = J_i \tag{5.5}$$

iii) Minimize mean and maximum tardiness; tardiness is the positive difference between the jobs completion time and its due date.

$$\text{Tardiness} = T_{ij} = \sum_{j=J_i} C_{ij} - d_i \ , \ \forall \ i \tag{5.6}$$

$$\text{Mean Tardiness} = \frac{1}{n}\sum_{i=1}^{n} [\ T_{ij}\ ] \tag{5.7}$$

$$\text{Maximum Tardiness} = \max\ (T_{ij}),\ \forall \quad \text{i, and j} = J_i \tag{5.8}$$

iv) Maximize average resource utilization; the percent utilization of individual machine is calculated based on the maximum flow time, i.e., for a schedule, the individual and average resource utilization is calculated as follows:

$$\text{Utilization} = U_k = \frac{\textit{Total busy time}}{\max\ (\ C_{ij}\ )}, \quad \forall \quad k \tag{5.9}$$

$$\text{Average Utilization} = \frac{1}{m}\sum_{k=1}^{m} U_k \tag{5.10}$$

Other performance criteria such as reducing the mean or maximum earliness, etc. can also be implemented while scheduling with GAs. All the criteria are evaluated in the evaluation function of the genetic algorithms. The evaluation function gives a complete schedule and an associated performance criterion value. The earliness of a job can be handled in the same way as tardiness, where a penalty is assigned for completing the jobs before its due date.

An important issue addressed at this stage of the research is the consideration of different scheduling rules in the presence of multiple process plans. It is observed from the literature review that scheduling rules were usually used without alternate routings. A scheduling rule is used to select the next job to be processed from a set of jobs awaiting service at the machine. The scheduling rules used for the MPPS problem are: i) SI, ii) SPT, iii) LI, iv) LPT, v) EDD, vi) OPNDD, vii) LST, viii) LSO, ix) SLACK/RP, x) SLACK/RO, xi) SRPT, and xii) FCFS.

5.2.2 Modeling Assumptions

Most of the scheduling work done so far is based on certain assumptions some of which are not practical in real life. Some of the assumptions made at this stage of the research are:

i) A task can be performed by one or more different (alternate) machines.

ii) Machines can be loaded and unloaded at any time during the production (set–up, for example, is not restricted to a particular shift).

iii) Jobs are independent and consist of strictly ordered operation sequences.

iv) Task times are deterministic and are known in advance. Because the entire system is under rigid computer control, the processing times for individual tasks can be considered as deterministic.

v) Each order has a due date representing the desired completion time.

5.3 Genetic Algorithms Design Issues for MPPS Problems

This section investigates the application of genetic algorithms in scheduling in the presence of multiple process plans. The complexity of scheduling problems increases in this case because there are several ways by which a part can be produced. Scheduling rules are employed in the evaluation function to reduce this complexity. The main design issues for genetic algorithms were introduced in chapter 4 and are the same for the genetic algorithms used for the MPPS problem. They are: (i) schedule representation, (ii) initialization, (iii) evaluation function, (iv) recombination operators, and (v) parameter values. Issues (ii) and (v) work the same way as for SPPS problems while issues (i), (ii) and (iv) differ in some ways. This is explained in the following subsections.

5.3.1 Schedule Representation

In this chapter, the sequence of operations is assumed to be fixed. In the case of multiple routing flexibility each operation has a choice of one or more machines. The

schedule will vary according to the operation–machine assignments. The GAs model employs a list of machines as a chromosome, i.e., a schedule. Different combinations of operation–machine assignments represent different production schedules and are used in the genetic population. This is shown in Figure 5.2.

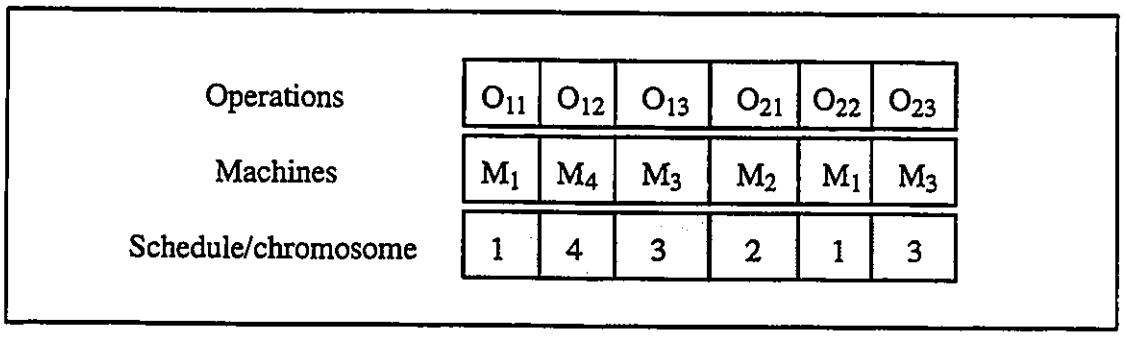


Figure 5.2: Schedule Representation for MPPS GAs

5.3.2 Evaluation Function

The fitness value of each schedule is calculated using the following algorithm:

i) Calculate ready time (arrival time) for all operations.

Let $t = 0$, and $a_k = 0$ (k = 1, ..., m)

Beginning with S as an empty set, while S' includes all the operations.

Set $r_{i1} = 0$, for all i

$r_{ij} = r_{i(j-1)} + t_{i(j-1)k}$, for all $i = 1, 2, ..., n$, and $j = 2, 3, ..., Ji$

ii) If there is any $[O_{ij} \in S' \mid r_{ij} \leq t]$, go to step iii).

Otherwise: Set $t = \min (r_{ij})$

iii) Place all $[O_{ij} \in S' \mid r_{ij} \leq t]$ in a set S''.

iv) For each operation $O_{ij} \in S''$, calculate a priority index according to a specific scheduling or priority rule. Assign the operation with the smallest index to respective machines and calculate the completion time of each operation, i.e.

$C_{ij} = \max (r_{ij}, a_k) + t_{ijk}$

v) For each operation in the set S'':

(a) Remove the operations, O_{ij}, from the set S' and the set S''.

(b) Place O_{ij} in the set S.

(c) Determine:

$$S_{ij} = \max \{r_{ij}, a_k\}$$

$$C_{ij} = S_{ij} + t_{ijk}$$

$$r_{i(j+h)} = r_{i(j+h)} + [C_{ij} - r_{i(j+1)}] \text{ for } h = 1, 2, ..., (J_i - j)$$

(d) Set $a_k = C_{ij}$.

vi) Check if S' is empty; if it is not, then go to (ii), and repeat the procedure until the schedule is complete.

5.3.3 Recombination Operators

The main difference between the genetic algorithms used for the SPPS and MPPS problems lies in the genetic recombination operators used. In the case of the MPPS problem, two genetic operators are used to vary the schedule chromosome from generation to generation. The edge recombination operator could not be used in this case because of the fact that there are duplicate nodes in the chromosome. An edge recombination operator requires that all the nodes or genes in a chromosome be different. In the case of genetic algorithms used for an MPPS problem, the population has chromosomes or schedules in which there are identical nodes or genes (in our case machines). For example, in Figure 5.2, nodes (machines) 1 and 3 appear twice and hence the edge recombination operator cannot be used for such schedules. Two new operators are used for genetic recombination to overcome this problem. They are: (i) reduced surrogate crossover, for crossing two parent schedules and (ii) adaptive mutation, for random introduction of new genes in the chromosome (schedule). The other recombination operators may result in an invalid machine assignment and therefore infeasible schedule.

5.3.3.1 Reduced Surrogate Crossover

Crossover is usually implemented by choosing one crossover point at random, then exchanging segments between the two parent strings, thus forming new children which contain information from each of the two parent chromosomes. The problem of using such a crossover is that it can easily create duplicate chromosomes (schedules) as indicated in Figure 5.3. Reduced surrogate crossover was introduced by Booker [1987]. With this operator, the offsprings are guaranteed not to be duplicates of the parent chromosomes (schedules). Reduced surrogate crossover first identifies all positions where the parent strings differ. Crossover points are only allowed to occur in these positions. The workings of reduced surrogate crossover is given in Figure 5.3.

5.3.3.2 Adaptive Mutation

The other operator that is used in the genetic algorithms for an MPPS problem is adaptive mutation [Whitley, 1988]. Adaptive mutation bases the amount of disruption to a given string on two factors: the relative similarity of its two parent strings, and a mutation rate. The more similar the two parent strings, the more likely mutation is to occur. The actual mutation which occurs is the product of this similarity and a fractional mutation rate. Thus, if the mutation rate is 0.20 and the parents of a particular string were identical, approximately 20% of the string will be mutated. Or, if the parents contained 50% unique information, only about 10% of the string will be mutated.

5.4 Genetic Algorithms Parameter Setting for MPPS Problems

As discussed in chapter 4, selecting a best combination of GAs parameters is the most difficult and time consuming task. The parameters of genetic algorithms used for the MPPS problem are population size (PSIZE), selection bias (SBIAS) and the adaptive mutation (AMUT) rate. To find the best parameter values, extensive experiments were conducted with different combinations of parameter values. The parameter values used for

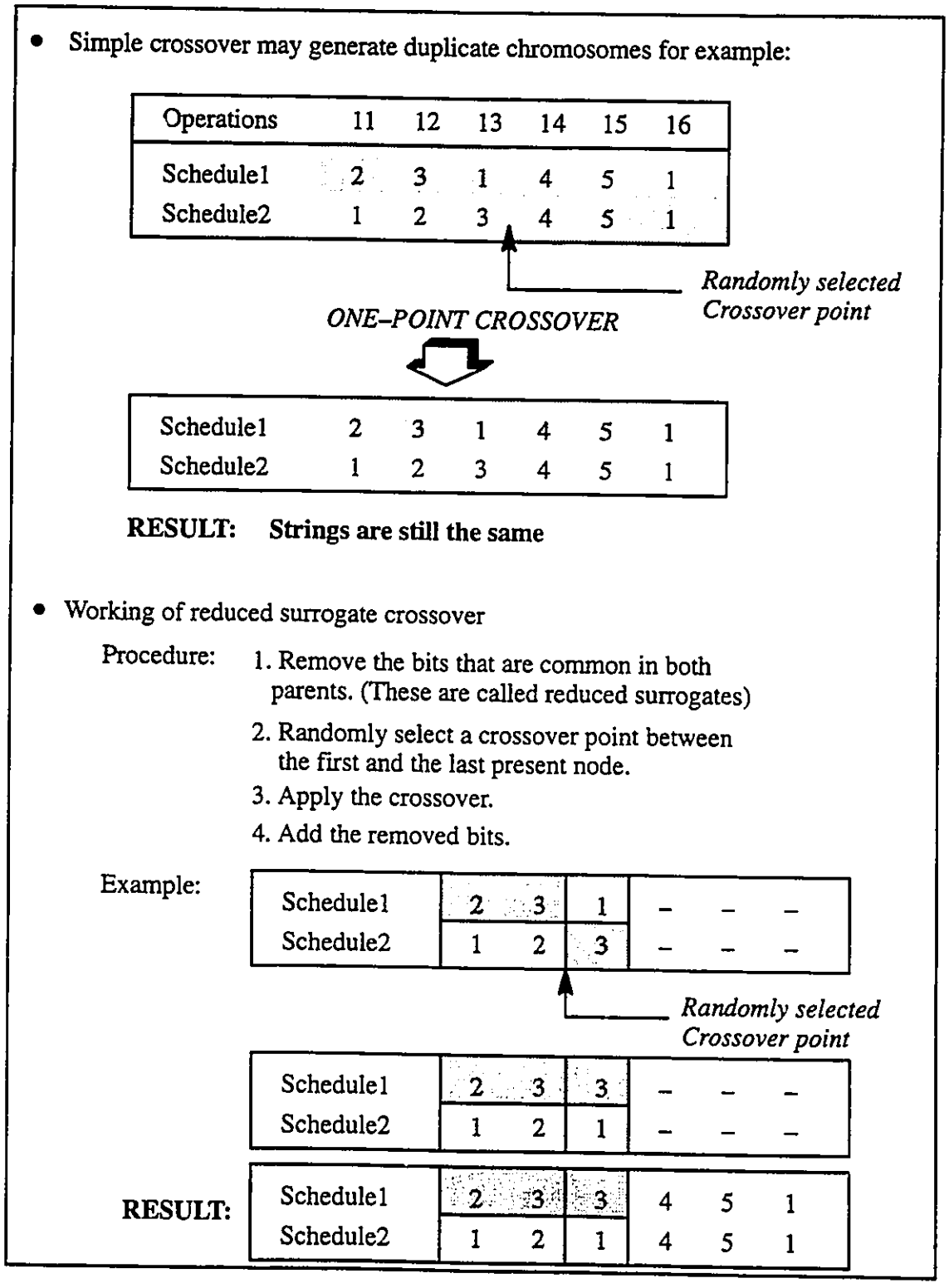


Figure 5.3: Working of reduced surrogate crossover operator

experimentation are given in Figure 5.4. Two sets of population size levels were used for

Parameters	Levels						
Population Size (PSIZE)	30	40	50	60	70	80	(for problem B1)
	40	80	120	160	200		(for problem B2)
Selection Bias (SBIAS)	1.1	1.3	1.5	1.7	1.9		
Adaptive Mutation (AMUT)	0.1	0.3	0.5	0.7	0.9		

Figure 5.4: Experimental design parameters for MPPS GAs

genetic algorithms because of the size of the problems. For small problems, population size (PSIZE) was varied from 30 to 80 with an interval of 10. For larger problems, it was varied from 40 to 200 with an interval of 40. The selection bias (SBIAS) was varied from 1.1 to 1.9 in the interval of 0.2 and the mutation rate was varied from 0.1 to 0.9 in the interval of 0.2. The number of generations was kept fixed at 10,000. For all parameters combinations, the experiments were stopped if the solution converged before reaching 10,000 generations. The first problem used for experimentation (problem B1) was small example with four jobs and six machines. This example was extracted from Nasr and Elsayed [1990] where each job has three operations be performed in a given sequence. Each operation has a choice of alternate machines on which the processing takes place. The authors used analytical formulation (developed as mixed integer programming) to solve the above problem. Their algorithm has two separate procedures, one for scheduling and another for conflict solving. The purpose of the conflict solving procedure is to obtain an assignment for jobs requiring the same machine at the same time. The characteristics of this problem is given in Appendix B.

The second problem (Problem B2) was chosen from Chryssloulis et al. [1992] with

some necessary additions such as setup times and due dates in order to compare the performance of different scheduling rules. This problem is a four–workcell production system that processes ten different jobs. The number of tasks in the job varies from one to five, some of which have a choice of more than one machine. All data sets for this problem are given in Appendix B. The task sequences and their quantity are listed in Table B.1. Machines available in each workcell and tasks that can be performed by them are summarized in Table B.2. There are five machines in a machining workcell, two in review and one each in the inspection and the wash workcells. All the machines in the machining workcell are capable of performing operations such as turning, drilling, facing, milling, boring and screwing. Tasks processing and setup times information are given in Table B.3. Due dates for each order are assigned based on the total work content of these orders as this is the most rational method of due date assignment [Cheng and Gupta, 1989]. Order due dates are set equivalent to (DDT x total processing time), where DDT is the due date tightness. In literature, a low value of 2 and a high value of 10 is usually used for tight and relaxed due dates. Since there are alternative process plans for the example problem, due date tightness value of 1.5 is used. The performance measures and scheduling rules listed in Section 5.2 are used for schedule evaluation.

Both the examples were run for different combinations of population size, selection bias and the mutation rate. The performance measure and the dispatching rule used were to minimize the mean flow time and the shortest imminent task processing time respectively. The search process begins with the proper representation of a schedule, generation of schedule population and evaluation of each schedule in the population, and application of genetic operators to this population for improving the schedules. The search process continues for a specified number of generations yielding an optimal or a near optimal solution. The results are tabulated in Table 5.1 and 5.2. The convergence trend for both problems are plotted in Figure 5.5 and Figure 5.6. For problem B1, it was observed

Table 5.1: Results of Problem B1

		Population Size (PSIZE)					
		30		40		50	
SBIAS	**AMUT**	**Value**	**GEN**	**Value**	**GEN**	**Value**	**GEN**
1.1	0.1	11.75	239	12.25	90	11.75	110
	0.3	12.00	129	11.75	120	11.75	160
	0.5	12.00	200	**11.50**	**90**	11.75	110
	0.7	11.75	238	11.75	40	**11.50**	**190**
	0.9	11.75	185	**11.50**	**150**	11.75	80
1.3	0.1	12.25	92	**11.50**	**140**	11.75	100
	0.3	12.00	251	**11.50**	**140**	**11.50**	**160**
	0.5	11.75	263	11.75	130	**11.50**	**120**
	0.7	12.00	130	11.75	100	11.75	220
	0.9	11.75	180	11.75	158	11.75	160
1.5	0.1	11.75	154	12.00	70	11.75	50
	0.3	11.75	142	11.75	110	11.75	140
	0.5	**11.50**	**201**	11.75	90	11.75	110
	0.7	11.75	179	11.75	30	11.75	140
	0.9	11.75	200	11.75	80	11.75	90
1.7	0.1	11.75	194	12.00	80	11.75	90
	0.3	12.00	113	11.75	100	11.75	70
	0.5	12.00	119	12.00	60	**11.50**	**140**
	0.7	12.25	107	12.00	80	12.00	140
	0.9	12.25	101	11.75	80	**11.50**	**130**
1.9	0.1	12.25	64	**11.50**	**130**	**11.50**	**80**
	0.3	12.00	81	11.75	70	**11.50**	**70**
	0.5	11.75	139	11.75	30	11.75	110
	0.7	11.75	138	11.75	60	**11.50**	**80**
	0.9	12.00	93	11.75	100	**11.50**	**100**

that the optimal value of the mean flow time is 11.50, which was obtained in several cases. The plots in Figure 5.5 are for an SBIAS value of 1.9 and an AMUT value of 0.1 and 0.3 respectively. Figure 5.6 gives the convergence trend of problem B2 for a SBIAS value of 1.9 and an AMUT value of 0.1 and 0.3 respectively. In all plots it is noticed that the curves

Table 5.1: Results of Problem B1 (Contd.)

		Population Size (PSIZE)					
		60		70		80	
SBIAS	**AMUT**	**Value**	**GEN**	**Value**	**GEN**	**Value**	**GEN**
1.1	0.1	**11.50**	180	11.75	150	**11.50**	**230**
	0.3	**11.50**	**160**	**11.50**	**270**	11.75	210
	0.5	**11.50**	**180**	**11.50**	**280**	11.75	150
	0.7	**11.50**	**140**	11.75	100	11.75	250
	0.9	11.75	160	**11.50**	**410**	11.75	270
1.3	0.1	**11.50**	**210**	**11.50**	**260**	**11.50**	**270**
	0.3	11.75	120	**11.50**	**240**	**11.50**	**330**
	0.5	11.75	110	11.75	130	**11.50**	**290**
	0.7	**11.50**	**190**	**11.50**	**230**	**11.50**	**310**
	0.9	**11.50**	**210**	**11.50**	**330**	11.75	250
1.5	0.1	12.00	140	11.75	130	11.75	200
	0.3	11.75	80	11.75	170	11.75	210
	0.5	11.75	120	**11.50**	**320**	**11.50**	**240**
	0.7	11.75	160	**11.50**	**140**	**11.50**	**170**
	0.9	12.00	60	**11.50**	**130**	11.75	210
1.7	0.1	**11.50**	**180**	12.00	110	11.75	130
	0.3	11.75	110	**11.50**	**250**	**11.50**	**190**
	0.5	12.00	110	11.75	110	**11.50**	**140**
	0.7	11.75	160	**11.50**	**150**	11.75	140
	0.9	12.00	130	11.75	170	11.75	180
1.9	0.1	11.75	30	11.75	130	11.75	120
	0.3	11.75	120	11.75	130	**11.50**	**160**
	0.5	11.75	120	11.75	170	11.75	160
	0.7	**11.50**	**250**	11.75	140	11.75	150
	0.9	11.75	110	**11.50**	**250**	11.75	80

start from a high value of population mean and converge down to an optimal or suboptimal value.

5.4.1 General Guidelines for Parameter Setting for MPPS Problems

It was observed that for problem B1, the solution converges to a value of 11.50, which

Table 5.2: Results of Problem B2

		Population Size (PSIZE)					
		40		80		120	
SBIAS	**AMUT**	**Value**	**GEN**	**Value**	**GEN**	**Value**	**GEN**
1.1	0.1	4176	540	4145	908	4080	2106
	0.3	4207	500	4121	3551	4173	1522
	0.5	4173	585	4051	1262	4064	4837
	0.7	4205	529	4029	1742	4085	1860
	0.9	4171	584	4174	739	4065	2626
1.3	0.1	4170	317	4101	1007	4123	1640
	0.3	4154	393	4121	1230	4096	1877
	0.5	4171	418	4152	1091	4046	2321
	0.7	4173	533	4206	1136	4115	1950
	0.9	4073	391	4149	2291	4068	2757
1.5	0.1	4179	309	4107	648	4026	1335
	0.3	4243	207	4182	741	4090	4888
	0.5	4135	343	4097	982	4121	1322
	0.7	4153	358	4082	794	4078	1454
	0.9	4133	809	4062	1093	4095	1268
1.7	0.1	4237	285	4178	864	4113	1094
	0.3	4145	226	4099	825	4074	2166
	0.5	4187	303	4076	886	4087	1572
	0.7	4183	384	4134	946	4097	2169
	0.9	4226	247	4099	764	4109	2071
1.9	0.1	4257	337	**4039**	**581**	4068	1200
	0.3	4128	1834	4188	646	4120	1226
	0.5	4134	243	4165	584	4096	2145
	0.7	4195	237	4052	1601	4131	1127
	0.9	4129	530	4221	861	4094	1437

seems to be the optimal solution. Since the problem size was small (4 jobs x 6 machines), population size was varied in the interval of 10 ranging from 30 to 80. Larger population sizes were also tried but no improvements were observed over this value. Larger populations took a larger number of generations, resulting in large computational time. With a

Table 5.2: Results of Problem B2 (Contd.)

		Population Size (PSIZE)			
		160		200	
SBIAS	**AMUT**	**Value**	**GEN**	**Value**	**GEN**
1.1	0.1	4031	2882	4059	2745
	0.3	4080	5050	4051	5073
	0.5	4087	5093	4054	4560
	0.7	4031	4997	4067	3982
	0.9	4041	6218	4065	4406
1.3	0.1	4118	3445	4073	2645
	0.3	4052	10,000	4104	4161
	0.5	4120	2296	4069	6167
	0.7	4092	2576	4076	5773
	0.9	4069	2543	4084	2867
1.5	0.1	4041	2180	4050	3664
	0.3	4101	2659	4066	4037
	0.5	4105	2649	4073	3785
	0.7	4080	2800	4077	3079
	0.9	4064	3179	4047	3859
1.7	0.1	4095	2787	4083	3857
	0.3	4071	1970	4121	3096
	0.5	4036	10,000	4057	3547
	0.7	4057	2134	4156	2988
	0.9	4050	3127	4063	3406
1.9	0.1	4085	2252	4006	3935
	0.3	4026	2437	4119	2578
	0.5	4051	2167	4047	1694
	0.7	4054	2541	4090	2306
	0.9	4066	2342	4042	2892

population size of 30, an SBIAS of 1.5 and an AMUT of 0.5 gave the best results and it took 201 generations to converge. The population size that took the least number of generations to reach the best value was 50. It took only 70 generations with an SBIAS value of 1.9 and an AMUT value of 0.3. Other values of population size of course attained the optimal value but it took more number of generations to converge. As far as selection bias (SBIAS) is

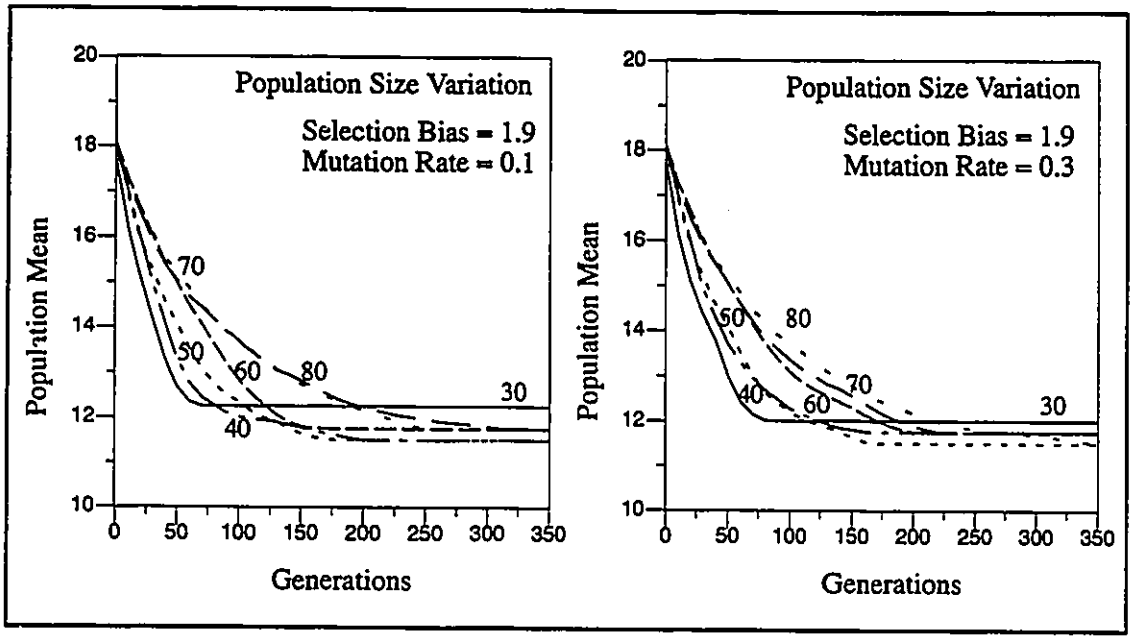


Figure 5.5: Population mean variation for problem B1

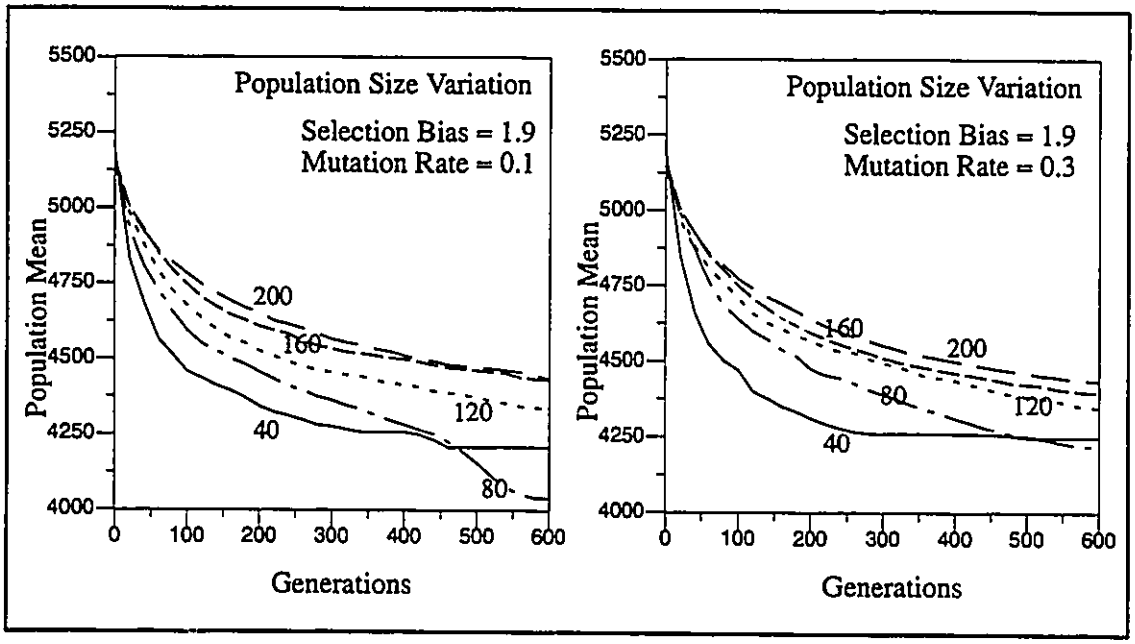


Figure 5.6: Population mean variation for problem B2

concerned, all values of SBIAS eventually yielded good results, some for a smaller population size and some for large population size. However, it is observed from the table that SBIAS value of 1.9 took the least number of generations (70) to reach to an optimal solution.

For selecting a mutation rate, it was observed that a low value of mutation rate gave the best performance. Again, comparing the number of generations it took to converge, it was observed that with a population size of 50, an AMUT value of 0.3 was the clear winner.

As indicated by several genetic algorithms researchers, these parameters are very much dependent on the chosen problem, and hence several experiments should be run before a decision can be made on exact parameter values. Since computation times for genetic algorithms are usually small, it may not be a difficult task to conduct experiments. From the results of problem B1, it can easily be concluded that for small–sized problems, a medium population size (50 – 60), high selection bias (>= 1.7) and low mutation rate (<=0.3) is probably the best choice.

The same trend was observed from the results of problem B2. Experiments were first conducted using the same population size as in problem B1. It was observed that the solution space provided by such a population did not give as good results because of the small solution space. This problem had more operations and relatively more machines resulting in a large solution space. Therefore, larger populations were tried varying from 40 to 200 with an interval of 40. For a population size of 40, the least value of mean flow time observed was 4073, which was attained with SBIAS value of 1.3, an AMUT value of 0.9 and an NGEN value of 391. All other values for this population size had values gretaer than 4100. Furthermore, with a population size of 80, the least value was 4029 but it took a large number of generations to obtain this value. The least number of generations in this category was 581, which gave a mean flow time of 4039. Since most of the values for this population size are in the range of 4000 to 4100 and converging in relatively less number of generations,

this population size seems to be the best one. Increasing the population size further increases the number of generations required to achieve similar results. The best value in all these experiments was 4006 when the population size was fixed at 200 with an SBIAS value of 1.9 and an AMUT value of 0.1. This combination of parameters took 3935 generations for convergence. Since the performance of any algorithm is judged by the speed, i.e., how fast the algorithm is converging, parameter values are selected to reduce degradation in the performance (in terms of objective function value). For a population size of 80, the observed value was 4039 (losing only 0.83%) with a large gain in computation time (85.24%). As mentioned before, running several experiments is the best strategy to obtain the best parameter combination. Based on the results obtained, a medium population size, high selection bias, and low mutation rate are recommended as a starting point. However, for large problem sizes, a population size of 80, an SBIAS value of 1.9 and an AMUT value of 0.1 are recommended.

The Gantt charts for the schedules obtained through genetic algorithms for problems B1 and B2 are shown in Figure 5.7 and 5.8 respectively.

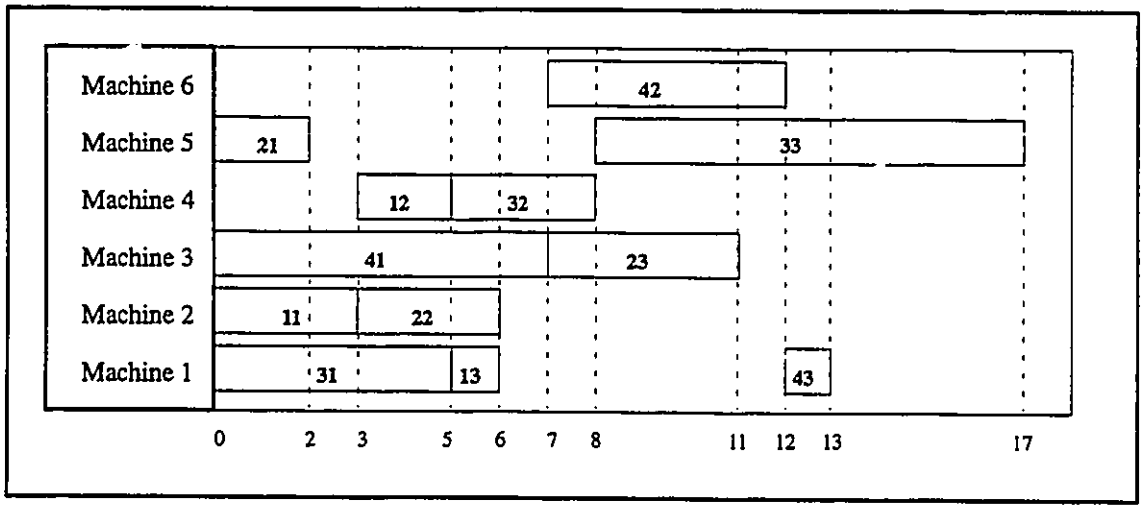


Figure 5.7: Gantt Chart for problem B1 using MPPS GAs

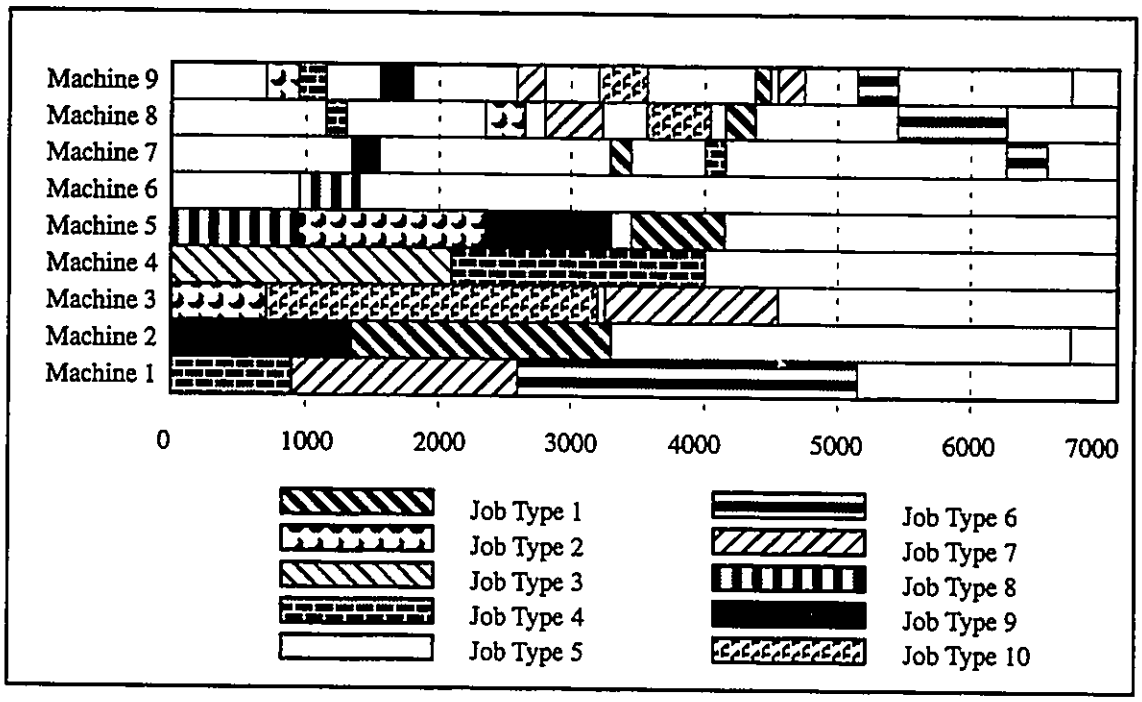


Figure 5.8: Gantt Chart for problem B2 using MPPS GAs

5.5 Computing Issues

In this section, the performance of genetic algorithms for multiple process plan scheduling is evaluated against two other existing approaches. These problems were chosen from published literature and are given in Appendix B.

5.5.1 Validation of Results

The genetic algorithms parameters that were chosen for validating the results are as follows:

(i) Population Size (PSIZE): 80

(ii) Selection Bias (SBIAS): 1.9

(iii) Adaptive Mutation rate (AMUT): 0.1

The first problem was taken from Nasr and Elsayed [1990] in which they solved the

problem of multiple routing scheduling using a bound algorithm. The best found schedule using bound algorithm for mean flow time was reported to be 12.25 units. By using genetic algorithms, the best found solution was 11.50 units.

The second example was extracted from Dutta [1990] who used an artificial intelligence approach to solve the problem. Dutta reported a value of 84.0 for the makespan whereas by using a GAs approach, a makespan of 76.0 was achieved. For both examples, the time taken to achieve the solution was not reported. Table 5.3 gives the comparison of results in terms of performance gain in the quality of solutions and time complexity.

Table 5.3: Result comparison with MPPS GAs

P	N	M	Method	Measure	Value			CPU Seconds		Ref.
					Reported	GA	D	Reported	GA	
1	4	6	Mixed Integer Programming	Mean Flow Time	12.25	11.50	6.1%	N/A	0.20	[82]
2	6	7	Artificial Intelligence	Makespan	84.0	76.0	9.5%	N/A	0.23	[29]

D = Percent Gain in Performance

5.6 Application of Scheduling Rules in Genetic Algorithms

To date scheduling with genetic algorithm has been limited to a single performance measure without using dispatching rules. A schedule is usually represented as a string of tasks and machines. A schedule is built from this string by allocating resources to the tasks one by one following the order in the string. The job of genetic algorithms is to generate different permutations of the orders and the corresponding machines. The use of dispatching rules provides an added advantage in that the search task is divided between the genetic

algorithms and the schedule builder. The genetic algorithm searches the space comprising all the permutations of tasks/machines. The schedule builder assigns the tasks based on machine availability and the dispatching rule. The schedule building is not limited to the ordering of tasks in the string. Given a performance measure and a scheduling rule, the genetic algorithm determines the best routing.

So far we have discussed the genetic algorithms parameter selection problem. In this section, the application of scheduling rules in genetic algorithms is discussed. In order to select the best scheduling rule to be used for a given performance measure, two issues are important; first, the choice of a set of genetic algorithms parameter values which is suitable to the problem, and second, the use of genetic algorithms with the chosen parameters to conduct exploratory tests for selecting the performance measure/scheduling rule. The genetic algorithms parameters selected to perform the tests for scheduling rules selection are as follows:

(i) Population Size (PSIZE): 80

(ii) Selection Bias (SBIAS): 1.9

(iii) Mutation Rate (AMUT): 0.1

The example chosen was problem B2 as this problem has a larger number of parts and more machines. Once the genetic algorithms parameters have been optimized, the next step is the analysis of scheduling rules with respect to certain performance measures. Table 5.4 summarizes the results of the experiments which compare the performance of twelve scheduling rules with respect to seven performance measures. The selected performance measures provide a comprehensive view of the performance of the system under the different policies. Figures 5.9a to 5.9g show the bar graphs of the each of the performance measures with respect to the scheduling rules used.

When the performance criterion is to minimize the mean flow time, SI, OPNDD, LST, and SRPT scheduling rules performed fairly well compared to other rules as can be seen

from Figure 5.9a. LI and LPT rules were the worst performing rules for mean flow time, unlike when the performance measure was to minimize maximum flow time. As can be seen from Figure 5.9b, the performance of all the rules was comparable except the SPT, EDD and LST scheduling rules. When mean tardiness was used as the performance measure, the SI, OPNDD, LST, and S/RP rules performed well compared to LI and LPT, which were the worst performing rules (Figure 5.9c). The LI and LPT rules also exhibited poor performance

Table 5.4: Performance measures verses scheduling rules

	Mean FT	Max FT	Mean T	Max T	Mean WT	Max WT	MU(%)
SI	4039	5858	89	624	1095	2260	61.9
SPT	4174	6038	137	795	1040	2365	55.9
LI	4336	5810	418	1889	1246	2798	54.2
LPT	4378	5756	515	1986	1353	3400	57.1
EDD	4132	6268	155	851	1034	2334	56.9
OPNDD	4160	5750	87	660	1163	2320	61.4
LST	4134	5964	109	821	1089	2146	59.2
LSO	4238	5810	181	798	1147	2222	63.1
S/RP	4274	5866	119	649	1178	2090	59.7
S/RO	4216	5798	138	620	1200	2250	59.0
SRPT	4188	5690	151	500	1091	2702	57.9
FCFS	4127	5814	225	959	1128	2594	57.2

FT = Flow Time, T = Tardiness, WT = Waiting Time, MU = Machine Utilization

when minimizing the maximum tardiness. The due date dispatching rules EDD and OPNDD worked well along with the SI, S/RO, S/RP and FCFS rules. When the mean waiting time was used as the measure, the SPT and EDD rules outperformed all other rules.

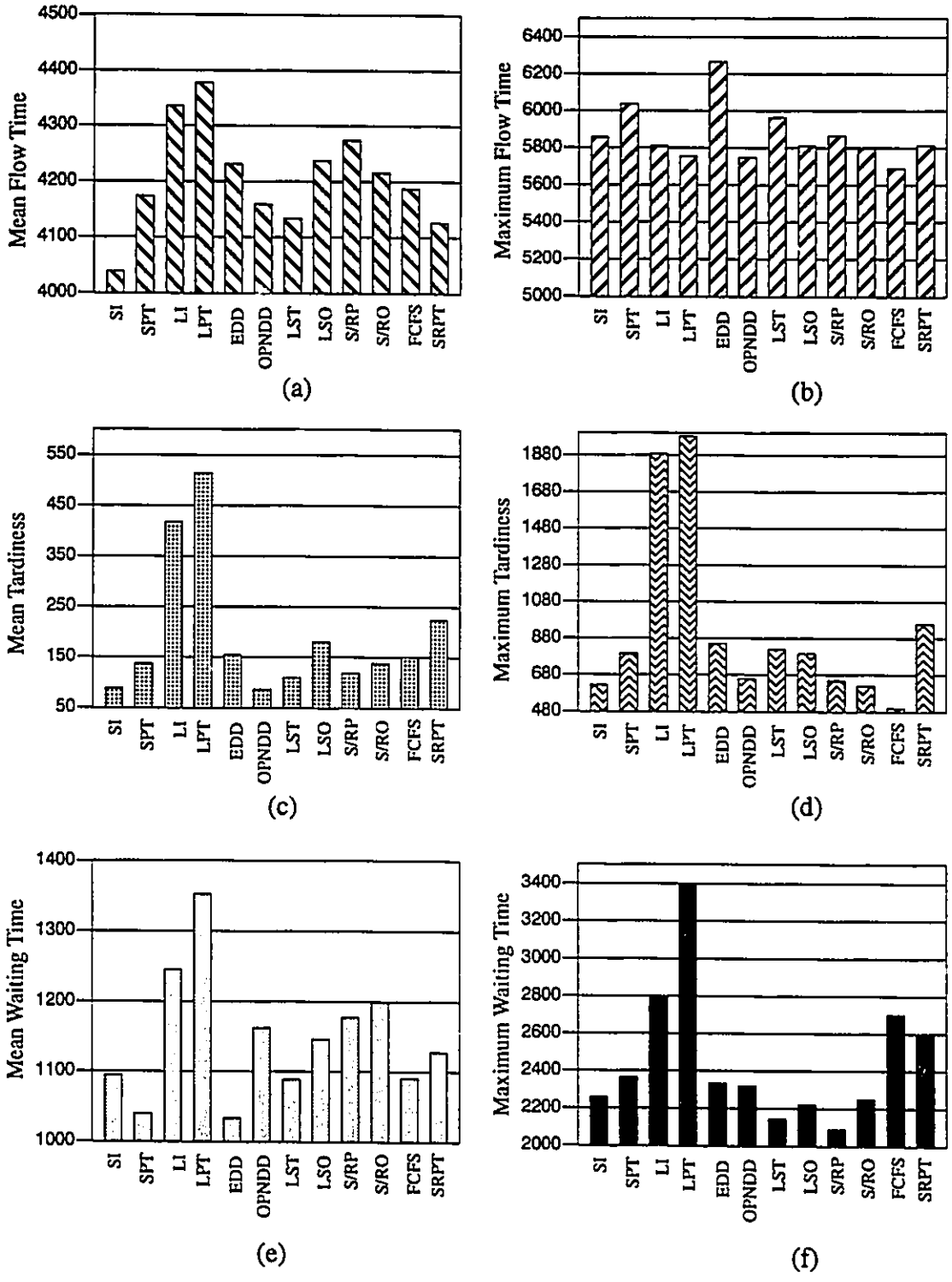


Figure 5.9: Performance measures versus Scheduling rules

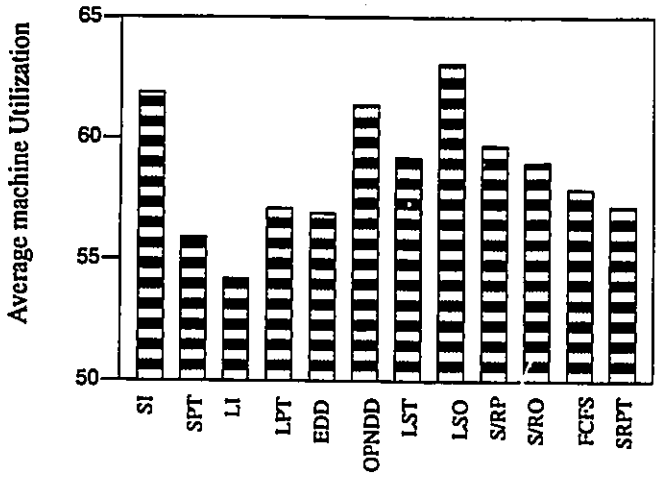


Figure 5.9g : Average machine utilization

The SI, LST and FCFS scheduling rules performed equally well and the LI and LPT were the worst performers. Similarly, while minimizing the maximum waiting time, the S/RP proved to be the best rule. From Figure 5.9g, the SI, OPNDD and LSO scheduling rules could easily be suggested for use when machine utilization is selected as a performance measure.

The use of a particular scheduling rule is very much dependent on the chosen performance measure and the production system tested [Montazeri et al. 1990]. For our example problem, the results show that all the rules except LI and LPT performed fairly well and can be used depending on the chosen criterion. The LI and LPT rules did not perform well at all except in the case of minimizing the maximum flow time. These results correspond to the conjecture provided by Blackstone et al. [1982].

In other scheduling research, it is often noticed that while testing the scheduling rules, authors tend to use a single job in each order which is rarely the case in practice. The difference in the results, i.e., whether the performance of one rule is better or worse than others, may be due to a special characteristic of our chosen problem, that is, each order has more than one part, and the tardiness is assessed according to when the order (not the jobs within an order) is completed. However, when throughput time and due dates are important

(which is often the case) SI, OPNDD and LST can satisfactorily be adopted as priority scheduling rules.

5.7 Summary and Conclusions

In this research, a new solution approach is applied to scheduling of manufacturing systems. Special aspects of the problems considered in this chapter are: (i) application of priority dispatching rules when using genetic algorithms for schedule optimization, (ii) the multiple identical jobs due to the order size being greater than one and (iii) alternative routings for each job.

We used a new solution approach for schedule generation and used various dispatching rules and performance measures. It is generally assumed that the processing time of a task is equal on all the machines within one workcell while solving the alternate routing problems. However, in actual systems the processing time may vary according to the adopted routes. This shortcoming is also addressed in this research where variable processing times have been considered.

CHAPTER 6

BATCH SPLITTING FOR SCHEDULING

This chapter presents the concept of splitting orders into batches to improve the scheduling performance of multi–product, multi–machine manufacturing systems. A review of literature reveals that queueing models have extensively been used for batch sizing and subsequent scheduling. In this research, an approach different from queueing models is used for splitting the batches. Assigning production tasks to the manufacturing resources is achieved through genetic algorithms. Six different batch–splitting policies are considered along with three job regeneration methods to improve the performance of manufacturing systems. In order to compare these batch–splitting policies, five different performance criteria are used with nine different dispatching rules. The results indicate that proposed batch–splitting policies improve the performance of the manufacturing system over the best performance obtained when there was no batch–splitting.

6.1 Introduction

Batch–splitting, machine grouping, tool loading, routing, part input sequencing, and scheduling are some of the factors that affect the performance of manufacturing systems. This chapter deals with batch–splitting and scheduling aspects of discrete manufacturing systems. Batch sizing pertains to the partitioning of the selected jobs into batches for simultaneous processing on a machine with a single setup. Batching is done in order to satisfy the due date requirements and improve the resource utilization. The performance of

these shops is strongly dependent on the batching policies employed for work in the shop. In particular, waiting time in queue and total manufacturing lead time for batches are functions of batch sizes. In turn, these affect work–in–process costs, safety stock requirements, schedule performance and part coordination for assembly [Karmarkar, Kekre and Kekre, 1985]. The batch–sizing models attempt to make a trade–off between the productivity losses caused by making too many small batches and the opportunity cost of tying up capital in inventory as large batches.

Another aspect of such manufacturing systems which is particularly difficult in practice is scheduling. Developing a production schedule involves selecting a process routing which will result in the completion of a job, designating the resources required to execute each operation in the routing, determining the run quantity (or lot or batch size) for each job, and assigning the time at which each operation in the routing will start and finish execution. The scheduling problem involves several decision functions of which the one concerned with due date is of special significance. A recent survey of U.S. companies using FMS indicates that meeting promised delivery dates, or due dates, is the most desirable objective management wants to achieve [Smith et al., 1986].

It is observed that most of the research on batch sizing has concentrated on a tradeoff between setup and inventory costs and has failed to analyze the interface between batch sizing and scheduling decisions. There is a significant relationship between batch sizing and scheduling decisions. Most of the studies available considered batch–sizing and scheduling together with the limitation of one operation per part. The relationships between batching and performance measures such as flow times, tardiness, waiting times and resource utilizations are not fully explored in the case of precedence constraints among operations of a part, alternate routings, and variable task processing times. Biggs [1979, 1985] investigates interactions between batch sizing algorithms and scheduling heuristics, and studies whether a batch sizing algorithm and a scheduling rule should be selected

independently. He finds a significant interaction between batch sizing and scheduling rules. He concludes that no single batch sizing rule, no single scheduling rule, and no combination of batch sizing and scheduling rules is universally superior for all the performance measures.

The intent of this chapter is to present comprehensively the batch sizing policies and their effect on scheduling when minimizing mean flow time, makespan, mean tardiness and mean waiting times and maximizing machine utilization. The other important issue covered in this research is the decision regarding dispatching rules when used in the presence of batch splitting policies. Scheduling rules are widely used in practice but there is only nominal information available about their individual performance against due date related criteria [Sabuncuoglu and Hommertzheim, 1993]. We use a batching policy that is quite different from the models used in queueing theory. It is simple and can be used readily for batching the orders for subsequent detailed scheduling. The batch sizing is performed based on the process plan of the jobs available for processing and detailed scheduling is performed using genetic algorithms.

6.2 Performance Measures and Dispatching Rules

An important decision that has to be made for batch sizing is the choice of objective function. The most common choices are minimizing, flow time, makespan, lateness, and tardiness [Dobson et al., 1987]. In this research, we consider five different performance measures, namely mean flow time, makespan, tardiness (to satisfy due date requirements), waiting time (to reduce the work–in–process inventory) and resource utilization (a naturally attractive aim of any company). In many manufacturing problems, the total flow time required to manufacture a product is an important consideration. Minimization of mean flow time is important because it also minimizes the work–in–process inventories [Dobson et al., 1987]. Long flow times introduce costs due to the high work–in–process inventories, larger safety stocks, and poorer performance for the due dates. Traditional batch sizing

models ignore flow time related costs, although there are systematic relationships between batch sizes and flow times. The objective of reducing the waiting time is important in discrete manufacturing systems because of the fact that the average workpiece in the average shop spends over 90% of its time in the shop waiting in queues. It also reduces the average work–in–process inventory in the system. An immediate result of queues is an increase in the manufacturing flow time for items due to waiting in queues.

Different dispatching rules used in this research to compare the batching policies with respect to performance measures are: a) SI (Shortest Imminent task processing time), b) LI (Longest Imminent task processing time), c) EDD (Earliest order Due Date), d) LST (Least Slack Time), e) LSO (Least Slack per Operation), f) SLACK/RP (Slack per Remaining Processing time), g) SLACK/RO (Slack per Remaining Operation), h) SRPT (Shortest Remaining Processing Time), and i) FCFS (First Come First Served).

6.3 Due Date Assignment Policies

A survey of due–date–based research reveals that due dates are usually treated as given information and taken as input to the scheduler problem. Different due date assignment policies observed to be used in the literature can be discussed under two main categories as follows [Cheng and Gupta, 1989]:

(i) Exogenous due date policies

  (a) CON (Constant): All jobs are given exactly the same flow allowance.

  (b) RAN (Random): The flow allowance for a job is randomly assigned.

(ii) Endogenous due date policies

  (a) TWK: Due dates are based on the total work content of a job.

  (b) SLK: Jobs are given flow allowances that reflect equal waiting times.

  (c) NOP: Due dates are assigned on the basis of the number of operations of a job.

The exogenous due date policies are representative of common practices where the

sales department quotes a uniform delivery date on all orders (CON) and where the customer establishes the due date (RAN). Most researchers agree that the performance of endogenous due date policies are superior to exogenous policies. In this research, TWK policy is used for assigning due dates as it is the most rational policy of due date assignment [Blackstone et al. 1982, Cheng and Gupta 1989, Sabuncuoglu and Hommertzheim 1993].

6.4 Batch Splitting Policies

In production systems, a batch of parts will have a number of required operations to be performed. There may be several feasible alternative sequences for carrying out these operations. Furthermore, several different workcenters may be capable of performing the same operations. A route through the shop consists of a feasible sequence of required operations and an assignment of each operation to a workcenter. In shops where considerable routing flexibility exists, batch routing may significantly affect production throughput and work–in–process inventory.

The approach used in this research for batch splitting is different from others in the sense that batches formed here are not based on the inventory models, but rather on the parts process plans. The demand, machine requirement and processing times are extracted from the process plans and are used as the basis for batch splitting. Different batch splitting policies that are used in this research include:

Policy 1: *No splitting*, where orders are scheduled without splitting them into batches.

Policy 2: *Splitting in equal sizes*, where all orders are divided such that each batch contains equal number of parts.

Policy 3: *Splitting in halves*, where all orders are divided into half the size of the original order.

Policy 4: *Splitting according to machine requirement*, where the batches are formed based on machine requirement of individual parts.

Policy 5: *Splitting according to total number of tasks*, where the batches are formed based on the number of tasks required for each part.

Policy 6: *Splitting according to the processing times*, where the batches are formed based on the individual parts total processing time.

All these policies are explained with the help of an example in section 6.6. Policies 1 (No splitting), 2 (Split equally), and 3 (Split in half) were first introduced by Lee [1987]. Karmarkar [1987] proposed to explore the possibility of equalizing the processing times across items for batch sizing. Policies 4 (Split according to machine requirements), 5 (Split according to total number of tasks), and 6 (Split according to processing times) were introduced recently by Jain and ElMaraghy [1994b]. The results indicate that in some cases Policy 5 (Split according to total number of tasks) outperformed other batch splitting policies.

6.5 Order Regeneration Methods

Once the job orders are grouped into batches, they are held in an order pool. They are released into the shop floor only when some predetermined conditions are met. The method employed for order release is called an order regeneration method. Three different methods are used in this paper: (a) static order regeneration, (b) dynamic constant order regeneration and (c) dynamic continuous order regeneration [Lee, 1987]. In static regeneration, jobs are released into the shop floor after all the job orders in the previous set are completed. In dynamic constant regeneration, a set of new job orders are released after every period (for example a shift), and in dynamic continuous regeneration, a new job order is released after completion of the previous order. For example, if there are three orders (Order 1, Order 2, and Order 3) present in the system and if each of them is batched into two orders i.e. 1a, 1b, 2a, 2b, 3a and 3b, then the three different regeneration methods can be described as shown in Figure 6.1.

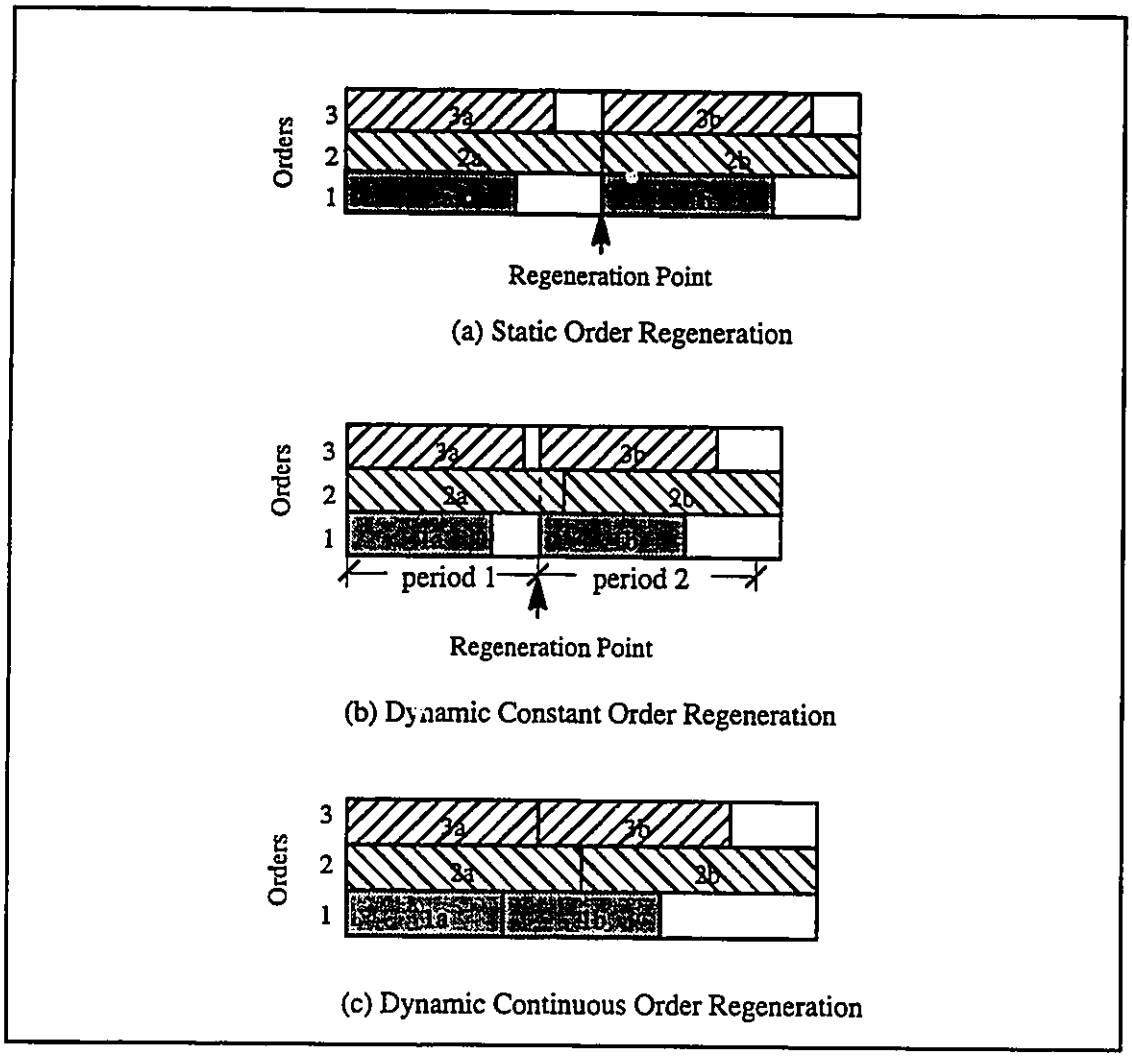


Figure 6.1 : Order regeneration methods

It has been observed that all three issues, i.e., splitting the batches, order regeneration methods and dispatching rules, play an important role in manufacturing scheduling and a decision is required for selecting the best of these parameters in order to increase the efficiency of the manufacturing system. The scheduling function is performed by genetic algorithms as they seem promising for the problems resulting in NP–completeness.

6.6 The Batch Scheduling Problem

The batch scheduling problem arises from the desire to accommodate the cyclical

production patterns that are based on economic manufacturing quantity calculations for individual items on a single production facility. When scheduling one item on one machine, the problem is easy and the solution is well known. However, when two or more items compete for the same facility, the phenomenon of 'interference' might occur, i.e., the facility will be required to produce two items at the same time, which is physically impossible. The problem then becomes how to economically schedule batches of one or more items on each manufacturing facility.

The detailed scheduling of the various elements of a production system is crucial to the efficiency and control of tasks. Orders have to be released into the system where each order may contain a set of jobs of a particular job type. Each order has a due date associated with it and in all cases it is the desire of the manufacturer to comply with these dates. Each job consists of a number of tasks to be performed in a given sequence on the available machines. The batch scheduling problem may be stated as: 'Given alternative process plans (routing flexibilities) for each part, the objective is to find a feasible schedule for a given set of orders such that some given performance criterion is optimized'. This research consists of two steps: (i) first, appropriate batches are formed based on the above mentioned batch splitting policies and (ii) second, detailed scheduling of these batches is performed using genetic algorithms. The input to the scheduler includes operational information such as job routing specifications, machine loading information, demand, task processing time, a job's priority as well as performance criteria to be optimized. This chapter addresses the need for selecting the best batching policy and priority rule, based on desired performance measures, in the presence of alternative routings.

6.6.1 The Problem Formulation

The prerequisite of any scheduling system is the process plan of each job which includes: the number and type of jobs, number of tasks in each job, number and types of

machines available, processing and setup times of tasks on machines, order due dates, release time of jobs into the shop floor, dispatching rules and performance criteria to be chosen. In this research, to demonstrate the effect of various batch sizing policies on scheduling, a four–workcenter production system which processes ten different jobs is modelled. This example is taken from chapter 5 (Problem B2). Details of this example can be found in Appendix B, where Table B.1 lists the machines available in each workcenter. The number of tasks in each job is known and varies from one to five. The processing order of tasks within a job is assumed to be strictly sequential. Table B.2 gives the demand and workcenter requirement of each task, where the task can be performed on any available machines within a workcenter (for example, if a task is assigned to the machining workcenter, then it can be performed on any machine, M1 through M5, probably with different processing times). Task processing times and setup times are also known and are given in Table B.3. Due dates are assigned based on the total work content of orders as this is the most rational method of due date assignment [Cheng and Gupta 1989]. Order due dates are set equivalent to (DDT x total processing time), where DDT is the due date tightness. The due date tightness factor (DDT) is set to 1.5 in the present example.

6.6.2 Batch Sizing Methodology (BSM)

The batch sizing policies mentioned earlier are discussed in detail in this section. The splitting procedure starts from defining the minimum number of parts that a sub–batch can contain. This is done in order to reduce the setup time required for each sub–batch. The more the number of sub–batches, the more the setups, and hence the more the setup time.

The following procedure is used when the batch sizing is according to policy 4 (number of machines), policy 5 (number of operations) and policy 6 (processing time). To calculate the number of the parts in each sub–batch, for $i = 0$ to $i = n$, do:

IF (D_i > Min_allowed_qty) THEN compute

(i.) P_i ($P_i = M_i$ for policy 4, $P_i = N_i$ for policy 5, and $P_i = T_i$ for policy 6)

(ii.) $TOTAL_i$, ($TOTAL_i = P_i * D_i$)

(iii.) MIN_i (MIN_i = minimum ($TOTAL_i$))

(iv.) $RATIO_i$, ($RATIO_i = TOTAL_i / MIN_i$)

(v.) Q_i ($Q_i = D_i / RATIO_i$)

ENDIF

Step (v) of this procedure gives the sub–batch size of an order. If Q_i is less than the minimum allowed quantity then this quantity, is added to the previous sub–batch if there is any. The above procedure is applied to the example problem and the sub–batches formed are shown in Figures 6.2 to 6.5. The minimum number of parts allowed in any sub–batch is four for the chosen example. It is observed from these figures that with policies 2 (Split equally), 3 (Split in half) and 4 (Splitting according to number of machines) in effect, the maximum number of sub–batches formed are two (a and b), in the case of policy 5 (Splitting according to the number of operations), the number of sub–batches are 4 (a, b, c and d) and it is 3 (a, b and c) when batch sizing is according to the processing times.

6.6.3 Scheduling Methodology

Once the batches and the quantities per batch are determined, they are sent to the order pool for subsequent scheduling purpose. The first set of orders are then released to the shop floor while the remaining sets wait in the order pool. These remaining sets are released for scheduling depending upon the order regeneration method used. The approach used in this research employs genetic algorithms for detailed scheduling. The search process begins with the proper representation of a schedule, generation of the schedule population and evaluation of each schedule in the population, and application of genetic operators to this population for improving the schedules. The search process is continued for a specified

Order No.	Policy 1	Policy 2	Policy 3
	Batch Size	Batch Size	Batch Size
	a	a, b	a, b
1	20	20	10, 10
2	20	20	10, 10
3	40	20, 20	20, 20
4	20	20	10, 10
5	40	20, 20	20, 20
6	40	20, 20	20, 20
7	20	20	10, 10
8	40	20, 20	20, 20
9	20	20	10, 10
10	40	20, 20	20, 20

Figure 6.2 : Batch splitting policy 1 (no splitting), 2 (splitting equally) and 3 (splitting in half)

Policy 4					
Order No.	Order Size	Machine Choice/part	Total	Ratio	Batch Size
			(1) x (2)	(3) / min	(1) / (4)
	(1)	(2)	(3)	(4)	(5)
					a, b
1	20	14	280	1.4	14, 6
2	20	12	240	1.2	16, 4
3	40	5	min ➧ 200	1.0	40
4	20	14	280	1.4	14, 6
5	40	6	240	1.2	32, 8
6	40	9	360	1.8	22, 18
7	20	13	260	1.3	15, 5
8	40	7	280	1.4	28, 12
9	20	13	260	1.3	15, 5
10	40	7	280	1.4	28, 12

Figure 6.3 : Batch splitting policy 4, splitting is according to the number of machines

Policy 5					
Order No.	Order Size	Operations per part	Total	Ratio	Batch Size
			(1) x (2)	(3) / min	(1) / (4)
	(1)	(2)	(3)	(4)	(5)
					a, b, c, d
1	20	5	100	2.5	8, 8, 4
2	20	4	80	2.0	10, 10
3	40	1	min ➧ 40	1.0	40
4	20	5	100	2.5	8, 8, 4
5	40	2	80	2.0	20, 20
6	40	4	160	4.0	10, 10, 10, 10
7	20	5	100	2.5	8, 8, 4
8	40	2	80	2.0	20, 20
9	20	4	80	2.0	10, 10
10	40	3	120	3.0	14, 14, 12

Figure 6.4 : Batch splitting policy 5, splitting is according to the number of operations

number of generations, yielding optimal or near optimal solutions. The genetic algorithms application procedure is the same as explained in chapter 5. The following genetic algorithms parameters were employed for detailed scheduling:

(i) Population Size (PSIZE): 80

(ii) Selection Bias (SBIAS): 1.9

(iii) Adaptive Mutation Rate (AMUT): 0.1

The genetic algorithm proceeds from generation to generation, saving the best schedules in each generation and getting rid of those schedules whose objective function values are low compared to the saved ones. In each generation the schedule population size remains fixed. Genetic algorithms use a limited search space, that is why they are popular for searching complex surfaces.

Policy 6					
Order No.	Order Size	Max Time per part	Total	Ratio	Batch Size
			(1) x (2)	(3) / min	(1) / (4)
	(1)	(2)	(3)	(4)	(5)
					a, b, c
1	20	155	3100	2.2	10, 10
2	20	126	2520	1.8	11, 9
3	40	54	2160	1.6	25, 15
4	20	159	3180	2.3	10, 10
5	40	95	3800	2.8	14, 14, 12
6	40	99	3960	2.9	14, 14, 12
7	20	190	3800	2.8	7, 7, 6
8	40	34	min ⧫ 1360	1.0	40
9	20	137	2740	2.0	10, 10
10	40	85	3400	2.5	16, 16, 8

Figure 6.5 : Batch splitting policy 6, splitting is according to the processing time

6.7 Results and Discussions

The proposed batch sizing and scheduling algorithms were programmed in 'C' for SUN computers. Table 6.1 to 6.6 summarizes the results of the experiments which compare the performance of nine scheduling rules with respect to five different performance measures. The selected performance measures provide a comprehensive view of the performance of the system under the different batching policies.

6.7.1 Batching–Policy Performance

In scheduling literature, it is often observed that while testing the scheduling rules, authors tend to use a single job in each order which is rarely the case in practice. The difference in the results, i.e., whether the performance of one rule is better or worse than

others, may be due to a special characteristic of the chosen example problem, that is, each order having more than one part, and the tardiness being assessed according to when the order (not the jobs within an order) is completed. The order size or batch sizes are important for deciding the number of optimal jobs per order where an order quantity is more than one. The performance of different batch–splitting policies are listed in Tables 6.1 to 6.6. Also, since the overall performance of the SI, LST, S/RP and FCFS dispatching rules were better than other dispatching rules, the graphs in Figures 6.6 to 6.9 compare different batching policies and regeneration methods with respect to these rules. It is observed from these figures that the performance of the system can be increased by splitting the order into small batches.

It was observed that when mean flow time is used as the criterion, policy 5 (split according to operations) was the best performer. Policies 2 (split equally) and 6 (split according to processing time) gave the next best results. This observation was true for all the dispatching rules in question.

In the case of minimizing the makespan, it was observed that in order to obtain the best results, no splitting policy should be used. From the figures it can easily be concluded that policies 5 (split according to number of operations) and 6 (split according to processing time) should never be used if makespan is used as the criterion no matter which dispatching rule is in effect. Policies 1 (No splitting), 2 (Split equally), 3 (Split in half) and 4 (Split according to the number of machines) showed comparable performance and should be used at the discretion of the user and order regeneration methods.

When mean tardiness is used as the criterion, policy 2 (splitting equally) and policy 5 (splitting according to number of operations) can be used interchangeably as they gave a comparable performance. The same trend was also observed for the other two performance measures. In the case of minimizing the mean waiting time and maximizing the machine

Table 6.1: Policy 1, no splitting

Performance Measures	Regeneration Methods	Dispatching Rules								
		SI	LI	EDD	LST	LSO	S/RP	S/RO	SRPT	FCFS
Mean Flow Time	None	4039	4336	4132	4134	4238	4274	4216	4188	4127
Makespan	None	5858	5810	6268	5964	5810	5866	5798	5690	5814
Mean Tardiness	None	89	418	155	109	181	119	138	151	225
Mean Waiting Time	None	1095	1246	1034	1089	1147	1178	1200	1091	1128
Average Machine Utilization	None	61.9	54.2	56.9	59.2	63.1	59.7	59.0	57.9	57.2

Table 6.2: Policy 2, split equally

Performance Measures	Regeneration Methods	Dispatching Rules								
		SI	LI	EDD	LST	LSO	S/RP	S/RO	SRPT	FCFS
Mean Flow Time	Static	3082	3404	3221	3237	3196	3252	3220	3221	3231
	Dy. Const.	2726	2955	2808	2800	2820	2795	2821	2768	2845
	Dy. Cont.	2599	2685	2610	2697	2661	2720	2673	2647	2638
Makespan	Static	7010	6754	6824	6610	6972	6802	6564	7144	6710
	Dy. Const.	5968	5576	5942	5804	5636	5716	5544	6094	6038
	Dy. Cont.	6112	5972	5893	6072	5810	5600	6028	5984	6028
Mean Tardiness	Static	467	566	482	447	368	466	454	514	488
	Dy. Const.	141	312	124	116	129	109	89	127	129
	Dy. Cont.	52	153	96	84	112	111	93	53	103
Mean Waiting Time	Static	341	438	366	405	398	397	376	334	396
	Dy. Const.	587	741	595	575	666	631	631	558	603
	Dy. Cont.	560	741	571	635	568	599	592	581	603
Average Machine Utilization	Static	52.5	49.8	52.8	50.0	51.9	52.5	53.5	50.0	51.6
	Dy. Const.	60.1	58.9	58.5	60.2	60.2	59.8	60.1	58.9	59.6
	Dy. Cont.	59.4	61.5	59.5	57.3	62.7	57.8	58.6	57.8	59.6

Table 6.3: Policy 3, split in half

Performance Measures	Regeneration Methods	Dispatching Rules								
		SI	LI	EDD	LST	LSO	S/RP	S/RO	SRPT	FCFS
Mean Flow Time	Static	3820	4189	3916	3823	3888	3979	3959	3888	3984
	Dy. Const.	3644	3858	3678	3714	3650	3683	3676	3591	3706
	Dy. Cont.	3592	3709	3589	3688	3509	3627	36.5	3589	3667
Makespan	Static	6158	6956	6598	6354	6466	6178	6190	6498	6572
	Dy. Const.	5966	6386	5942	6152	5850	6138	6000	6178	6010
	Dy. Cont.	5956	6506	6138	5966	6060	6028	6180	6160	6094
Mean Tardiness	Static	495	672	472	445	491	402	457	421	450
	Dy. Const.	241	423	171	256	284	277	235	219	291
	Dy. Cont.	155	296	121	162	129	179	226	119	144
Mean Waiting Time	Static	649	742	607	635	639	713	658	628	686
	Dy. Const.	758	904	723	769	768	815	750	709	788
	Dy. Cont.	601	758	681	668	710	714	701	659	673
Average Machine Utilization	Static	57.1	56.9	56.3	56.5	59.3	57.6	56.5	55.1	57.0
	Dy. Const.	62.6	58.3	60.9	59.7	61.2	60.6	59.2	57.4	60.7
	Dy. Cont.	62.9	55.0	57.7	58.2	59.0	58.8	59.0	59.5	60.8

Table 6.4: Policy 4, splitting according to number of machines

Performance Measures	Regeneration Methods	Dispatching Rules								
		SI	LI	EDD	LST	LSO	S/RP	S/RO	SRPT	FCFS
Mean Flow Time	Static	4320	4371	4308	4460	4374	4403	4323	4237	4360
	Dy. Const.	3695	3953	3664	3742	3720	3890	3850	3640	3878
	Dy. Cont.	3843	3993	3880	3775	3808	3982	3838	3816	3842
Makespan	Static	6534	6682	6634	6127	6545	6618	6404	6678	6602
	Dy. Const.	5796	5773	5888	5834	5691	5888	5675	5896	5764
	Dy. Cont.	5940	6204	6402	5962	5780	6077	5828	6242	6439
Mean Tardiness	Static	545	692	533	554	579	555	433	633	572
	Dy. Const.	124	290	108	127	139	147	107	144	208
	Dy. Cont.	217	250	108	215	207	292	208	207	238
Mean Waiting Time	Static	559	646	612	593	551	618	579	558	574
	Dy. Const.	908	1044	1048	967	899	1071	998	906	1095
	Dy. Cont.	670	754	583	723	662	699	737	650	717
Average Machine Utilization	Static	56.7	57.2	55.3	56.4	57.4	57.3	55.8	55.4	57.0
	Dy. Const.	63.8	61.9	62.6	61.8	64.4	63.5	64.2	62.6	63.1
	Dy. Cont.	61.6	57.8	56.7	60.1	61.3	58.1	60.5	57.9	59.8

Table 6.5: Policy 5, splitting according to number of operations

Performance Measures	Regeneration Methods	Dispatching Rules								
		SI	LI	EDD	LST	LSO	S/RP	S/RO	SRPT	FCFS
Mean Flow Time	Static	2618	2837	2684	2733	2654	2817	2699	2725	2758
	Dy. Const.	2611	2767	2632	2743	2516	2777	2565	2645	2784
	Dy. Cont.	2090	2231	2183	2229	2260	2335	2195	2085	2153
Makespan	Static	8258	8274	8053	8174	7946	7962	8070	7852	8084
	Dy. Const.	7123	7334	7114	7254	7413	7643	7312	7123	7565
	Dy. Cont.	6186	6294	6316	6234	6058	6320	6066	6022	6294
Mean Tardiness	Static	343	382	332	360	329	323	302	376	352
	Dy. Const.	232	269	217	206	223	219	232	235	274
	Dy. Cont.	96	149	53	62	83	81	78	73	150
Mean Waiting Time	Static	244	281	260	239	267	280	228	222	241
	Dy. Const.	625	667	671	645	652	690	645	656	706
	Dy. Cont.	568	773	657	662	647	681	675	573	611
Average Machine Utilization	Static	48.9	45.5	48.9	48.3	49.2	46.6	48.8	47.9	45.7
	Dy. Const.	54.3	53.9	54.9	52.7	54.3	57.7	55.4	54.8	56.6
	Dy. Cont.	63.9	63.3	60.1	61.0	62.9	61.5	63.7	62.5	63.2

Table 6.6: Policy 6, splitting according to processing time

Performance Measures	Regeneration Methods	Dispatching Rules								
		SI	LI	EDD	LST	LSO	S/RP	S/RO	SRPT	FCFS
Mean Flow Time	Static	3169	3238	3195	3211	3207	3162	3163	3325	3158
	Dy. Const.	2964	3028	3033	3013	2951	2984	2995	2974	2994
	Dy. Cont.	2776	2775	2713	2754	2818	2808	2827	2732	2750
Makespan	Static	6993	6425	7512	7265	7258	7165	7094	7339	7566
	Dy. Const.	6464	6572	6500	6470	6484	6506	6[illegible]8	6430	6476
	Dy. Cont.	5810	6260	6268	6161	5899	6155	6016	6155	6288
Mean Tardiness	Static	289	387	285	321	309	308	279	399	333
	Dy. Const.	113	213	127	115	132	110	133	147	183
	Dy. Cont.	107	207	75	98	79	75	86	129	175
Mean Waiting Time	Static	365	372	376	375	379	381	359	373	374
	Dy. Const.	426	460	443	426	463	415	423	470	426
	Dy. Cont.	707	723	666	726	758	791	749	703	698
Average Machine Utilization	Static	53.1	51.4	49.6	51.6	52.6	52.1	52.9	47.4	51.2
	Dy. Const.	58.7	57.3	58.3	58.6	59.1	58.9	58.6	57.4	57.7
	Dy. Cont.	63.3	61.7	65.2	62.7	61.4	63.0	63.1	58.8	59.1

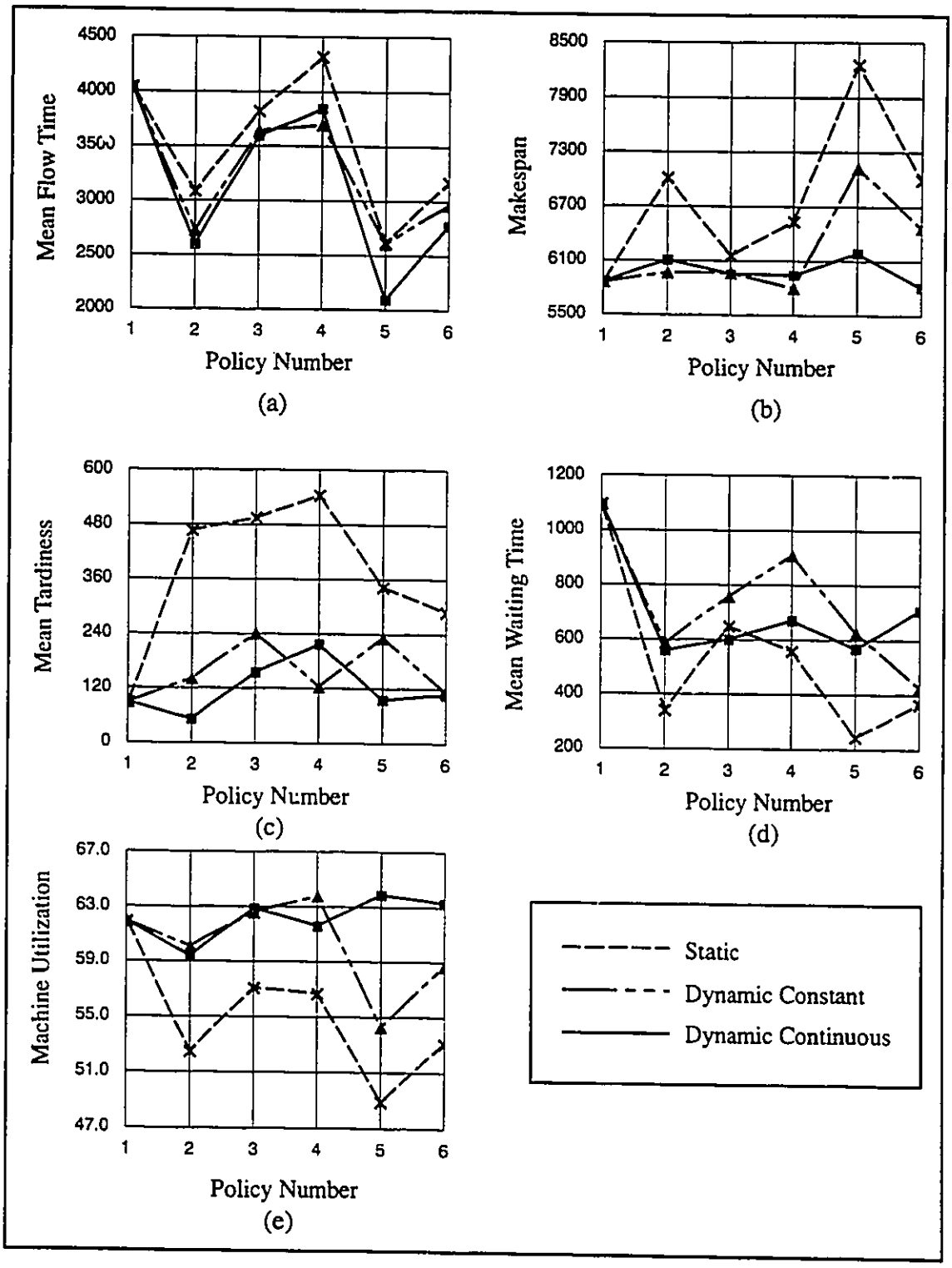


Figure 6.6 : Performance of batch splitting policies and order regeneration methods (SI Rule)

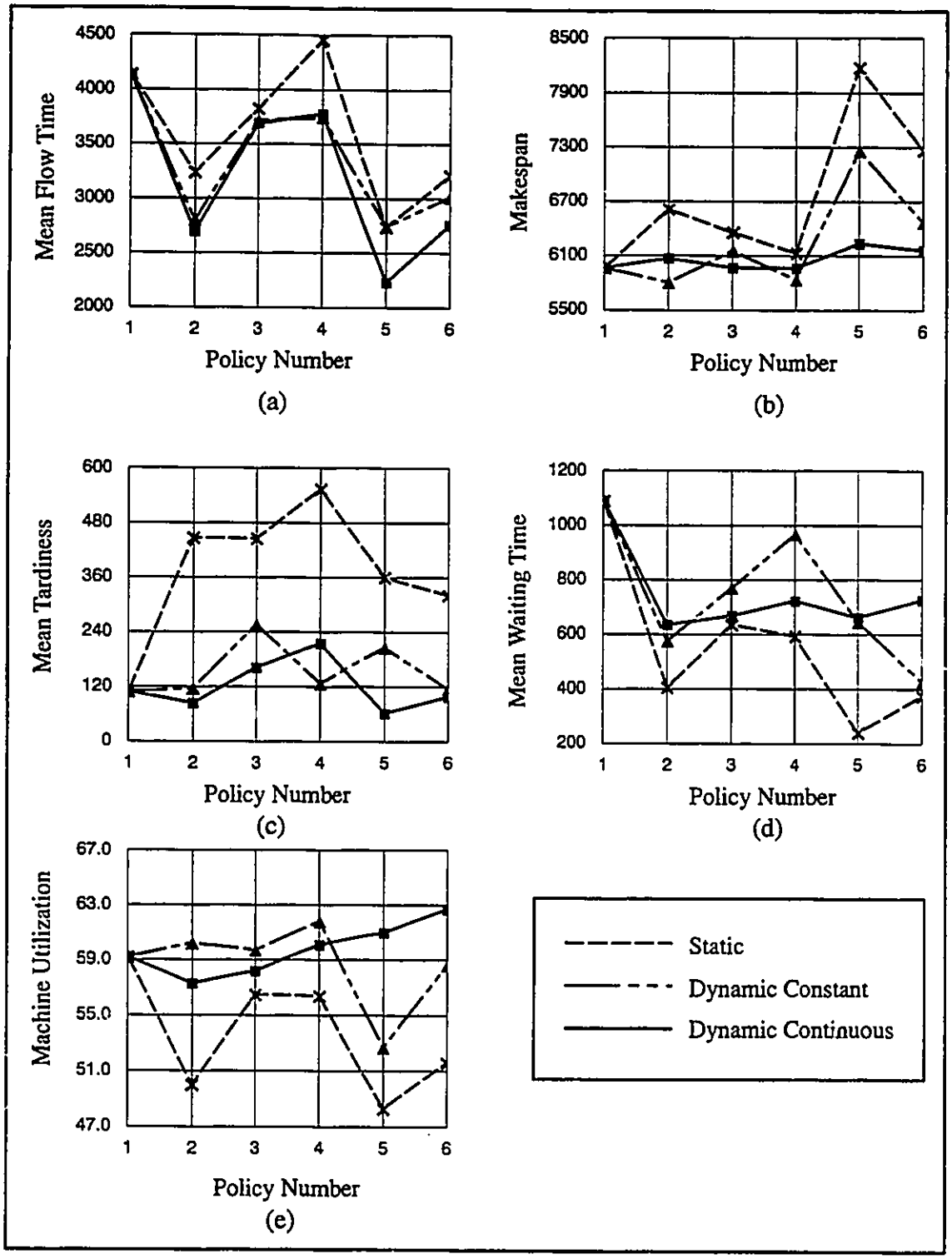


Figure 6.7: Performance of batch splitting policies and order regeneration methods (LST Rule)

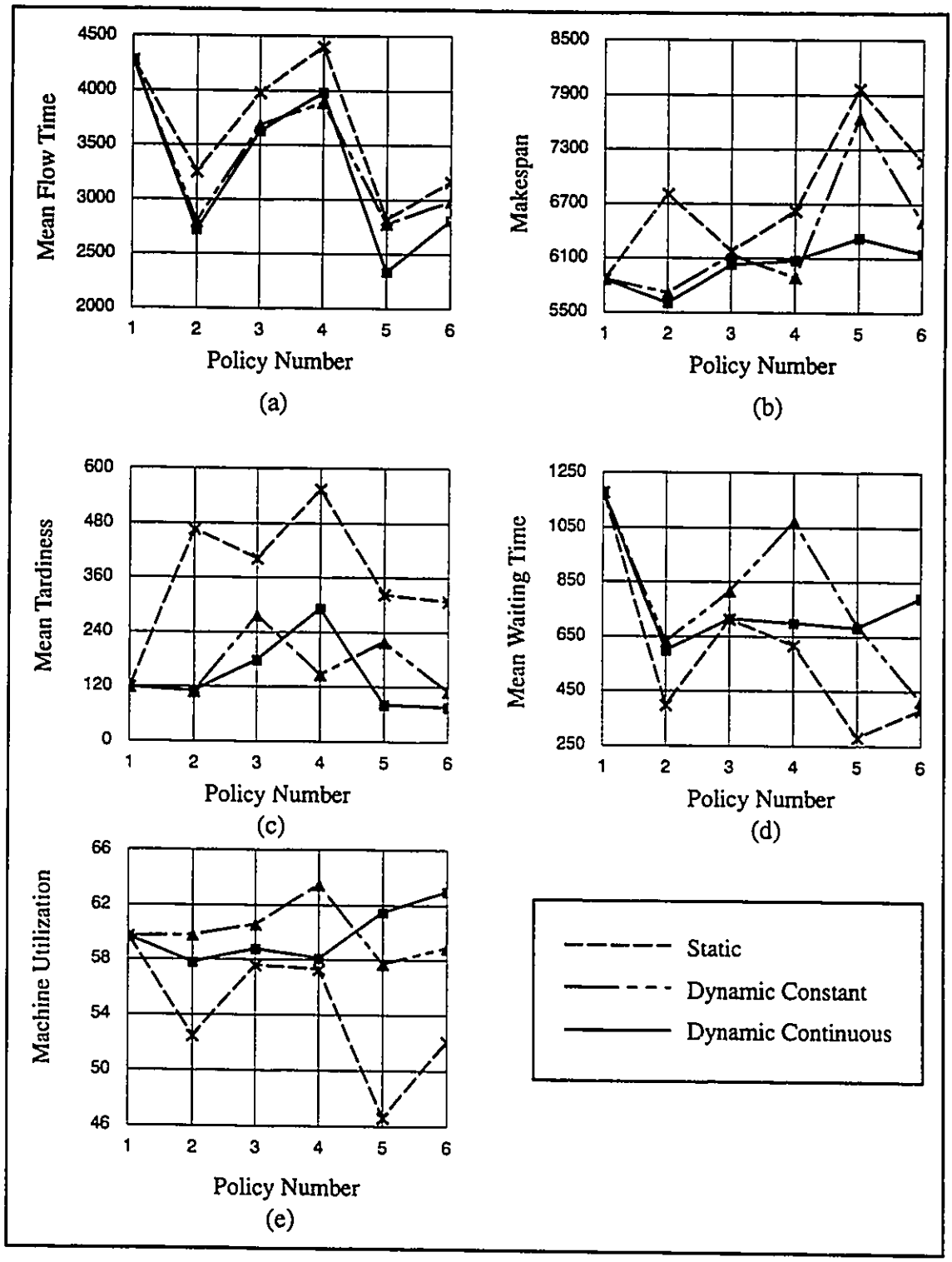


Figure 6.8: Performance of batch splitting policies and order regeneration methods (S/RP Rule)

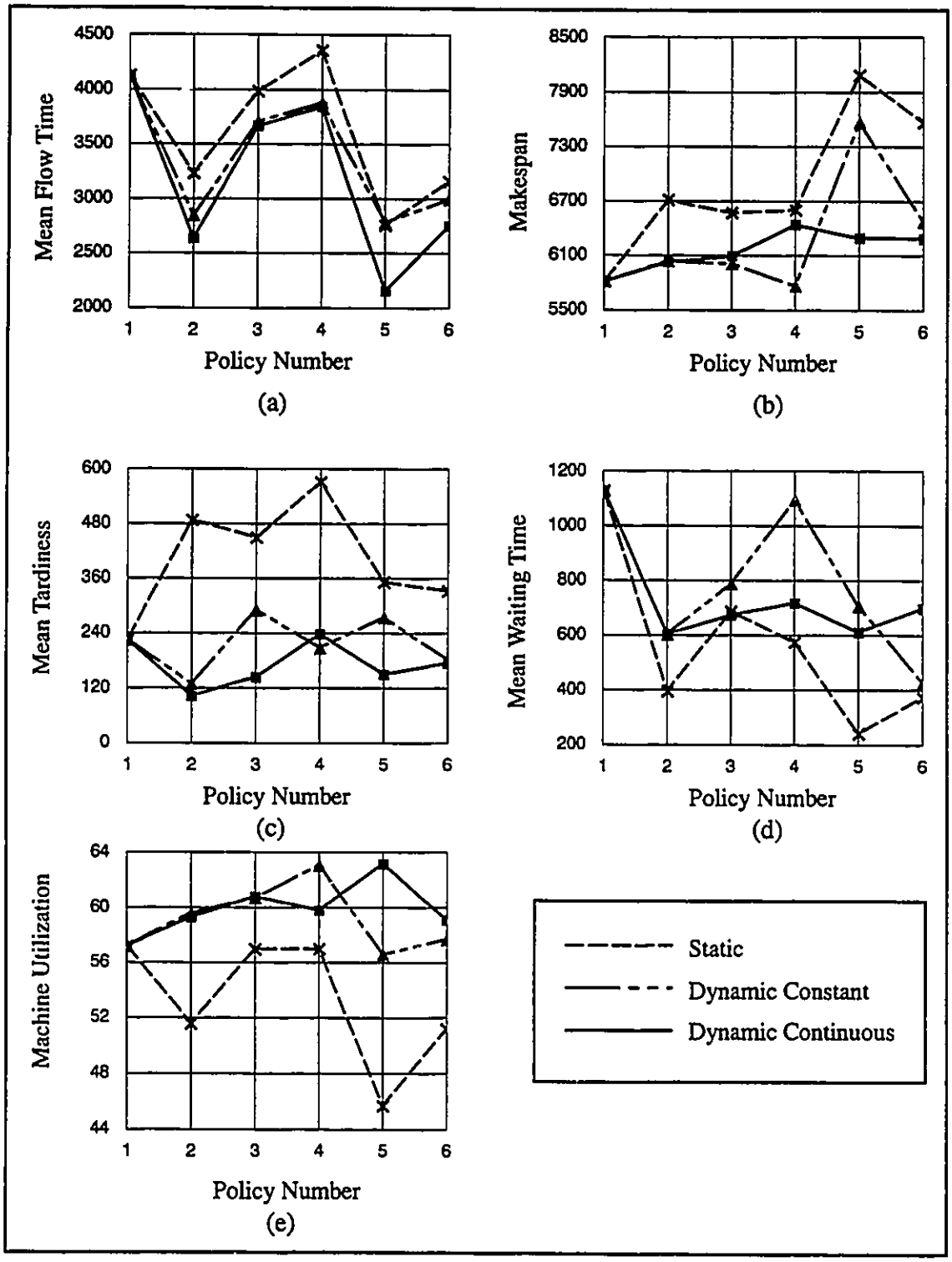


Figure 6.9: Performance of batch splitting policies and order regeneration methods (FCFS Rule)

utilizations, policies 2 (Split equally), 5 (Split according to the number of tasks) and 6 (Split according to the processing times) again performed better than the other ones.

Overall, it can be concluded that policy 5 (Splitting according to the number of tasks) can easily be used for splitting the batches except in the case of minimizing the makespan where no splitting at all is recommended.

6.7.2 Order Regeneration Method Performance

Once the batch splitting policy has been chosen, the decision regarding the job regeneration method should be taken in order to release the orders onto the shop floor. Three job regeneration methods mentioned earlier are used here to release the job orders. The period used in dynamic constant regeneration method was 2500 time units. The effect of different job regeneration methods is also shown in Figures 6.6 to 6.9.

It is observed from these figures that while minimizing the mean flow time, the dynamic continuous order regeneration method was the clear winner. The second best was the dynamic constant order regeneration method. In the case of minimizing the makespan, dynamic continuous regeneration method gave the best performance if policies 5 (Splitting according to the number of tasks) and 6 (Splitting according to the processing times) were in effect. For all other cases, performance of the dynamic continuous and dynamic constant regeneration methods were comparable. It is observed from these figures that when mean tardiness is used as a criterion, dynamic continuous regeneration performed well with all policies except policy 4 (splitting according to number of machines). For all the rules, dynamic constant regeneration method was the winner in the case when batch splitting was according to the number of machines. When mean waiting time is used as a performance criterion, the static regeneration method should be used for job regeneration. This corresponds to the results provided by Lee [1987]. The average machine utilization was high when the dynamic constant regeneration method was employed but only for batching

policies 1 to 4. The dynamic continuous regeneration method gave better average machine utilization, when policies 5 and 6 were used for batch sizing. These figures also show that the choice of dispatching rule does not affect the choice of batching policies or the regeneration methods in any way.

6.8 Summary and Conclusions

In this chapter, an approach based on the part process plans has been used for splitting the batches. This approach is different from queuing models that are usually employed for batch sizing. Assignment of these batches to the manufacturing resources is achieved through genetic algorithms. The research was motivated by the common occurrence of such problems in many real situations, manufacturing facilities in particular. The formulations studied here are for multi–product, multi–machine manufacturing systems which include several complicated factors such as batch sizing, precedence constraints among operations, alternate routings and variable processing times. Numerical experiments indicate that the policies provided for batch splitting are quite effective and can be used to represent the real situations of automated manufacturing systems.

CHAPTER 7

DYNAMIC SCHEDULING IN MANUFACTURING

Scheduling of production in discrete manufacturing systems has been extensively researched over the past years and it continues to attract the interest of both academic researchers and practitioners. The generation of new and modified production schedules is becoming a necessity in today's complex manufacturing environment. Once an initial schedule has been implemented, it is almost immediately subjected to new production conditions, demands and constraints. Uncertainties in the production environment and modelling limitations inevitably result in deviations from the generated schedules. This makes rescheduling or reactive scheduling essential. Four different types of uncertainties that normally cause discrepancies between the actual output and the planned output are considered in this paper. These include unforeseen machine breakdowns, increased order priority, rush orders arrival and order cancellations. The proposed algorithms revise only those operations that need to be rescheduled. They can be used in conjunction with the existing scheduling methods to improve the efficiency of discrete manufacturing systems.

7.1 Introduction

At present, most industries are confronted with perpetual customer demands for a wider variety of products, faster production rates, shorter delivery times and higher delivery reliability. The flexibility to manufacture a wide range of products in a short time has been achieved at the expense of manufacturing efficiency. The performance of a production

system depends greatly on good and proper rescheduling with the uncertainties present in the production environment. These uncertainties often result in orders following a route through the shop floor different from what was originally developed. In such cases, previously generated schedules become invalid and have to be regenerated.

Several design and control factors can cause interruptions in a manufacturing system. In this research, we limit our discussions to disruptions caused by unforeseen machine breakdowns, increased order priorities, rush order arrival and order cancellations. The proposed rescheduling algorithms can be implemented in existing scheduling systems to improve effectiveness. Efforts are focussed on necessary local rescheduling only to maintain the stability of existing schedules and provide quick solutions. The main emphasis of rescheduling is to find immediate solutions to problems resulting from disturbances in the production system. This may result in a less effective schedule; compared with total rescheduling, however, it meets the new constraints with the least disruption of production. Rescheduling commences from the time a disturbance occurs and takes into account the current state of the shop floor. It is an iterative process of two steps [Li et al., 1993]: (i) Reschedule and evaluate the existing schedule depending on the change of the conditions, demands or constraints. If the result of this revised schedule is acceptable, then stop, otherwise: (ii) Determine an improved solution by performing iterations. An acceptable revised schedule is one that overcomes the new constraints, for example, a schedule that allows re-routing parts from a machine that has just broken down. In reality, the definition of an acceptable revision is dependent on the prevailing requirements.

In this chapter, the proposed rescheduling algorithms assume that some of the jobs in the system have an alternate process plan, a common situation in modern discrete manufacturing systems. In the case of unforeseen disturbances such as breakdowns, shortage of resources (machines, tools, etc.), bottlenecks etc., alternate process plans could be used in order to increase throughput, reduce waste and improve resource utilization and

productivity. However, having an alternative process plan is not always the best solution in the case of interruptions. Sometimes, process plans themselves have to be modified in order to deal with the shop floor disturbances. In reality, 20–30% of the process plans and routing sheets are modified locally to cope with production bottlenecks, equipment failures, resources shortages and changes in order priorities [H. ElMaraghy 1993, H. ElMaraghy and W. ElMaraghy 1993]. They proposed some issues related to bridging the gap between computer aided process planning (CAPP) and production planning and control (PPC). They developed an 'integrator' module to integrate CAPP and PPC functions. A reactive planning environment (RPE) [Stranc, 1992] was developed to capture plans and resource alternatives and provide an effective means of evaluation and selection of plans based on the dynamically changing shop floor environments. RPE uses a feature–based, object oriented approach to represent a product structure hierarchically. The output of RPE is all the input required by the production planning and control system such as precedence constraints, resources required, alternate resources, and alternate routes. It also allows for alternate plans evaluation according to user defined criteria such as time, scrap rate, load balancing and cost, and selects the best plan under given conditions such as absence or over–utilization of certain resources. The precedence graph, obtained as output from RPE, is converted to the usual sequential process plan format in a flat file for use by traditional production planning and control system.

However, in this research, the rescheduling algorithms are developed based on the assumption that process plans are already available and cannot be changed once they have been incorporated into the PPC or scheduling system. The focus is entirely on the production planning and control (PPC) or scheduling stage where uncertainties on the shop floor are dealt locally. There are a number of techniques used for generating an initial schedule including mathematical programming, simulation, artificial intelligence, and genetic algorithms. These techniques were discussed in chapter 2. The performance of the

proposed rescheduling algorithms is not affected in any case by the approach that has been used to obtain the initial schedule, and therefore, these algorithms can be implemented on any existing computer–based scheduling systems.

7.2 Rescheduling in Manufacturing

Rescheduling implies that a given schedule is to be revised at specific points in time due to certain significant changes in operating conditions. The rescheduling procedure is never planned in advance but is brought into action due to certain unavoidable circumstances. Because of the simplicity of the proposed algorithms, it is expected that they can be used for the real–time control of any advanced manufacturing systems, as opposed to rescheduling in production and inventory control, where scheduling is considered over a period of weeks or days.

Several types of random variables (uncertainties) that affect the actual shop output should be taken into account if scheduling is to be realistic. A production schedule typically specifies tasks of a particular job, machines on which these tasks can be performed and their starting and ending times. In real–life situations, many dynamic events occur on the shop floor that require adjustments to existing schedules. Rescheduling is a necessary and important part of production planning because of the dynamic environment to which manufacturing systems are often subjected. This research considers the following four different types of uncertainties:

(a) machine breakdown,

(b) the arrival of rush orders,

(c) increased order priority (i.e. the change in due dates), and

(d) order cancellations.

In each of the above cases, tasks are performed in accordance with the predetermined task sequence, even after the disturbance occur. If a task is unable to continue as scheduled,

then based on the priority of the task, an attempt is first made to find an alternative free machine, failing which pre–emption is attempted. The system state such as machines and tasks status and ready and completion time for each task are updated whenever either a task is ready to start or is completed, or when an interruption occurs.

When a machine breakdown occurs, the remaining operations of the job may have to be performed using other machines. This affects the scheduling decisions previously made. The rescheduling becomes even more difficult if certain tasks can be completed only on one machine. At the time of interruption, if a task is being performed on the broken machine, the system is checked for availability of alternate machines. If an alternate machine is free, the pre–empted task is assigned to it. If the alternate machine is performing another task, the priority of the two tasks are compared, and the task with higher priority is assigned to that machine. Also, task setup time on a machine plays an important role in rescheduling. In the case of a breakdown, a comparison is made between the task setup time of the pre–empted task on an alternate machine and the down time of the failed machine. If the task setup time is lower, only then is the task switched to the alternate machine, otherwise it waits until the broken machine becomes available for production.

When a rush order arrives, the tasks of the rush order are immediately assigned to the required machines based on its priority. If the new order is not important, then priority is assigned based on standard priority rules, for example FCFS or due date based, otherwise highest priority is assigned to it. This may cause pre–empting some of the tasks already assigned to the machines by the new rush order. All the pre–empted tasks are subsequently moved forward in time in order to accommodate the rush order. It is also possible that pre–empted tasks have a choice of alternate machines, but that situation is not taken into consideration. Local rescheduling is attempted where alternate machines are considered only in the case of a machine breakdown, and attention is paid only to the tasks that are

affected by the disturbance. When a job priority is increased dynamically, rescheduling is accomplished by bringing its remaining tasks forward in time on their respective machines. In the case of an order cancellation, all the remaining tasks of that order are deleted from the task list and at the same time the system updates the time and machine status of the remaining jobs.

It is assumed that an initial schedule, and thus the machines' utilization, is available a priori. In this research, an initial schedule is obtained by using GAs for rescheduling consideration. An example problem with the machine and task parameters shown in Figure 7.1 was generated for the description of the proposed rescheduling algorithms. A

Job (i)	Task (O_{ij})	Machines					Priority
		M1	M2	M3	M4	M5	
1	11	10/20	–	12/21	–	–	1
	12	–	–	–	8/22	12/20	
	13	5/15	–	6/12	–	–	
2	21	8/14	–	–	–	–	2
	22	–	8/22	10/18	–	–	
	23	7/29	–	10/30	–	–	
3	31	12/38	12/40	–	–	14/36	3
	32	–	–	–	9/21	7/23	
	33	23/41	–	–	–	21/39	
4	41	20/53	–	18/52	22/50	–	4
	42	–	9/33	–	12/28	–	
	43	6/22	8/23	–	–	–	

Figure 7.1: Setup/Processing time requirement of tasks

manufacturing system with five machines, which are capable of processing four different jobs, is considered. Each job has three tasks that should be performed in a strict sequential manner. Priorities among the jobs are assigned at random. The search process begins with the proper representation of a schedule, generation of schedule population and evaluation

of each schedule in the population, and application of genetic operators to this population for improving the schedules. The search process continues for a specified number of generations yielding an optimal, or a near optimal, solution. The Gantt chart of the schedule obtained by using genetic algorithms is shown in Figure 7.2.

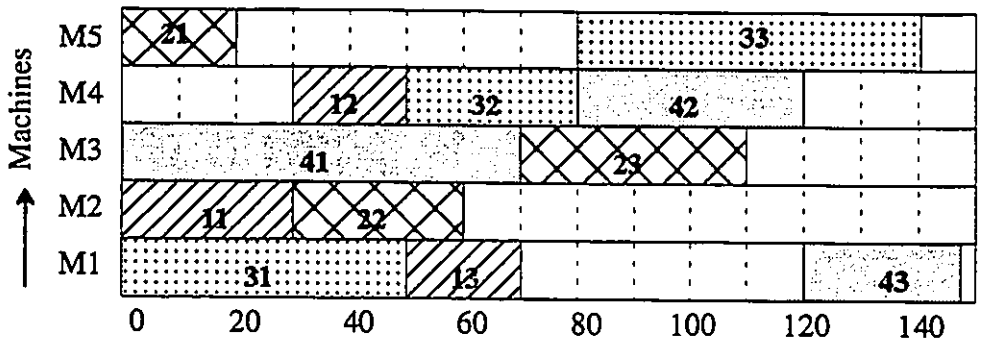


Figure 7.2: Initial schedule for the chosen example

Utilization of machines is an important issue to be considered while rescheduling manufacturing systems. For example, in case of a breakdown, if the affected task has a choice of more than one machine, it is re-routed to the least utilized machine. Machine and task status are updated continuously while running the schedule. Yamamoto and Nof [1985] identified two phases before rescheduling is invoked. These are: (i) the planning phase, where an initial schedule is to be generated, and (ii) the control phase, where schedule progress is proposed and abnormal status are identified. Once an abnormality is identified by the control phase, a reschedule procedure can be invoked. Based on the type of abnormality, certain decisions are made before the rescheduling algorithm is applied to the schedule. For example, in case of machine breakdowns, the expected duration of breakdown has to be considered. In the case of new orders, it must be determined whether it is a normal order or a rush order. If it is a normal order, then it is merged into an existing schedule, and all tasks of the new order are given the same treatment as other tasks. If it is a rush order,

the highest priority is assigned to it and machine assignment is made accordingly. The framework of the control phase employed in this research is depicted in Figure 7.3.

7.3 Rescheduling Algorithms

As discussed earlier, the control phase of a manufacturing system includes monitoring and adjusting the system's schedule to ensure smooth order progress on the shop floor. The basic research issue addressed in this chapter is that of determining how real–time control in discrete manufacturing systems can be achieved in the presence of uncertainties. After an initial schedule is loaded, the basic execution loop is as follows:

While All tasks are not complete Do;

STEP 1. Monitor Schedule Progress

STEP 2. Examine Disturbances into the system

STEP 3. Classify the Control problems (such as breakdown, rush order etc.)

STEP 4. Select alternative actions if available

STEP 5. Update Status of Machines and Uncompleted Tasks

STEP 6. Reschedule remaining Tasks

End;

Figures 7.4, 7.5, 7.6 and 7.7 show the response of the rescheduling algorithm after corrective action is taken. The algorithms for different types of disturbances are explained with examples in the following section.

7.3.1 Machine breakdowns

The following loop is executed after the initial schedule is loaded, for time T = 0, 1, 2, 3,... until the schedule is complete.

At any time t, IF there is a machine breakdown then

Find the broken machine and interrupted task

Assign expected downtime randomly for this machine

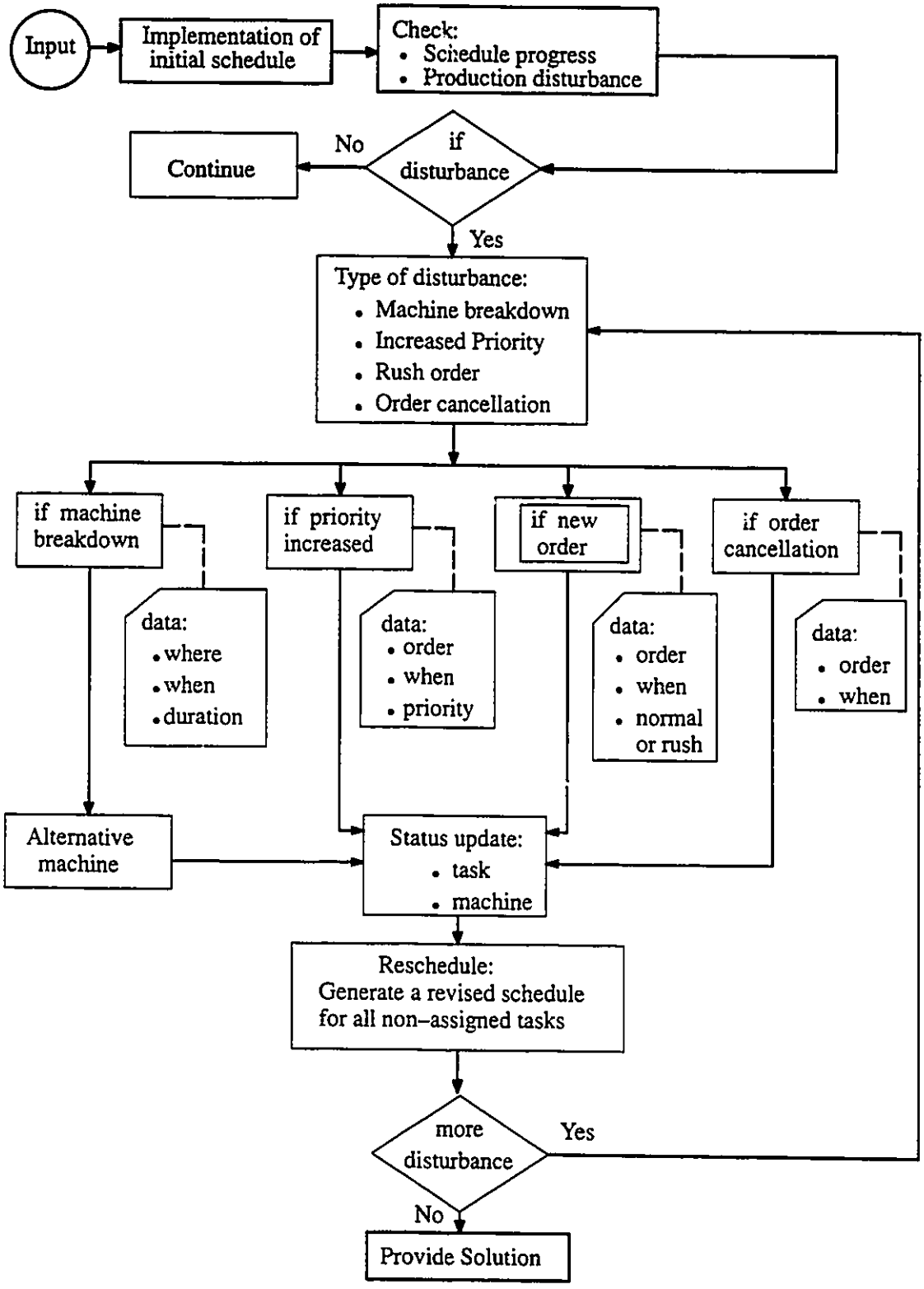


Figure 7.3: Control framework

IF there is any operation currently on this machine, then

Split the task and revise the task status (time remaining)

IF there are alternative machines available, find setup and processing time required for all the alternative machines. If more than one choice is available, choose the least utilized machine

ELSE broken task will start on the same machine whenever it becomes operational

ENDIF

IF there is a choice of alternate machine

IF setup time on alternate machine is less than the downtime of the broken machine

IF alternate machine is free, assign broken task to alternate machine and update the system status

ELSE (if alternate machine is not free)

IF broken task priority is higher than the task currently performed on alternative machine, then pre–empt the alternate machine and update the system status

ELSE (if priority is lower)

IF the difference between the ready time of the broken machine and release time of an alternate machine is more than the setup time on alternate machine, then assign broken task to alternate machine when it becomes free

ELSE broken task will start on the same machine whenever it becomes ready

ENDIF

ENDIF

ENDIF

ELSE broken task will start on the same machine whenever it becomes operational

ENDIF

ELSE broken task will start on the same machine whenever it becomes operational

ENDIF

ELSE update the machine status list

ENDIF

This algorithm was applied to the example problem and results are shown in Figures 7.4a through 7.4e. Numbers inside the blocks are the tasks associated with the jobs, and black rectangles represent the machine failure periods. Figure 7.4a shows the situation when machine 5 fails at time T = 15 and there is no other machine that can perform the broken task 21. This interrupted task started on the same machine at time T = 36. No setup is necessary, since the interrupted task starts on the same machine. Because the completion time of task

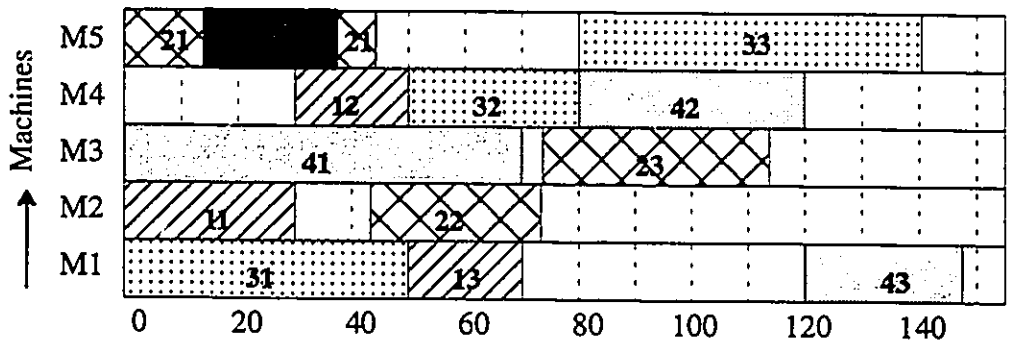


Figure 7.4a: Machine failure, no alternate machine

21 is changed, successive tasks of job 2 i.e. 22 and 23 are moved forward in time in order to satisfy the precedence constraints. Because of the machine breakdown the job completion time is also changed. Figure 7.4b shows the case of machine 1 breakdown. First the algorithm looks for the alternative machine for the broken task (31), and once it is found, the downtime of the machine is compared with the alternate setup time. Since an alternate

machine (machine 5) is free and the breakdown time (40 time units) is longer than the task setup time on alternate machine, the broken task is split and immediately starts on machine 5. This changes the start time of tasks 32 and 33. Figure 7.4c shows when the downtime

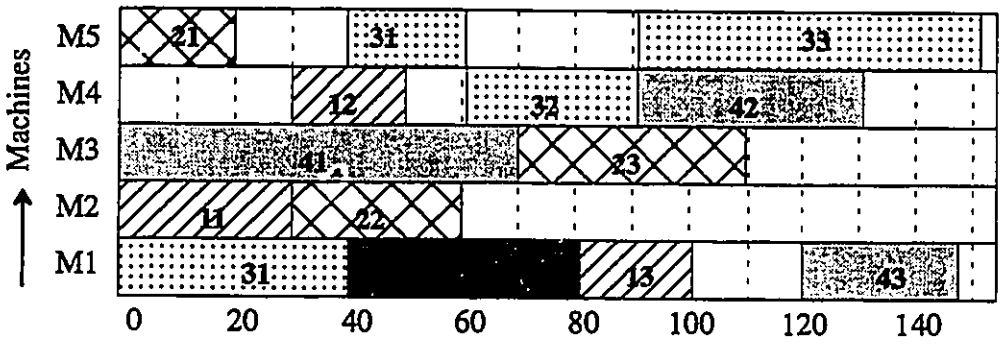


Figure 7.4b: Free alternate machine, switchover succeeded

of the broken machine (M1) is less than the setup time on the alternate machine. Hence switching over does not take place and task 31 starts on the same machine when it is ready for processing. Figure 7.4d shows the case where the priority of tasks come into action.

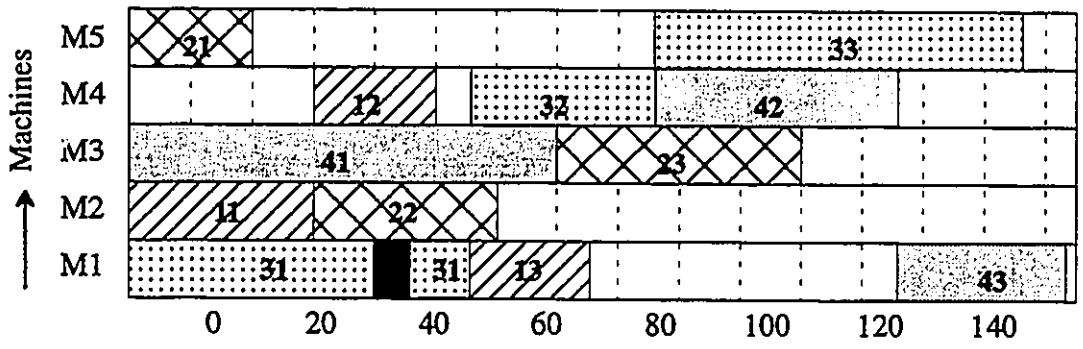


Figure 7.4c: Free alternate machine, switchover not succeeded

Machine 2 fails at time T = 52 for 20 time units. The broken operation has an alternative option of machine 3 on which the setup time (10 time units) is lower than the downtime of the broken machine (20 time units), but the switchover does not occur because the priority of task 22 is lower than that of task 41 which is currently being processed on machine 3.

Therefore, task 22 waits for the downtime duration and starts on the same machine. This moves task 23 forward as shown in the figure. The pre–emption of a machine takes place

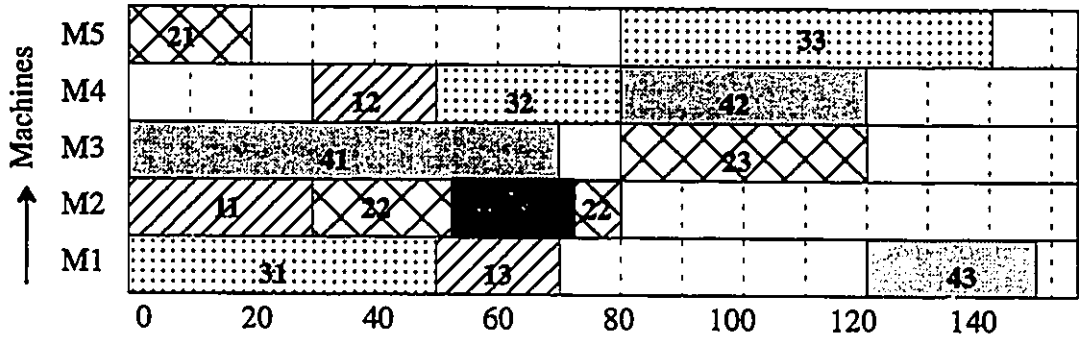


Figure 7.4d: Alternate machine occupied, pre–emption fails, broken operation starts on the same machine

in Figure 7.4e where machine 1 failure causes task 13 to start on alternate machine 3, because of the high priority of task 13. This process splits the low priority task 41, which starts on the same machine after the task 13 is completed. Since task 41 starts again, the setup for task 41 on machine 3 is again required. Because of this, tasks 42 and 43 are moved forward in

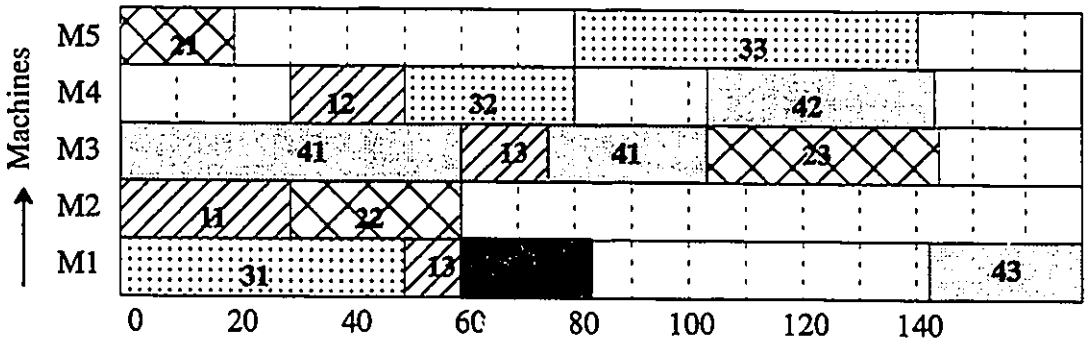


Figure 7.4e: Alternate machine occupied, pre–emption succeeds

time. In Figure 7.4f, the failure of machine 2 causes task 22 to be split and sent to the alternate machine 3. The expected duration of the machine failure is 50 time units. Since the difference between the ready time of the broken machine (T = 100) and release time of an alternate machine (T = 50) is less than the setup time of the broken task 22 on the alternate machine (M3), and because task 22 has lower priority than task 41, task 22 waits for machine

3 to become free. This is the best that can be done to finish job 2 as soon as possible. Task 23 is also shifted, because of the precedence constraint which is already known to the scheduler. Task 23 cannot be started until task 22 is complete.

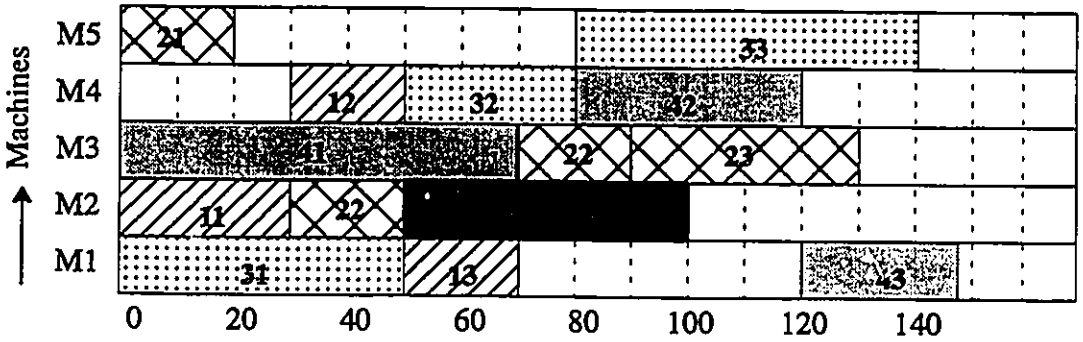


Figure 7.4f: Alternate machine occupied, preemption fails, broken operation starts on alternate machine

7.3.2 Increased Priority

Whenever the priority of any order is increased, all the succeeding tasks of this order are advanced in time. The next task of this order will start as soon as the preceding task is completed. The tasks of this high priority order are marked 'urgent' so that the ready time of the remaining tasks (of this order) becomes fixed in time. While scheduling the tasks on the machines, a search is made through the task list for any urgent task. If an urgent task is found, it is first assigned to the machine while other tasks requiring the same machine wait in queue. This is particularly useful when assigning tasks to machines on the basis of dispatching rules. The assignment of the urgent tasks does not follow the dispatching rule, while normal tasks do. Therefore, when there is a choice between an urgent task and a dispatching rule task (say SPT), the urgent task will be selected for assignment. The increase in priority is handled in the following manner:

Run schedule obtained from stage 1 starting at time T = 0, 1, 2, ... until the schedule is complete.

At any time t, if there is an increase in priority then

Find the order whose priority is increased

Assign the highest priority to this order and revise the task status

Assign the same priority to all the tasks belonging to this order

IF the increased priority task is currently not loaded on any machine

IF machine required by the increased priority task is free, assign task to the machine

ELSE pre–empt the machine and start the high priority task immediately and update the system status

ENDIF

ELSE advance all the remaining tasks to start immediately

ENDIF

This is shown in Figure 7.5. The priority of job 2 is increased at time T = 50 which

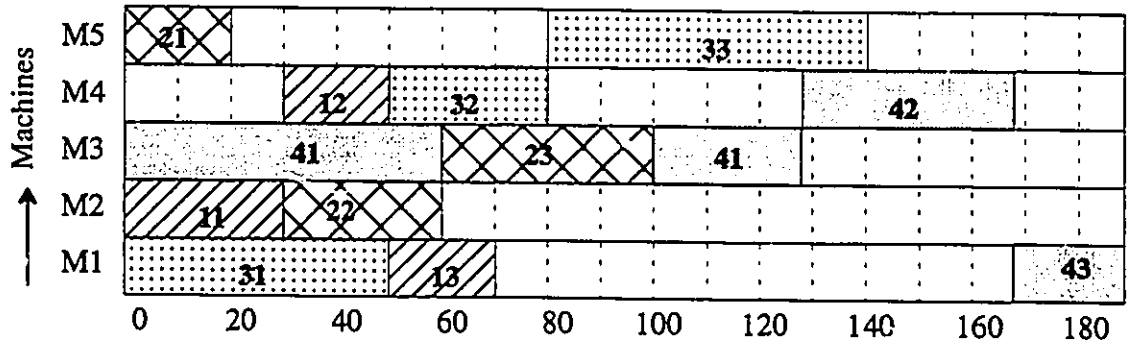


Figure 7.5: Job 2 priority increased at time T = 50

causes task 41 to split and wait at time T = 60 until task 23 is completed. The result is that job 2 is completed earlier than its originally scheduled time. Additional setup of machine 3 is required at time T = 100 for the task 41 because of the restart of the task. This moves tasks 42 and 43 forward in time in order to satisfy the precedence constraints.

7.3.3 Rush Order Arrival

When a new order arrives into the system, it is first determined whether it is a rush or a normal order. If it is a normal order, due dates are assigned and it is merged into the current schedule. If it is a rush order, then the highest priority is assigned to it and it is treated similar to an increased priority order. All the machines required by the rush order are released whenever they are required. The algorithm for rush order arrival is almost the same as the algorithm for increased order priority except that there is an increase in the number of orders flowing through the system. This is shown in Figure 7.6.

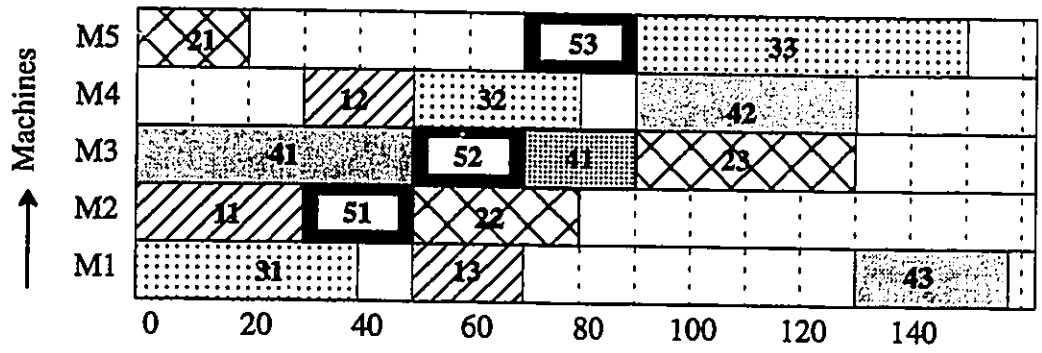


Figure 7.6: Rush order 5 arrived at time 30 with priority 5

7.3.4 Order (Job) Cancellation

Figure 7.7 shows the effect of order cancellation on the schedule. At time T = 40, order 4 has been cancelled, which advances task 23 forward. The result is the earlier completion of job 2. Also, the makespan of the schedule decreases because of the order cancellation.

7.4 Results and Discussions

Dynamic reaction to developments on the shop floor is essential for realizing a truly flexible control of the manufacturing system. In order for a controlling mechanism to perform in a dynamic production environment, it must consider scheduling or dispatching

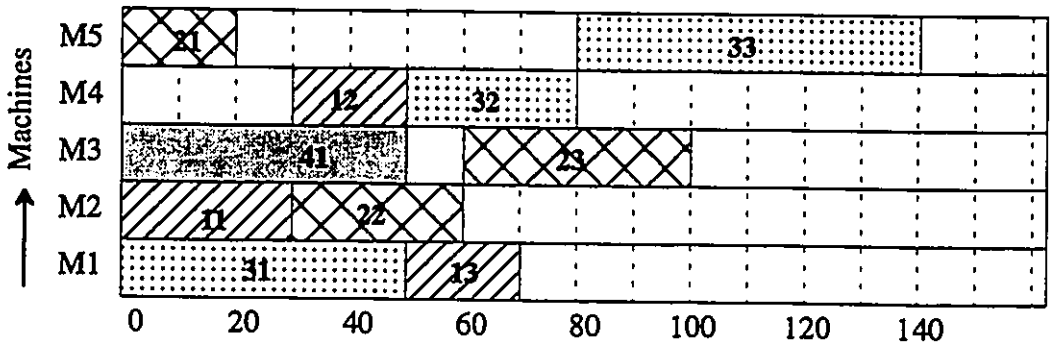


Figure 7.7: Order 4 cancelled at time T = 50

rules, as well as the system performance. We examined three performance measures, mean flow time, mean tardiness and average machine utilization with respect to three dispatching rules, namely the shortest processing time, earliest due date, and first–come–first–served rule and studied their degradation with respect to machine failure rates. Since the previous example was a small one for the result verification, an example with four workcenters and ten jobs from chapter 6 was used for testing purposes. The number of machines at each workcenter varied between 1 and 5. All machines at a particular workcenter were capable of performing the same task, i.e., if two machines M1 and M2 were available in workcenter WC1 then a task processed on M1 could also be processed on M2, perhaps with a different setup and processing time. The number of tasks in each job, sequence of tasks within a job, machine requirement for each task and processing and setup times were known in advance and are given in Appendix B. Order priorities are strictly based on the due dates of the individual orders. For this example, a planning horizon of 10,000 units is considered. An assumption made about the machine failure is that a failure occurs according to some predefined failure probability and not according to a machine failure rate. At any discrete point in time, any machine could fail randomly and independently with the same probability of failure. This failure probability is dependent on the random number cutoff value. At the

beginning of the scheduling period, 10,000 numbers are randomly generated (between 0 and 1) to represent discrete points of time in planning horizon, and cut off values are used for fixing the machine failure probability. For example, if a cut off value is 0.0004, then at each discrete point of time in planning horizon, value of corresponding random number is compared with the cutoff value. A value less than the cutoff value will result in a machine failure. For our example, a cutoff value of 0.0004 resulted in a failure probability of 0.06% or a failure frequency of 6 i.e., machines will fail 6 times during the planning horizon. Duration of failure was machine–dependent and randomly generated. It is assumed that the machine repair time is the same as the machine failure duration. The repair takes place as soon as the machine goes down. Table 7.1 shows the random number cutoff value, probability of machine failures and frequency of machine failures. Table 7.2 shows the

Table 7.1: Cutoff value, Failure probability and Frequency

Cutoff value	0.0000	0.0004	0.0008	0.0012	0.0016	0.0020
Failure probability	0.00	0.06	0.09	0.12	0.18	0.21
Frequency	0	6	9	12	18	21

effect of machines breakdown on the mean flowtime, mean tardiness and average machine utilization with respect to dispatching rules. The mean completion time is computed using individual completion times. Machine utilization is equal to the time a machine was occupied divided by the maximum flow time. For mean tardiness, due dates for each order are assigned, based on the total work content of orders and are set equivalent to (DDT * total processing time), where DDT is the due date tightness. In this experiment a setup due date tightness factor (DDT) was set to 1.5. These performance criteria have been charted in Figures 7.8, 7.9, and 7.10. Because alternate routings are available in the system, the graph

Table 7.2: Performance measures versus failure probability

Performance Measure	Dispatching Rule	Failure Probability					
		0.00	0.06	0.09	0.12	0.18	0.21
Mean Flow Time	SI	4072	4259	4265	4293	4380	4727
	EDD	4156	4333	4340	4463	4794	5305
	FCFS	4420	4561	4611	4615	4766	5736
Mean Tardiness	SI	139	217	224	312	447	788
	EDD	153	244	250	512	772	1308
	FCFS	470	554	591	594	759	1639
Average Machine Utilization	SI	52.37	52.31	52.09	50.79	44.36	40.66
	EDD	51.28	52.30	51.14	41.90	37.63	34.63
	FCFS	50.93	50.91	49.78	49.73	46.14	36.20

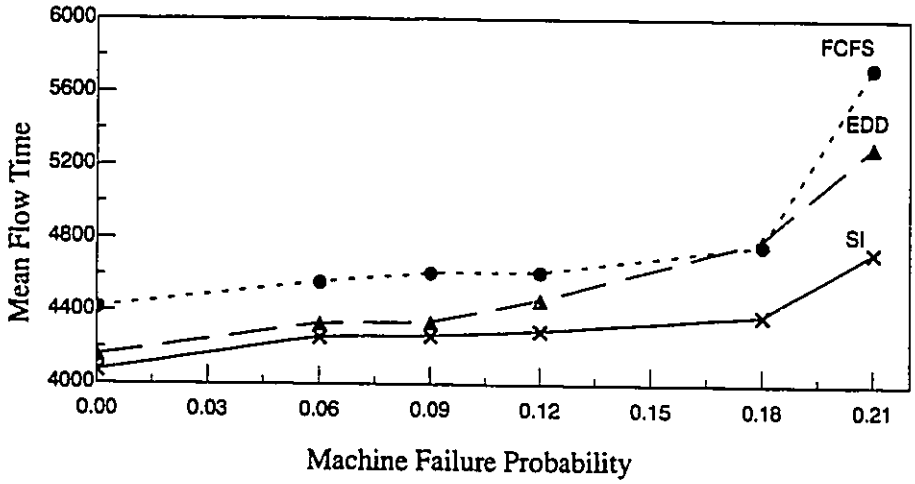


Figure 7.8: Mean flow time versus failure probability

shows that in most cases, the system performance degradation is initially gradual up to the machine failure probability of 0.09% (for EDD rule) and 0.12% (for FCFS and SI rule). Later in the run, when the probability of failure increased, the system performance deteriorated drastically. This is due to the larger number of jobs competing for the machines

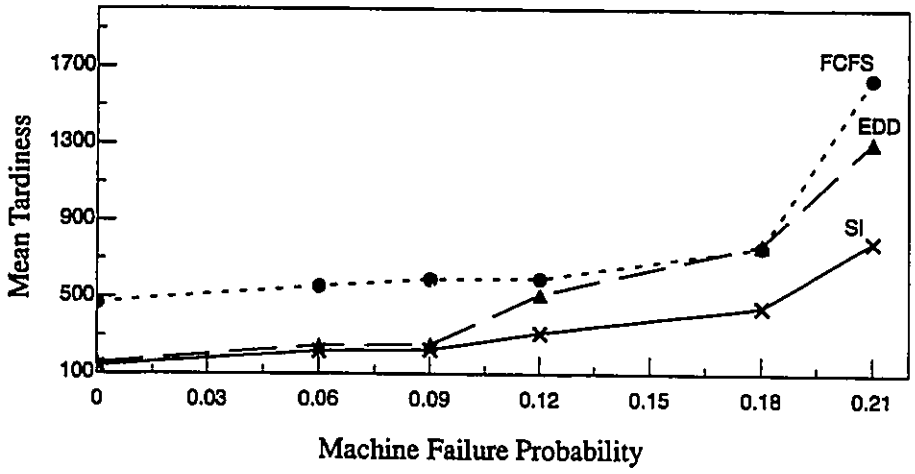


Figure 7.9: Mean tardiness versus failure probability

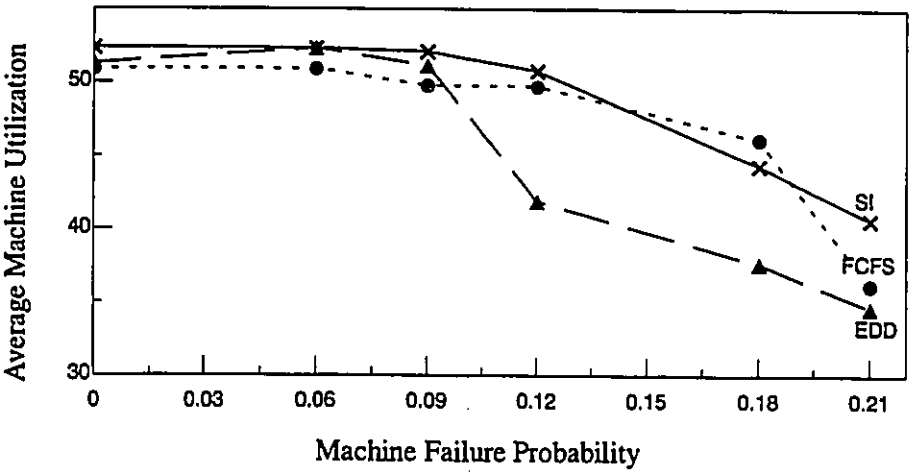


Figure 7.10: Average machine utilization versus failure probability

available for production. Figure 7.11 shows the Gantt chart of a schedule when there are no interruptions. The schedule shown in this chart is obtained for the case of SI rule. The effect of rescheduling algorithms is shown in Figures 7.12, 7.13, 7.14 and 7.15. Figure 7.12 shows the performance of rescheduling algorithms in the case of machine breakdowns. In this figure, the machine failure probability is fixed at 0.06%. There are six breakdowns in total and the machine that fails is randomly chosen. In the figure, machine 9 fails at time

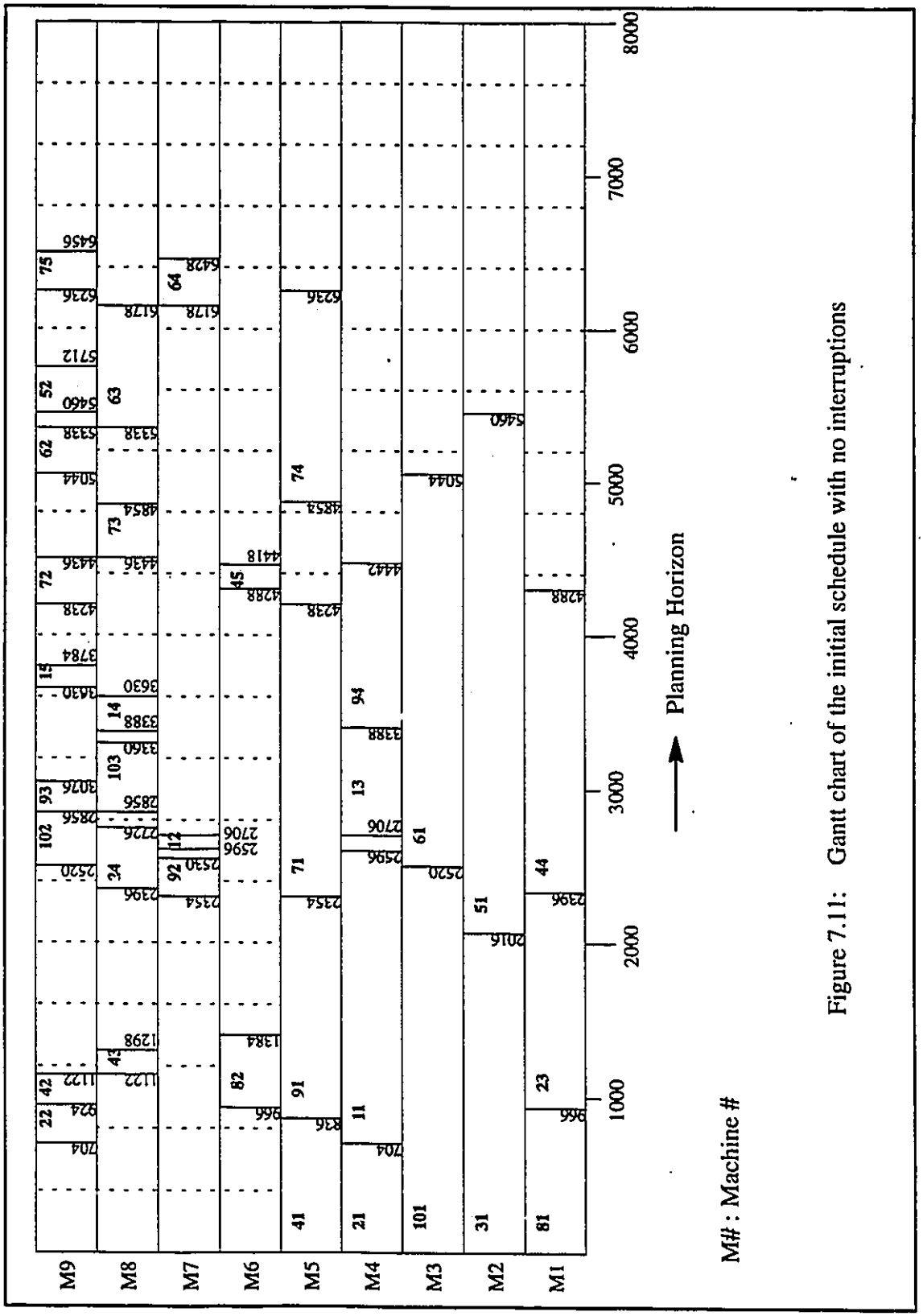


Figure 7.11: Gantt chart of the initial schedule with no interruptions

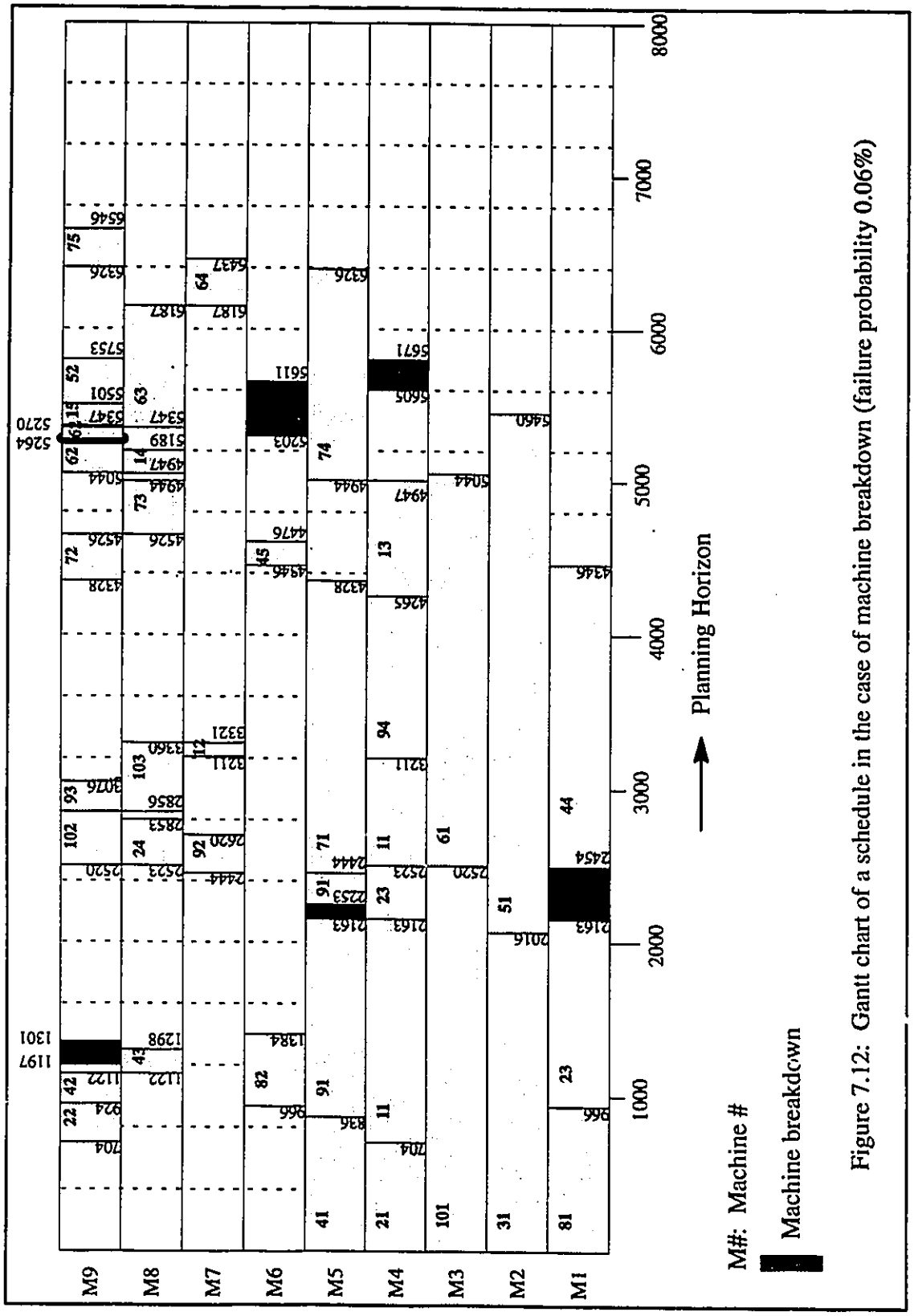


Figure 7.12: Gantt chart of a schedule in the case of machine breakdown (failure probability 0.06%)

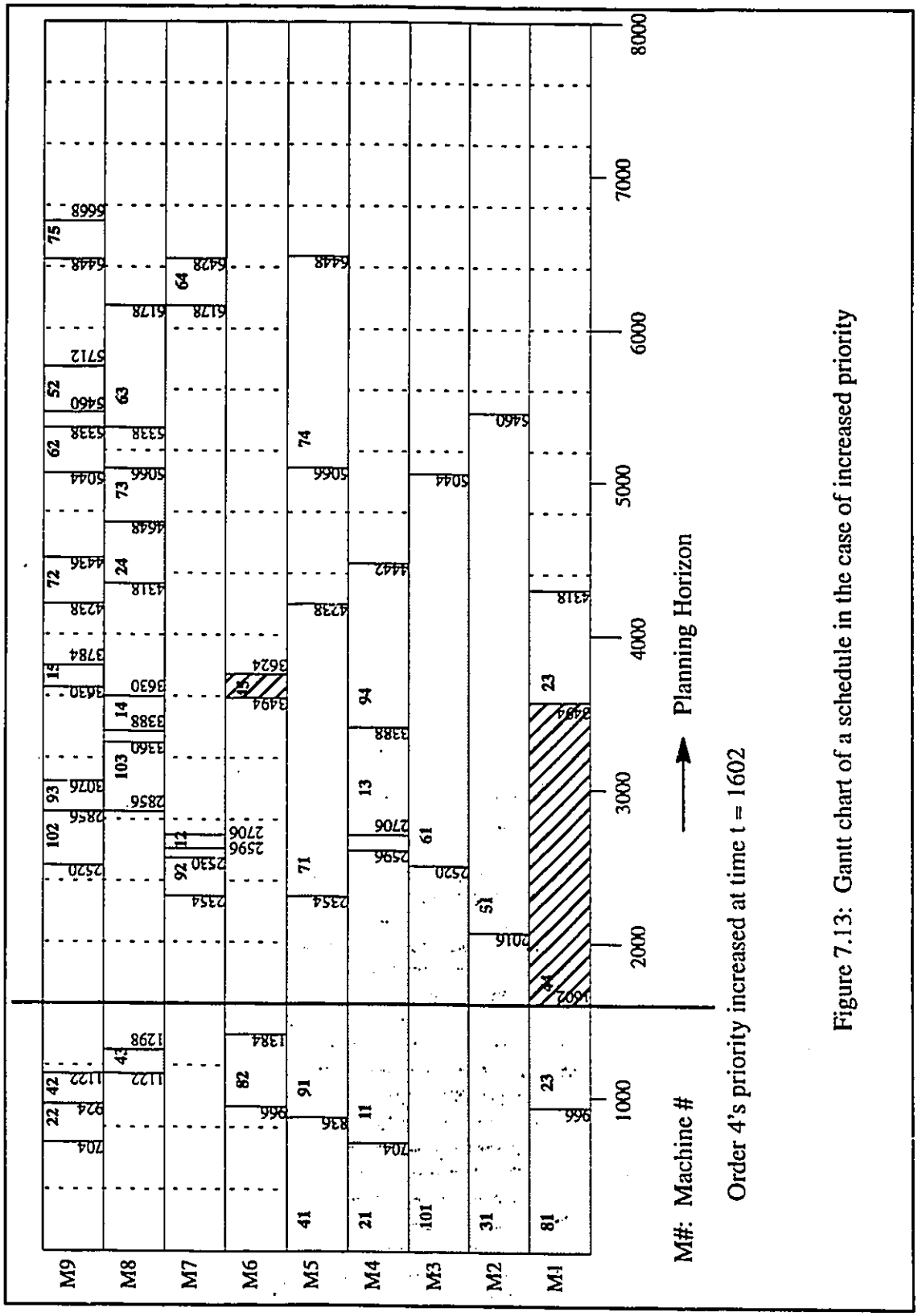


Figure 7.13: Gantt chart of a schedule in the case of increased priority

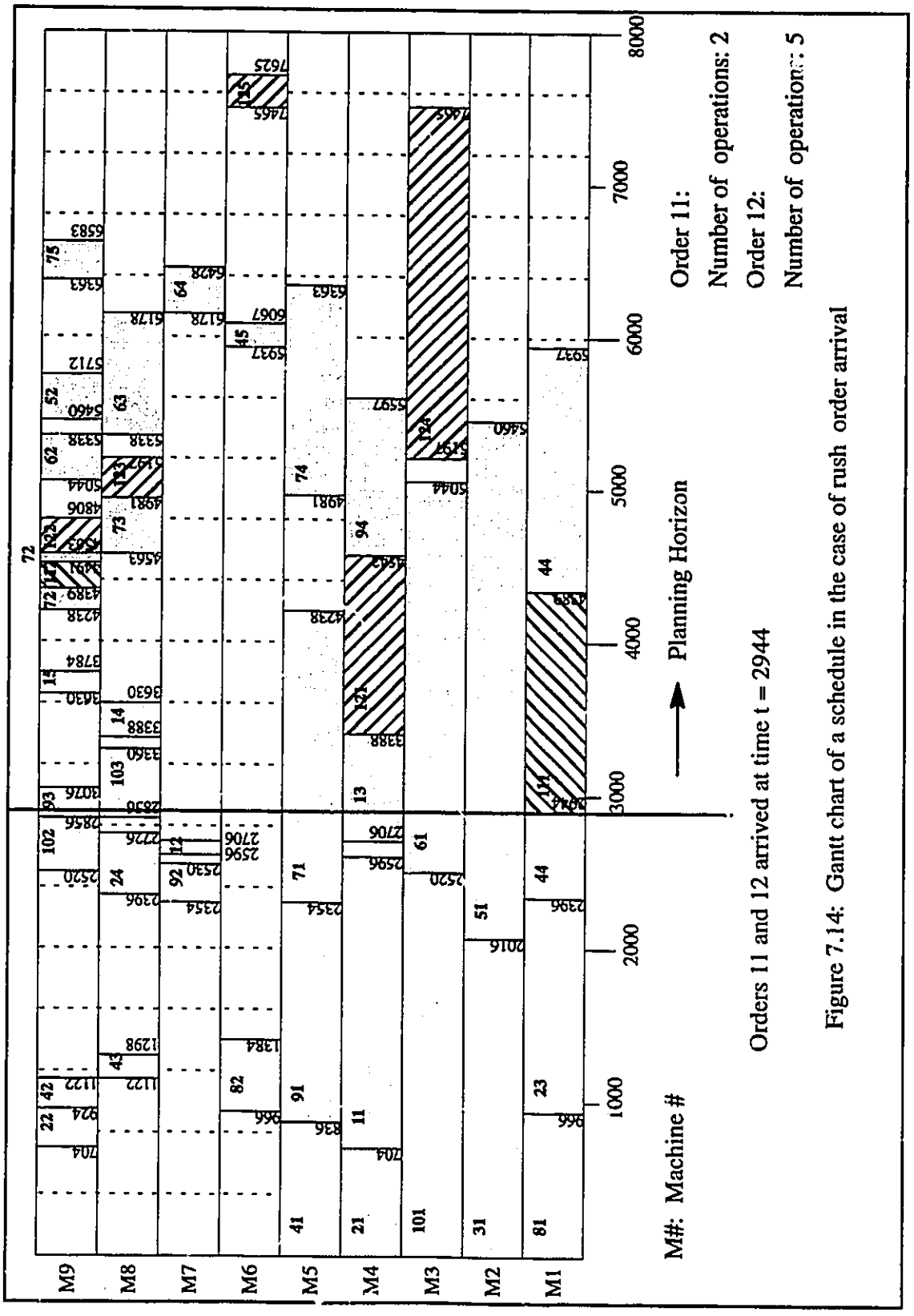


Figure 7.14: Gantt chart of a schedule in the case of rush order arrival

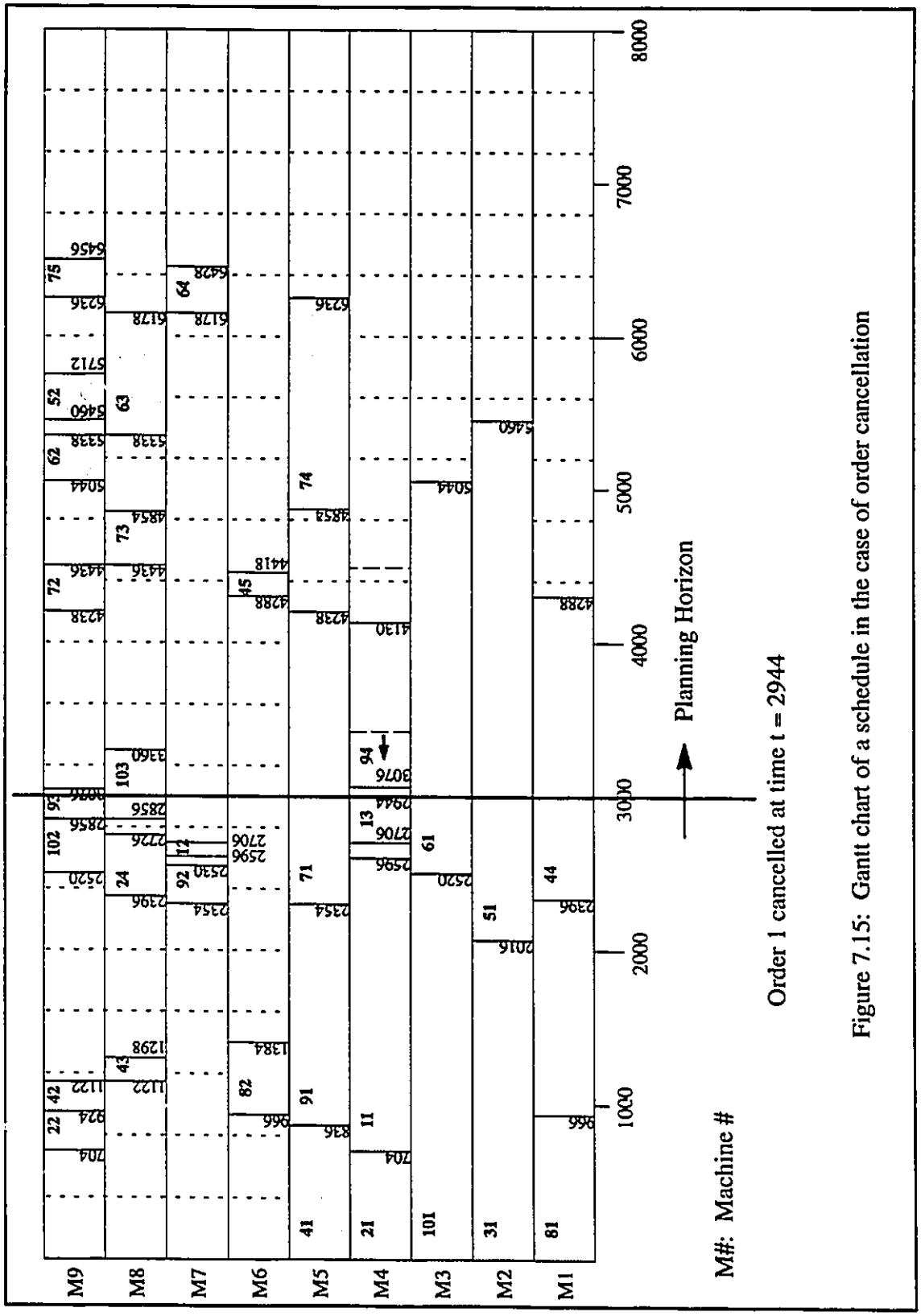


Figure 7.15: Gantt chart of a schedule in the case of order cancellation

t = 1197 and is expected to be down for 104 time units. Since there are no operations scheduled on this machine when the breakdown occurred, there is no change in the original schedule. At time t = 2163 two machines, machine 1 and machine 5 breaks down. Expected down time for machine 1 is 291 time units and for machine 5 is 90 time units. Operation 23 can also be performed on machine 4 and since setup time for operation 23 on machine 4 is less than the down time of machine 1, and priority of operation 23 is higher than operation 11, machine 4 is pre–empted and operation 23 is switched to machine 4. Operation 11 resumes after the completion of operation 23. This switchover delays operation 12 and operation 13 subsequently. On the other hand, because of machine 5's failure, operation 91 could not be switched to alternate machine as the setup time on alternate machine is higher than the expected down time. Therefore operation 91 resumes on the same machine as shown in the figure. The same affect is observed in the case of machine 9's failure at t = 5264, where operation 62 was stopped and then resumed on the same machine. The other two breakdowns of machine 6 and machine 4 at time t = 5203 and 5605 respectively, occurred when the machines were idle and hence no further change in the schedule was observed.

Figure 7.13 shows the performance of rescheduling algorithms in the case of increased order priority. In the figure, order 4's priority is increased at time t = 1602 which resulted pre–empting operation 23 on machine 1. Operation 45 is also pulled forward in time. Operation 23 starts again after the completion of operation 44 pushing operation 24 after operations 103 and 14 on machine 8. This increase in order 4's priority also delays operation 73, 74 and 75, as shown in the figure.

The effect of rush order arrival is shown in Figure 7.14. In the figure, two new orders (order 11 and 12) arrive at time t = 2944. The rush orders are assumed to be similar to one of the previously existing orders. Order 11 is similar to order 5 and has two operations. Order 12 is similar to order 4 and has 5 operations. Since operation 111's priority is higher

than operation 44 on machine 1, operation 44 is pre–empted at time t = 2944 and is resumed after the completion of operation 111. This affects completion of orders 4, 7 and 9 as shown in the figure.

The effect of order cancellation is shown in Figure 7.15. At time t = 2944 order 1 is cancelled, which pulled operation 94 forward in time.

7.5 Summary and Conclusions

In this research, we have developed rescheduling algorithms that generate a new schedule without re–evaluating all tasks in the old schedule. These algorithms use the system status as input and reschedule the tasks when disturbances occur. They reschedule only those tasks which are to be scheduled at the time of interruption, hence focusing only on local rescheduling as shown in Figures 7.11, 7.12, 7.13, 7.14 and 7.15. This is particularly important in real–time and dynamic scheduling environments in modern manufacturing systems. It is expected that the proposed algorithms can be used in existing computer–based scheduling systems. Since alternate machine choices are available for the tasks in most discrete manufacturing systems, the effect of disruptions on the system performance is minimized, as shown in Figures 7.8, 7.9 and 7.10. It is observed from these figures that system performance is not affected significantly up to a failure probability of 0.12% except in the case of EDD dispatching rule where it is 0.09%. The proposed algorithms can also be used in the selection of dispatching rules. It can be concluded from the results that the SI and FCFS dispatching rules performed better than the EDD rule for the three performance measures considered. In other words, there was less deviation in the system performance when the SI and FCFS rules were used compared with the EDD rule. We also observed that routing flexibility had a significant impact in the case of disruptions, and the presented framework displayed some desirable characteristics for real–time scheduling and control of discrete manufacturing systems.

CHAPTER 8

SYSTEM IMPLEMENTATION

In this chapter, all the algorithms and procedures discussed in previous chapters are implemented into a small and simple software structure. Details of each module are discussed separately and examples of system input and output format are given. The analysis of the schedules is performed by the analyzer.

8.1 Implementation of the Scheduler

The system was implemented on SUN computers, written in 'C' running in a UNIX environment. The basic genetic algorithms package (GENITOR) was obtained from Colorado State University, and was greatly modified for scheduling application. The basic steps of the developed system can be represented as follows:

(i) Data entry, i.e., enter all given input data regarding job orders and the manufacturing system.

(ii) Enter genetic algorithms parameters.

(iii) Enter primary and secondary performance criteria.

(iv) Enter scheduling parameters (scheduling rules, batch splitting policy, order regeneration method) or search the knowledge base for these parameters.

(v) Perform batch splitting.

(vi) Perform schedule optimization with genetic algorithms.

(vii) Specify 'N', the number of best schedules to be saved.

(viii) Output performance measure values and individual resource utilizations for all 'N' schedules.

(ix) If secondary performance criterion is specified, then output the best schedule based on the relative weights of primary and secondary performance criteria.

(x) Print list of start and completion times of all the job orders and summarize the results.

The overall structure of the system is given in Figure 8.1. The system consists of four modules, namely user interface, scheduler, analyzer and rescheduler. The input and output functions are performed through the user interface. The scheduler performs the necessary scheduling functions based on the input and output requirements. The output from the scheduler is a list of schedules, which are then analyzed by the analyzer. The output from the analyzer is the best schedule ready for implementation. If after the schedule implementation, any disturbance in the manufacturing system occurs, the schedule is posted on to the rescheduler where the necessary modifications are performed so that loss in efficiency is as small as possible. The performance criteria implemented into the system are as follows:

(i) Mean Flow Time

(ii) Makespan

(iii) Mean Tardiness

(iv) Maximum Tardiness

(v) Mean Waiting Time

(vi) Maximum Waiting Time

(vii) Average Machine Utilization

The implemented scheduling rules are as follows:

(i) Shortest Imminent processing time (SI)

(ii) Shortest Processing Time (SPT)

(iii) Longest Imminent processing time (LI)

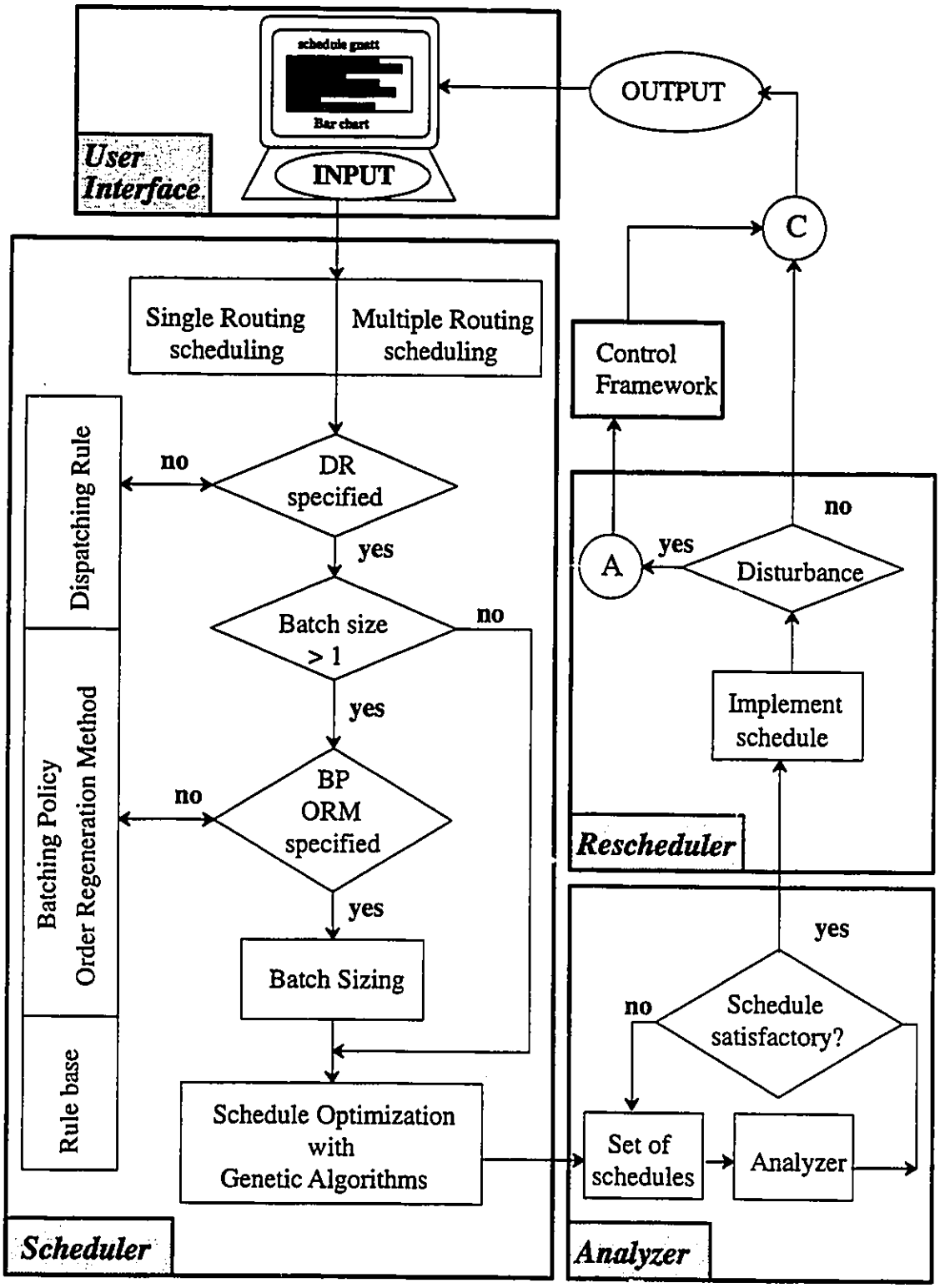


Figure 8.1: System structure

(iv) Longest Processing Time (LPT)

(v) Earliest Due Date (EDD)

(vi) Operation Due Date (OPNDD)

(vii) Least Slack Time (LST)

(viii) Slack per Remaining Processing time (S/RP)

(ix) Slack per Remaining number of Operations (S/RO)

(x) Shortest Remaining Processing Time (SRPT)

(xi) First Come First Serve (FCFS)

8.1.1 User Interface

The user interacts with the scheduler through the user interface. The following are the input requirements fed into the system:

(i) Number of orders to be scheduled

(ii) Number of machines in the manufacturing system

(iii) Maximum number of sub–batches allowed in any order

(iv) Minimum number of parts in any sub–batch

(v) Demand or order size of each order

(vi) Number of operations in each job

(vii) Operation – Machine requirement

(viii) Processing time of each operation

(ix) Setup time for each operation

(x) Due date of each order

(xi) Due date tightness factor

(xii) Primary performance criterion and its weight

(xiii) Secondary performance criterion and its weight

(xiv) Genetic algorithms parameters

(a) Population Size (PSIZE) (50 to 200)

(b) Selection Bias (SBIAS) (1.0 to 2.0)

(c) Adaptive Mutation rate (AMUT) (0.0 to 1.0)

(d) Number of Generations (NGEN) (1 to 10,000)

The suggested values of the genetic algorithms parameters falls within the range of the implemented parameters. This is incorporated in order to test other values of genetic algorithms parameters.

The optional input requirements include the batch splitting policy, order regeneration method, and dispatching rule to be used. If the user chooses not to specify these pieces of information, then the system searches its knowledge base to select the best parameter to be used. The knowledge base was created by running different combinations of twelve scheduling rules with seven performance measures, six batch splitting policies and three order regeneration methods.

The information entered is then saved in the respective files which are then used by the scheduler for schedule optimization. At each stage of data entry the user has a choice of modifying data, if entered incorrectly. Part 1 of Appendix C shows the input procedure of the user interface. This session was extracted while entering input data for problem B1 in chapter 5.

8.1.2 Scheduler

Once all the information for scheduling has been entered, the scheduler takes over and performs some initial analysis to determine whether the dispatching rule, batching policy and order regeneration methods has been specified. First, it enquires whether the dispatching rule has been specified. If the dispatching rule has been specified, then it looks to see whether the batch size is greater than unity. If the batch size for each part is found to be one, then a schedule optimization is performed using genetic algorithms. If the batch size

is more than one, and a batch splitting policy and order regeneration method has been specified then batch sizing is performed according to the batch splitting policy used and scheduling is performed by using genetic algorithms. If the dispatching rule, batch splitting policy or order regeneration method has not been specified by the user, then the system searches the knowledge base for the best dispatching rule, batch splitting policy, and order regeneration method to be used based on the chosen performance criteria. Part 2 of Appendix C gives the interaction with the scheduler where the user chooses to define his/her own parameters. The user is also asked to specify genetic algorithms parameters. The help session at various stages is also shown in this part. This session was recorded while interacting with the scheduler for problem B2 in Chapter 5 (ten–jobs/nine–machines).

The knowledge base contains some IF – THEN production rules in order to select the required scheduling parameter such as dispatching rule, batch splitting policy and/or order regeneration methods. An example of production rules when no scheduling parameter is specified by the user is given in Figure 8.2. Figure 8.3 gives a sample rule for the case when

IF Performance criteria is to minimize the mean flow time

THEN Select, Shortest Imminent processing time dispatching rule,
Policy 5 (split based on operations) batch splitting policy, and
Dynamic Continuous order regeneration method

IF Performance criteria is to minimize the makespan

THEN Select, Longest Imminent processing time dispatching rule,
Policy 4 (split based on machines) batch splitting policy, and
Dynamic Constant order regeneration method

Figure 8.2: Examples of production rules when none of the parameters are specified

performance measure, batch splitting policy, and order regeneration method are specified, and knowledge base is searched for the best dispatching rule. Similar rules were defined

IF Performance Criteria selected is to minimize the mean waiting time, and
Batch splitting policy specified is 2 (split equally) and
Order regeneration method selected is Static
THEN Select Shortest Remaining Processing Time as the dispatching rule

Figure 8.3: Example of a production rule when dispatching rule is not specified

for other combinations such as when only performance measure and batch splitting policy are known and knowledge base is searched for order regeneration methods and dispatching rule. When the performance measure and dispatching rules are specified the knowledge base is searched for the batch splitting policy and order regeneration methods.

8.1.3 Analyzer

The output from the scheduler is a list of 'n' schedules, where n is specified by the user. Attached to this list of schedules is all the performance measure values. The analyzer analyzes the schedule and outputs the best schedule based on the primary and secondary criteria and their weights. This schedule is then selected subsequently for implementation.

8.1.4 Rescheduler

After the schedule has been implemented, the manufacturing system is checked for possible disturbances. If a disturbance is found, the control framework described in chapter 7 is called and the schedule is modified according to the prevailing disturbance.

The special features of the system includes:

(i) Modify existing data, i.e., after data entry session or some other time in future, if the user wishes to modify or change some existing data for running a different experiment,

it is made possible through the individual data files created during the first interactive session. The very first information that is required of the user is whether the user wants to enter an entirely new set of data or modify existing data. This is shown in Figure 8.4.

```
Do you wish to enter new data or modify existing data ?

1. Enter entirely new scheduling data.
2. Modify existing scheduling data.

Enter your choice :-> 2

You can change the following information in the existing files:

1. Modify the Demand of the order
2. Modify the Machine Requirement of a particular task
3. Modify the Setup time of a particular task
4. Modify the processing time of a particular task
5. Add a new order
6. Add a new machine

Enter your choice :-> 1
```

Figure 8.4: Sample session of data modification

(ii) Correct data at any time during data entry, i.e., during the data entry session, if the user finds that an incorrect datum has been entered, it is made possible to rectify the incorrect value. After each data entry session (for example, processing time entry for 50 operations is one session), the system responds with a list of values that have been entered and a question about whether the user wishes to change any value. If the user finds that some of the values entered are incorrect, he or she can do so by answering affirmatively at this time.

(iii) Access help where in doubt, i.e., the help module is provided which the user can access at any time during the data entry session. User can enter '?' to access help wherever it is available.

8.2 System Output

The output of the system is a set of 'n' best schedules. For all 'n' schedules, all the performance criterion values and individual machine utilizations are also posted. The user may choose any of them for subsequent implementation. If the decision is left to the analyzer, then the best schedule is posted based on the weights of primary and secondary performance criteria. Finally, for each operation, start and completion times as well as the detailed Gantt chart of the schedule are produced. This is shown in Part 3 of Appendix C.

CHAPTER 9

CONCLUSIONS

9.1 Research Summary

This research effort explored the feasibility and advantages of using genetic algorithms to solve the production scheduling problems in discrete manufacturing systems. It is well established that the production scheduling problem is NP–complete, indicating the time required to compute an optimal solution to the problem increases exponentially with the size of the problem and no efficient polynomial time algorithms are likely. Because of this reason, the current emphasis is to find a methodology that can perform well both in terms of nearness to optimality and computation time. The major issues addressed in this research were:

(a) *Application of genetic algorithms to the schedule optimization problem.* Two areas of research were covered under this topic. First, genetic algorithms were used for the case when there is only one process plan available for part processing. This resulted in no routing flexibility i.e., each task could be designated to only one machine. Second, genetic algorithms approach was applied in the case of multiple process plans resulting in the routing flexibility. Because of routing flexibility, the possible number of schedules to be optimized increased, and hence dispatching rules were employed along with the genetic algorithms.

(b) *Introduction of batch-splitting policies and order regeneration methods to improve the performance of the manufacturing system.* The research introduced three new batch splitting policies for batch sizing and experiments were performed for detailed analysis of these batching policies. It was observed that the manufacturing performance indeed increased in the presence of batch splitting.

(c) *Dynamic scheduling.* Rescheduling algorithms were developed to minimize the schedule disruption. The different uncertainties considered were: (i) unexpected machine breakdowns, (ii) rush order arrival, (iii) increased order priority and, (iv) order cancellations. The developed rescheduling algorithms revise only those tasks that need to be revised. This is superior to previous work cited in literature, where rescheduling is usually performed on the entire set of operations. The performance of the developed algorithms was illustrated with a numerical example.

(d) *System Implementation.* Finally, a prototype system has been implemented which consists of user interface, scheduler and analyzer.

The developed scheduling and rescheduling algorithms were illustrated with numerical examples. Genetic algorithms parameters were selected through extensive experimentations. In order to test the validity of the genetic algorithms approach, various example problems from the published literature were formulated and run using the developed scheduler. The results indicated that the genetic algorithms approach yielded solutions close to optimal (and in some cases optimal). The computational time requirements were also modest, obtaining solutions in a few minutes. This makes the proposed scheduling approach suitable for real-time applications. In terms of solution nearness to optimal or speed, the genetic algorithms approach compared favorably with other approaches.

9.2 Research Conclusions

The conclusions made in this section are primarily based on comparing the results reported in the literature and those produced in this research. In order to clearly identify these conclusions, this section is divided into two parts. In the first part, general conclusions are presented. In the second part, specific conclusions are given based on the numerical results generated by the conducted experiments.

9.2.1 General Conclusions

1. It is observed that in many scheduling situations, scheduling methods are not the only means to control manufacturing performance. In fact, batch splitting or batch sizing has a major impact on the manufacturing system performance, such as reducing the mean flow time and tardiness. Parts are batched together to avoid small runs and frequent machine setups. As batch sizes become small, shop time is consumed with nonproductive setups, thus reducing performance. On the other hand, large batches tie up machines for extended periods of time, thus increasing unit flow time. In most cases batch sizes are predetermined; therefore, the possibility for performance improvement available at the scheduling level is severely restricted. One numerical example showed that batch splitting can have a significant impact on the shop performance. Hence, batching decisions are certainly required when jobs are to be scheduled in an efficient manner.
2. Most of the research on batch splitting is based on queuing models and require cost estimates for setup, inventory holding, etc. which are not usually readily available. This calls for a batching methodology where batches can be formed based on the information that could easily be obtained. A part process plan method has been developed in this research, where the information available through the jobs process plans is used for batching and no cost related data are required. The new splitting

policies were compared to existing ones and the results show that in some cases the newly developed policies, such as splitting the order according to number of operations in the job, performed better than the existing ones, such as splitting the order to have equal jobs in all orders.

3. An automated manufacturing system is an open dynamic flow shop / job shop model with deterministic data variability. Since automated manufacturing systems exist in a dynamic environment, efficient scheduling and rescheduling algorithms must have the capability of changing decisions based on the operating environment over time. In real–life situations, many dynamic events occur on the shop floor that require adjustments to existing schedules. Uncertainties in the production environment inevitably result in deviations from the generated schedules. Therefore, dynamic scheduling algorithms are absolutely required. The developed rescheduling algorithms proved successful in accommodating the dynamic changes on the shop floor with relatively little loss in performance.

4. It is observed that in the case of multiple process plans and hence multiple routes, direct optimization methods require long computation time in order to obtain satisfactory schedules due to the mushrooming search space. Because of these multiple process plans, the number of feasible schedules grows, and the problem of finding an optimal solution changes to a situation where the aim of the scheduler is to obtain a 'near optimal' or a 'satisfactory' schedule. This is because of the vast number of schedule combinations available. This opened a new area in which dispatching rules are used to release the jobs to the manufacturing resources. To date no single dispatching rule has been declared as the clear winner in all manufacturing situations. This is because the use of a particular dispatching rule is very much dependent on the chosen performance criteria and the characteristics of the production system. This is illustrated in this thesis with numerical examples. In this research, twelve different

dispatching rules and seven performance criteria have been employed and the performance of each rule has been studied extensively with respect to each of the performance criteria. It was observed that when throughput times and due dates are important (which is often the case) shortest imminent processing time (SI), operation due date (OPNDD) and least slack time (LST) can satisfactorily be adopted as priority dispatching rules.

5. Most of the research in scheduling is based on operations research models, simulation or artificial intelligence techniques. The genetic algorithms approach, developed in this research, has been observed to be very effective and efficient in computation time for generating the optimal or near optimal schedules compared to other methods. They are more suitable for optimization problems, not only from an economic point of view, but also from a practical application point of view. Because there are multiple search points in a population instead of a single point, the probability of a search getting trapped into local minima is greatly reduced. The genetic algorithms approach certainly has an advantage because of the fixed size search space as shown in Figure 9.1. This advantage was verified by testing several examples acquired from the literature.

6. Choosing an appropriate representation is the first step in applying GAs to any optimization problem. Although bit string representations have been used for scheduling problems, they do not represent the simple scheduling constraints such as precedence among operations. Value string representations were found to be more appropriate for the scheduling problems and because of these representations, simple genetic operators could not be used. Specific operators should be developed that can incorporate scheduling constraints and create feasible schedules. Because of the complexity of scheduling problems, these operators are difficult to develop. If a proper representation could be found, unfeasible schedules would never be generated.

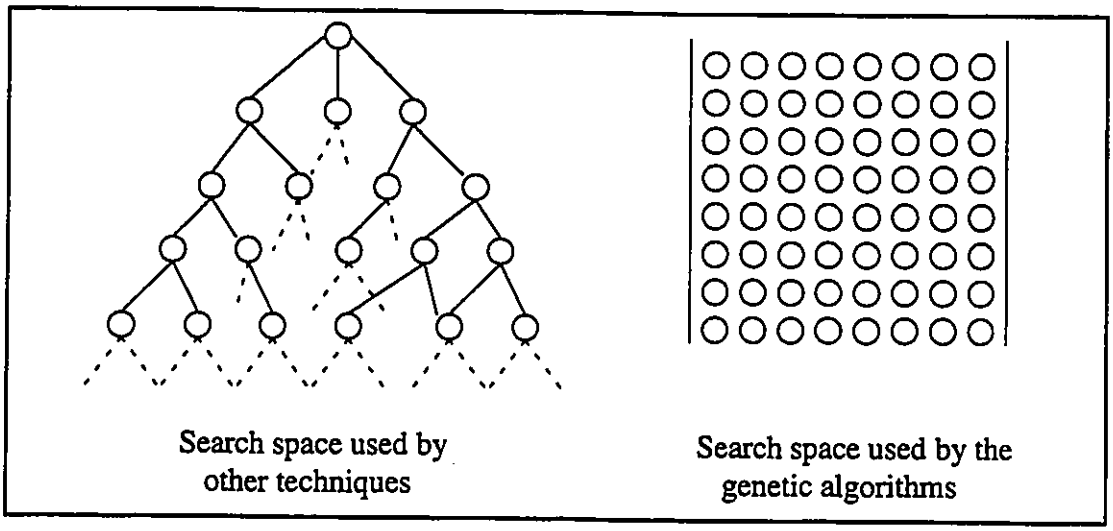


Figure 9.1: Search space used by conventional and genetic algorithms

Hence, in order to obtain an efficient schedule, proper schedule representation is a must.

7. The choice of parameters can have significant impact on the effectiveness of optimization algorithms. It has been observed that the performance of genetic algorithms is very much dependent on the selected parameters such as 'ation size, selection bias and the mutation rate. For binary encoding, robust genetic algorithm parameters are known and could be used for optimization purposes. For problems where binary encoding could not be used, such as scheduling problems (where operations number or machine itself is used for encoding), selecting parameters can be very difficult and time consuming. In practice, however, there is no single a priori choice for these values that would apply for all problems. It was observed that optimal parameters settings varies from problem to problem, but robust settings work well across a variety of problems. For this reason, many GA practitioners use only traditional gene: · lgorithms. Therefore, appropriate parameter values have to be found for problems incorporating non–binary encoding.

8. The proposed genetic algorithm approach could be adapted to different manufacturing situations with certain modifications pertinent to the given system. Initially the scheduling algorithm was developed for job shop situations, but testing the algorithms in flow shops also produced good results. Hence, the proposed algorithm can be used for both manufacturing situations, namely job shops and flow shops.
9. The genetic algorithms approach is much simpler to apply than other optimization and scheduling approaches.

9.2.2 Specific Conclusions

To validate the results, we tested the developed algorithms on 20 example problems that were acquired from various published articles and dissertations. Based on the obtained numerical results, the following specific conclusions can be made concerning the genetic algorithms performance:

1. For the examples tested, the performance of genetic algorithms approach is much better than the existing methods both in terms of optimizing the specified performance criteria as well as the computation time.
2. The performance of the manufacturing system was improved significantly using batch splitting. When the mean flow time is used as the performance criterion, splitting according to number of operations gave the best results. The performance was improved by almost 35.0% in the case of static and dynamic constant order regeneration and by 45.0% in case of the dynamic continuous order regeneration. This was irrespective of the dispatching rules used. Similar tests were performed with other performance measures such as makespan, mean waiting time and machine utilization. In all cases, the newly developed batching policies such as splitting the order according to number of operations in a job performed better than the existing ones,

with the exception of makespan. Comparable results were obtained for the test cases where the optimal results could not be achieved.

3. For the various examples tested, the genetic algorithms approach performed significantly better than other approaches in terms of finding the optimal solutions. In problems where the optimal solutions were known, shorter computation time was required by the GAs method to achieve this value. The saving in computation time was up to 50% in some cases. For the example problems whose optimal solutions were not known, GAs was able to achieve either the same value or better in less computational time than was reported. The exception was a 10x10 benchmark problem where the optimal solution could not be found in a reasonable amount of time by the GAs scheduling algorithm. The time required to find the optimal solution to this problem is reported to be 5.00 hrs of CPU time. However, by using the GAs approach, the saving in computation time is by at least a factor of 28 with only 6.9% loss in the optimal value. For cases where this approach could not obtain optimal results, it is possible to improve the performance of the system, but at the expense of computation time.

4. In this research, the dynamic scheduling aspects were considered and rescheduling algorithms were developed for four kinds of uncertainties. These uncertainties include unforeseen machine breakdown, rush order arrival, order cancellation and changed order priority. The developed algorithms reschedule only interrupted tasks, hence focusing only on local rescheduling. This maintains the stability of existing schedules and provides quick solutions. These algorithms were tested on three different performance criteria, namely mean flow time, mean tardiness and average machine utilization. The three dispatching rules used to release the jobs were the shortest imminent processing time, earliest due date and first come first served. The results indicate that the proposed algorithms are effective and could easily be used to

dynamically schedule the manufacturing systems in the presence of disturbances. When the mean flow time is used as the performance criterion, the loss in the performance is only 16.0% of the original value. This is in the case of a machine failure probability of 0.21%, i.e., any machine in the system could fail with this probability. For this probability, the number of times machines fail during the scheduling horizon is 21. This degradation is much lower (7.5%) in the case of 0.18% failure probability (number of times machines fail = 18). Similar results were obtained for the other two performance measures.

5. Decisions regarding selecting the appropriate parameter values for genetic algorithms were taken through a series of experimentations. These parameters include population size, selection bias and mutation rate. GAs generally perform poorly with small populations (<= 75) as they provide an insufficient sample size for experimentation. A large population (>= 150) requires more evaluations per generation, possibly resulting in an unacceptably slow rate of convergence. Selection bias is a floating point number used in the selection of two schedules for genetic reproduction. This number specifies the amount of preference to be given to the superior individuals in a genetic population. Mutation is a secondary search operator that increases variability in the population. The empirical results indicate that our approach gives optimal / near optimal solutions when using the following parameter values:

   (a) Population Size (PSIZE): 75 to 150 (varies with the problem size)

   (b) Selection Bias (SBIAS): 1.9 (Usually >= 1.5)

   (c) Mutation Rate (AMUT): 0.1 (Usually <=0.3)

Through the implementation and a series of experiments we demonstrated both the viability and high levels of performance of the proposed genetic algorithms scheduling approach. They have considerable potential and further investigations along these lines are well worth the effort. Our experiments showed that this approach represents a good

scheduling alternative. It is reasonably fast, gradual (users can watch the schedule progress) and gives better results than other approaches most widely used for this purpose. It was noticed that in almost all cases, genetic algorithms achieved a better performance both in terms of the objective function value and computational time. It is expected that the new genetic scheduler can adapt to real situations and control the scheduling systems effectively, thereby resulting in further increases in productivity.

In this research, we have investigated the use of GA–based techniques in the context of a manufacturing scheduling problem. This work, and other current investigations, demonstrate that the genetic algorithms method is a broad spectrum, approximate search procedure with application in diverse problem areas. The method does not depend upon an underlying continuity of the search space and requires no information other than fitness or payoff values. Since genetic algorithms always work with a fixed population size, i.e., the number of strings (chromosomes) is fixed in a population and does not change with the generation, there is a tremendous reduction in the space in which GAs look for the optimal solution. Also, since search is limited to a fixed size search space, the time required to reach an optimal or near optimal solution is expected to be small compared to other search techniques. Together, these qualities permit the use of GAs in complex areas such as manufacturing scheduling.

9.3 Research Contributions and Achievements

The research contributions and achievements are summarized as follows:

1. **Introduction of novel genetic formulation of manufacturing scheduling**

Previous work in manufacturing scheduling is limited to knowledge–based approaches, simulation techniques, and operations research methods. These techniques are time consuming. Genetic algorithms approach to scheduling is an attractive and feasible approach that seems to be promising in scheduling because of its ability to work efficiently

with (NP–complete) problems. In this research, the problem of scheduling has been formulated and solved using genetic algorithms.

Since genetic algorithms always work with a fixed population size, i.e., the number of strings (chromosomes) is fixed in a population and does not change with the generation, there is a tremendous reduction in the space in which GAs looks for the optimal solution. As far as time complexity is concerned, since the search is limited to a fixed size search space, the time required to reach an optimal/near optimal solution is small compared to other techniques. The performance of genetic algorithms was compared with other scheduling techniques for time complexity and it was observed that, in most cases, genetic algorithms outperformed other scheduling approaches.

In order to validate the results obtained by the genetic algorithms approach, several example problems from the literature using heuristics, mathematical programming or artificial intelligence techniques were tested. When optimal solutions were known, the GAs approach was able to obtain the optimal result in about half the time that was previously reported. GAs approach was also able to achieve better values of performance criteria for some test cases.

2. Use of dispatching rules while optimizing with genetic algorithms

From the available literature, it was observed that genetic algorithms have been applied to manufacturing scheduling problems, but to a very limited extent. To date scheduling with genetic algorithms has been limited to a single performance measure without using dispatching rules. Dispatching rules are particularly useful when multiple routes and process plans alternatives exist. The use of a particular dispatching rule is very much dependent on the chosen performance measure and the production system tested. In this research, twelve different dispatching rules in combination with seven performance measures have been used while scheduling with genetic algorithms. Given a performance

measure and a scheduling rule, the genetic algorithms approach determines the best schedule to be used.

3. Consideration of alternate routings and order sizes greater than one and dispatching rules simultaneously during scheduling with genetic algorithms

In other scheduling research, it is often noticed that while testing the scheduling rules, authors tend to use a single job in each order, which is rarely the case in practice. Also, alternate routings are usually considered with single job in each order. This research considers both alternate routings and order size greater than one and simultaneously uses them with dispatching rules during scheduling with genetic algorithms. The experiments were run on two example problems, one for a simple case (4 jobs, 6 machines, 3 operations/job) and another for a complex case (10 jobs, 9 machines, with operations varying from 1 to 5 for each job). The results obtained were satisfactory and the difference in the results, i.e., whether the performance of one rule is better or worse than others, may have been due to a special characteristic of our chosen problem, that is, each order had more than one part, and the performance was assessed according to when the order (not the jobs within an order) was completed.

4. Introduction of three new batch splitting policies

When an order quantity is more than one, the order size or lot sizes are important for deciding the optimal number of jobs per order. Six different batch splitting policies have been used in this research in conjunction with three job regeneration methods. Policies (iv), (v) and (vi) were newly introduced in this research. The different batch splitting policies that were used in this research are:

(i) No batch splitting

(ii) Splitting the batches in half

(iii) Splitting the batches in equal sizes

(iv) Machine dependent batch splitting

(v) Task dependent batch splitting

(vi) Processing time dependent batch splitting.

Regeneration methods are used to release the job orders into the shop floor. After the jobs are grouped into small orders, they are held into the order pool and are released only when predetermined conditions are met.

5. Development of rescheduling algorithms

The generation of new and modified production schedules is becoming a necessity in today's complex manufacturing environment. Once an initial schedule has been implemented, it is almost immediately subjected to the new production conditions, demands and constraints. Uncertainties in the production environment and modelling limitations inevitably result in deviations from the generated schedules. This makes rescheduling or reactive scheduling essential. Four different types of uncertainties that normally cause discrepancies between the actual output and the planned output were considered in this research. These included unforeseen machine breakdowns, increased order priority, rush orders arrival and order cancellations. The proposed algorithms revise only those operations that need to be rescheduled. They can be used in conjunction with the existing scheduling methods to improve the efficiency of automated manufacturing systems.

6. System implementation

In this research, a scheduling prototype system has been developed where the user can input data using the user interface. A small knowledge base has been developed consisting of 154 simple rules for the selection of scheduling parameters such as best batching policy, regeneration method and dispatching rule. Given the performance criteria, the user has an option of either searching the knowledge base for scheduling parameters or specifying his/her own parameters for scheduling. Once all the required data has been entered, the system produces several satisfactory (optimal/near optimal) schedules using genetic algorithms technique.

Once the data is entered, tests can be performed for different combinations of scheduling parameters. The user specifies the primary and secondary performance measures to be used for scheduling. Weights to be assigned to both measures are also specified. Optimization with genetic algorithms starts with the randomly generated population of schedules (e.g. 50 schedules) and are run for some specified number of generations (e.g. 1000). Genetic operators are applied on the population of schedules of each generation, introducing new schedules into the population with better objective function values and discarding schedules with low objective values. The best schedule of each generation is saved for further analysis. The output of this stage is the list of several good schedules, various performance measure values and machine utilizations. The output analyzer examines the output based on the weights of the performance measures, produces the best schedule and presents it in the form of a Gantt chart.

9.4 Recommendations for Future Research

The approach suggested in this research to solve the manufacturing scheduling problem is based on a few assumptions given in the beginning of chapters four and five. There are some general directions in which the work described in this research may be extended. The following areas are recommended for future research:

(i) In this research, it was assumed that all sets of jobs are available at the beginning of the scheduling period. In large manufacturing systems, where the number of jobs to be scheduled could be quite large, the selection of the initial sets of jobs plays an important role in reducing job congestion and control of the whole manufacturing system. This aspect of the manufacturing problem needs to be addressed. Also, arrivals to the shop should be taken into consideration, and the relationship between batch sizes and arrival process needs to be explored.

(ii) In studying the scheduling problem, an assumption was made that transportation time between facilities is negligible. In general, this is not the case. Scheduling of material handling equipments (robots, AGVs, etc.) in conjunction with parts and machines is a potential area of future research.

(iii) Another relevant issue is the scheduling or dispatching policies used to control the production. Further research is needed to study the impact of dynamic scheduling procedures on the problems of routing, batch sizing and scheduling. This is particularly important in the case of routing problems, since routing decisions may be made dynamically. The knowledge base developed regarding the selection of dispatching rules is very small at the current stage where only twelve dispatching rules have been used. More dispatching rules need to be added to this knowledge base for productivity improvement. With the realization that each manufacturing system is different, there is a need for further research to develop new rules and to continue testing existing ones.

(iv) This research assumes that the sequence of tasks within a job is fixed resulting in a fixed sequencing problem [Lin and Soldberg 1991]. Further research is required for the flexible sequencing case (where certain tasks can be performed in an arbitrary order) and flexible processing case (where neither tasks nor their sequence are fixed). In this case, all possible alternative operations and sequence of manufacturing a part are considered in real time to make routing decisions. Such flexibility in sequencing operations, if properly exploited, may have beneficial effects on various measures of system performance, by relieving bottlenecks or avoiding down machines [Rachamadugu and Stecke, 1993].

(v) Since the successful implementation of GAs depends on the careful selection of parameter values, more rigorous methods have to be researched in order to obtain a universal set of parameter values. Furthermore, it appears that interaction between

these parameters also has a significant impact on the GAs' performance; hence, this aspect of the problem needs to be looked into the future research.

(vi) In this research, we used edge recombination operator (SPPS problem) and reduced surrogate crossover operator (MPPS problem), while a number of alternatives are available in the literature. It would be an interesting study to investigate the performance of other alternative crossovers for the scheduling problems.

(vii) Another topic for future research is the use of genetic algorithms to rapidly reduce the search space to a size that can subsequently be handled by deterministic search algorithms [Grefenstette, 1987]. The idea here is to use the genetic algorithms to identify the possible 'hills' in the search space, which can later be climbed by conventional algorithms. Such a hybridization of search techniques, which could boost search performance in difficult tasks such as scheduling, needs to be researched.

(viii) In this research, dynamic scheduling or reactive scheduling function is carried out by several rescheduling procedures that are not incorporated into genetic algorithms. Therefore, improvements in search and scheduling methods are necessary in the provided genetic algorithms in order to use them for dynamic scheduling.

(ix) In this research, the user interacts with the scheduler through a user interface. In order to refine the scheduling process, building an expert system that serves as an advisor to the user (scheduler) can be useful. Since strictly deterministic algorithms are inadequate in generation decisions and parameters needed to operate the system, the artificial intelligence approach can provide access to an expert system's real–time data base (knowledge base) to produce some general guidelines. This knowledge base can be updated as the process requires. The expert system should be designed to deal with the following tasks:

(a) Forming the set of jobs from orders (demand)

(b) Generating production priority indices

(c) Determining which dispatching rule, batching policy, and order regeneration method to be used

(x) An extension to this research could be the addition of a learning and knowledge acquisition module. Great advantages can be gained if the current scheduler develops the ability to learn and acquire new knowledge efficiently. The scheduler can be customized as well as expanded automatically for any type of manufacturing system. Highly sophisticated intelligence can be achieved when the scheduler is able to acquire knowledge from human schedulers, learn from its own experience and keep updating its knowledge base for the particular system to which it is applied. Completing the design of learning and knowledge acquisition should provide a promising area of future research.

REFERENCES

[1] Adams, J, E. Balas, and D. Zawack (1988), "The Shifting Bottleneck Procedure for Job Shop Scheduling", *Management Science*, Vol. 34, No. 3, pp. 391–401.

[2] Afentakis, P. (1985), "Simultaneous Lot Sizing and Sequencing for Multistage Production Systems", *IIE Transactions*, Vol. 17, No. 4, pp. 327–331.

[3] Atabakhsh, H. (1991), "A survey of constraint based scheduling systems using an artificial intelligence approach", *Artificial Intelligence in Engineering*, Vol. 6, No. 2, pp. 58–73.

[4] Bagchi, S., S. Uckun, Y. Miyabe and K. Kawamura (1991), "Exploring Problem–specific Recombination Operators for Job Shop Scheduling", *Proc. of the Fourth Int. Conf. on Genetic Algorithms*, San Diego, CA, pp. 10 –17.

[5] Baker, K. R. (1974), *Introduction to Sequencing and Scheduling*, John Wiley & Sons Inc., New York.

[6] Baker, C. T. and S. Dzielinski (1960), "Simulation in a Simplified Job Shop", *Management Science*, Vol. 6, pp. 311–323.

[7] Bakshi, M. S. and S. R. Arora (1969), "The Sequencing Problem", *Management Science*, Vol. 16, pp. 247–263.

[8] Ballakur, A. and H. J. Steudal (1984), "Integration of Job Shop Control Systems: A State–of–the–Art Review", *Journal of Manufacturing Systems*, Vol. 3, No. 1, pp. 71–79.

[9] Barker J. R., and G. B. McMahon (1985), "Scheduling the General Job–Shop", *Management Science*, Vol. 31, No. 5, pp. 594–598.

[10] Bertrand, J. W. M. (1985), "Multiproduct Optimal Batch Sizes with In–Process Inventories and Multi Work Centers", *IIE Transactions*, Vol. 17, No. 2, pp. 157–163.

[11] Biggs, J. R. (1979), "Heuristic Lot–Sizing and Sequencing Rules in a Multistage Production Inventory System", *Decision Sciences*, No. 10, pp 96–115.

[12] Biggs, J. R. (1985), "Priority Rules for Shop Floor Control in a Material Requirements Planning System Under Various Levels of Capacity", *International Journal of Production Management*, Vol. 23, No. 1, pp. 33–46.

[13] Blackstone, J. H. Jr., D. T. Phillips and G. L. Hogg (1982), "A state–of–the–art survey of dispatching rules for manufacturing job shop operations", *International Journal of Productions Research*, Vol. 20, No. 1, pp. 27–45.

[14] Booker, L. (1987), "Improving Search in Genetic Algorithms", *Genetic Algorithms and Simulated Annealing* (Ed. Lawrence Davis), BBN Laboratories, Cambridge, MA, pp. 61–73.

[15] Burns, R. (1993), "Direct Chromosome Representation and Advanced Genetic Operators for Production Scheduling", *Proceedings of the Fifth International Conference on Genetic Algorithms*, University of Illinois at Urbana – Champaign, Urbana, IL, pp. 352–359.

[16] Carlier, J. and E. Pinson (1989), "An Algorithm for Solving the Job–Shop Problem", *Management Science*, Vol. 35, No. 2, pp. 164–176.

[17] Chan, D. Y. (1989), "Design of a Scheduling System for a Flexible Manufacturing Cell", Ph.D. Dissertation, Arizona State University, Tuscon, AZ.

[18] Chandra, J. and J. Talavage (1991), "Intelligent Dispatching for flexible manufacturing", *International Journal of Productions Research*, Vol. 29, No. 11, pp. 2259–2278.

[19] Cheng, T. C. E. and M. C. Gupta (1989), "Survey of scheduling research involving due date determination decisions", *European Journal of Operational Research*, Vol. 38, pp. 156 – 166.

[20] Chryssolouris, G., J. E. Pierce and K. Dicke (1992), "A Decision–Making Approach to the Operation of Flexible Manufacturing Systems", *The International Journal of Flexible Manufacturing Systems*, Vol. 3, No. 4, pp. 309 – 330.

[21] Conway, R. W. (1965a), "Priority Dispatching and Work–In–Process Inventory in a Job Shop", *Journal of Industrial Engineering*, Vol. 16, No. 1, pp. 228.

[22] Conway, R. W. (1965b), "Priority Dispatching and Job Lateness in a Job Shop", *Journal of Industrial Engineering*, Vol.16, pp. 123.

[23] Conway, R. W., W. L. Maxwell and L. W. Miller (1967), *Theory of Scheduling*, Addison–Wesley, Reading, MA.

[24] Dar–El, E. M., and R. A. Wysk (1982), "Job Shop Scheduling – A Systematic Approach", *Journal of Manufacturing Systems*, Vol. 1, No. 1, pp. 77–88.

[25] Davis, L. (1985), "Job Shop Scheduling with Genetic Algorithms", *Proceedings of an International Conference on Genetic Algorithms and Their Applications*, Carnegie–Mellon University, Pittsburgh, PA, pp. 136–140.

[26] Davis, L. (1991), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold Publishers, New York.

[27] Dejong, K. A. (1975), Analysis of the Behavior of a Class of Genetic Adaptive Systems, Ph.D. Dissertation, Department of Computer Sciences, University of Michigan, Ann Arbor, MI.

[28] Dobson, G., U. S. Karmarkar and J. L. Rummel (1987), "Batching to Minimize Flow Times on One Machine", *Management Science*, Vol. 33, No. 6, pp. 779 – 784.

[29] Dutta, A., (1990) "Reacting to Scheduling Exceptions in FMS Environments", *IIE Transactions*, Vol. 22, No. 4, 300 – 314.

[30] ElMaraghy, H. A. (1982), "Simulation and graphical animation of advanced manufacturing systems", *Journal of Manufacturing Systems*, Vol. 1, No. 1, pp. 77–88.

[31] ElMaraghy, H. A. (1993), "Evolution and Future Perspectives of Computer Aided Process Planning (CAPP)", Keynote paper CIRP Annals, Vol. 42/2/1993, pp. 739–751.

[32] ElMaraghy, H. A. and W. H. ElMaraghy (1993), "Bridging the Gap Between Process Planning and Production Planning and Control", *Proceedings of the CIRP Journal of Manufacturing Systems*, Vol. 22, No. 1, pp. 5–11.

[33] Elsayed, E. A. and O. I. Unal (1989), "Order batching algorithms and travel–time estimation for automated storage/retrieval systems", *International Journal of Production Research*, Vol. 27, No. 7, pp. 1097–1114.

[34] Falkenauer, E. and S. Bouffouix (1991), "A Genetic Algorithm for Job Shop", *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, Sacramento, CA., pp. 824 – 829.

[35] Fang, H. L., P. Ross and D. Corne (1993), "A Promising Genetic Algorithm Approach to Job–Shop Scheduling, Rescheduling, and Open–Shop Scheduling Problems", *Proceedings of the Fifth International Conference on Genetic Algorithms*, University of Illinois at Urbana – Champaign, Urbana, IL, pp. 375–382.

[36] Fox, M. S. (1987), *Constraint–Directed Search: A Case Study of Job–Shop Scheduling*, Morgan Kaufmann Publishers, Inc., Los Altos, CA.

[37] Fox, M. S., and S. F. Smith, (1984), "ISIS: A Knowledge–Based System for Factory Scheduling", *Expert Systems*, 1(1), pp. 25–49.

[38] Fox, B. R. and K. G. Kempf (1985), "Complexity, Uncertainty, and Opportunistic Scheduling", *Proceedings: 2nd IEEE Conference on AI Applications*, pp. 487–492.

[39] French, S. (1982), *Sequencing and Scheduling: An Introduction to the Mathematics of the Job–Shop*, John Wiley & Sons Inc., New York.

[40] Garey, M. R. and Johnson, D. S. (1979), *Computers and Intractability*, Freeman and Company, San Francisco, CA.

[41] Goldberg, D. E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison–Wesley Publishing Company Inc., Reading, MA.

[42] Goldberg, D. E., and R. Lingle (1985), "Alleles, Loci and the Traveling Salesman Problem", *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Carnegie–Mellon University, Pittsburgh, PA, pp. 154–159.

[43] Graves, S. C. (1981), "A Review of Production Scheduling", *Operations Research*, Vol. 29, No. 4, pp. 646–675.

[44] Greenberg, H. H. (1966), "A Branch–bound Solution to the General Scheduling Problem", *Management Science*, pp. 353–361.

[45] Grefenstette, J. J. (1986), "Optimization of Control Parameters for Genetic Algorithms", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC–16, No. 1, Jan./Feb., pp. 122–128.

[46] Grefenstette, J. J. (1987), "Incorporating problem specific knowledge into Genetic Algorithms", *Genetic Algorithms and Simulated Annealing*, L. Davis, ed., Morgan Kaufmann Publishers Inc., Los Altos, CA, pp. 42–60.

[47] Grefenstette, J., R. Gopal, B. Rasmaita and D. Van Gucht (1985), "Genetic Algorithms for the Traveling Salesman Problem", *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Carnegie–Mellon University, Pittsburgh, PA, pp. 160–168.

[48] Groover, M. P. (1987), *Automation, Production Systems, and Computer Integrated Manufacturing*, Prentice Hall Inc., Englewood Cliffs, NJ.

[49] Gupta, J. N. (1972), "Heuristic Algorithms for the Multistage Flowshop Scheduling Problem, *AIIE Transactions*, Vol. 4, pp. 11–23.

[50] Gupta, M. C., Y. P. Gupta and G. W. Evans (1993), "Operations planning and scheduling problems in advanced manufacturing systems", *International Journal of Productions Research*, Vol. 31, No. 4, pp. 869 – 900.

[51] Gupta, M. C., Y. P. Gupta and A. Kumar (1993), "Minimizing flow time variance in a single machine system using genetic algorithms", *European Journal of Operational Research*, Vol. 70, pp. 289–303.

[52] Hatzikonstantis, L. and C. B. Besant, (1992), "Job–Shop Scheduling Using Certain Heuristic Search Algorithms", *International Journal of Advanced Manufacturing Technology*, Vol. 7, pp. 251–261.

[53] Hershauer, J. C. and J. Ebert (1974), "Search and Simulation Selection of a Job Shop Scheduling Rule", *Management Science*, Vol. 21, pp. 883–888.

[54] Holland, J. H. (1975), *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI.

[55] Jackson, R. H. F. and A. T. Jones (1987), "An architecture for decision making in the factory of the future", *Interfaces*, Vol. 17, No. 6, pp. 15–28.

[56] Jain, A. K. and H. A. ElMaraghy (1994a), "Manufacturing Scheduling using Genetic Algorithms", Proceedings of the CSME Forum, Montreal, pp.712–727.

[57] Jain, A. K. and H. A. ElMaraghy (1994b), "The Use of Batch Sizing to Improve Flow and Waiting Times in FMS", *Proceedings of the Rensselaer's 4th International Conference on Computer Integrated Manufacturing and Automation Technology*, Troy, NY, pp. 403–408.

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

[59] Karmarkar, U. S. (1987a), "Lot Sizing and Sequencing Delays", *Management Science*, Vol. 33, No. 3, pp. 419–423.

[60] Karmarkar, U. S. (1987b), "Lot Sizes, Lead Times and In–Process Inventories", *Management Science*, Vol. 33, No. 3, pp. 409–418.

[61] Karmarkar, U. S., S. Kekre and S. Kekre (1985), "Lotsizing in Multi–Item Multi–Machine Job Shops", *IIE Transactions*, Vol. 17, No. 3, pp. 290–298.

[62] Karp, R. M. (1975), "On the Computational Complexity of Combinatorial Problems", *Networks*, Vol. 5, pp. 45–68.

[63] King, C. U., S. S. Adams and E. L. Fisher (1988), "Representation of Manufacturing Entities", *Intelligent Manufacturing*, pp. 77–81.

[64] King, J. R. (1976), "The Theory–Practice Gap in Job–Shop Scheduling", The *Production Engineer*, Vol. 55, pp. 137 – 143.

[65] King, J. R. and A. S. Spachis (1980), "Scheduling: Bibliography & Review", *International Journal of Physical Distribution and Materials Management*, Vol. 10, MCB Publications Limited, West Yorkshire, England.

[66] Kusiak, A. (1986), "Efficient Implementation of Johnson's Scheduling Algorithm", *IIE Transactions*, Vol. 18, No. 4, pp. 215–216.

[67] Kusiak, A. and M. Chen (1988), "Expert systems for planning and scheduling manufacturing systems", *European Journal of Operational Research*, Vol. 34, pp. 113–130.

[68] Laarhoven, P. J. M. V., E. H. L. Arts, and J. K. Lenstra (1992), "Job Shop Scheduling by Simulated Annealing" *Operations Research*, Vol. 40, No. 1, pp. 113–125.

[69] Lee, I., R. Sikora and M. J. Shaw (1993), "Joint Lot Sizing and Sequencing with Genetic Algorithms for Scheduling Evolving the Chromosome Structure", *Proceedings of the Fifth International Conference on Genetic Algorithms*, University of Illinois at Urbana–Champaign, Morgan Kaufmann Publishers, San Mateo, CA, pp. 383–389.

[70] Lee, T. (1987), An Integrated Knowledge Based Scheduler for Batch Manufacturing Systems (IKBAS), Ph.D. Dissertation, Cleveland State University, Cleveland, OH.

[71] Li, R., Y. Shyu and S. Adiga (1993) "A heuristic rescheduling algorithm for computer–based production scheduling systems", *International Journal of Productions Research*, Vol. 31, No. 8, pp. 1815–1826.

[72] Liepins, G. E., M. R. Hilliard, M. Palmer and M. Morrow (1987), "Greedy Genetics", *Proceedings of the Second International Conference on Genetic Algorithms*, Cambridge, MA, pp. 90–99.

[73] Lin, G. Y and J. J. Soldberg (1991), "Effectiveness of Flexible Routing Control", *The International Journal of Flexible Manufacturing Systems*, Vol. 3, pp. 189–211.

[74] Logendran, R. and N. Nudtasomboom (1991), "Minimizing the makespan of a group scheduling problem: a new heuristic", *International Journal of Production Economics*, Vol. 22, pp. 217–230.

[75] Maccarthy, B. L. and J. Liu (1993), "Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling", *International Journal of Productions Research*, Vol. 31, No. 1, pp. 59–79.

[76] Matsuura, H., H. Tsubone and M. Kanezashi (1993), "Sequencing, dispatching and switching in a dynamic manufacturing environment", *International Journal of Productions Research*, Vol. 31, No. 7, pp. 1671 – 1688.

[77] McClain, J. O. and L. J. Thomas (1985), *Operations Management: Production of Goods and Services*, Prentice–Hall Publishers, Englewood Cliffs, NJ.

[78] Michalewicz, Z. (1992), *Genetic Algorithms + Data Structures = Evolution Programs*, Springer–Verlag Publishers, New York.

[79] Montazeri, M. and L. N. Van Wassenhove (1990), "Analysis of scheduling rules for an FMS", *International Journal of Productions Research*, Vol. 28, No. 4, pp. 785–802.

[80] Muth, J. F., and G. L. Thompson (1963), *Industrial Scheduling*, Prentice–Hall Publishers, Englewood Cliffs, NJ.

[81] Nakano, R. and T. Yamada (1991), "Conventional Genetic Algorithm for Job Shop", *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Mateo, CA, pp. 4774–4779.

[82] Nasr, N., and E. A. Elsayed (1990), "Job Shop Scheduling with Alternative Machines", *International Journal of Productions Research*, Vol. 28. No. 9, pp. 1595–1609.

[83] Noronha, S. J. and V. V. S. Sarma (1991), "Knowledge–Based Approaches for Scheduling Problems: A Survey", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 3, No. 2, pp. 160–171.

[84] Panwalkar, S. S. and W. Iskander (1977), "A Survey of Scheduling Rules", *Operations Research*, Vol. 25, No. 1, pp. 45–61.

[85] Potts, J. C, T. D. Giddens and S. B. Yadav (1994), "The Development and Evaluation of an Improved Genetic Algorithm Based on Migration and Artificial Selection", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 24, No. 1, pp. 73–86.

[86] Rachamadugu, R. and K. E. Stecke (1993), "Classification and review of FMS scheduling procedures", *Production Planning and Control*, Vol. 5, No. 1.

[87] Rajendran, C. and D. Chaudhuri (1991), "A Flowshop Scheduling Algorithm to Minimize Total Flowtime", *Journal of Operations Research Society of Japan*, Vol. 34, No. 1, pp. 28–46.

[88] Raman, N. (1985), A Survey of the Literature on Production Scheduling as it Pertains to Flexible Manufacturing Systems, Report: University of Michigan #NBS–GCR–85–499.

[89] Ramashesh, R. (1990), "Dynamic Job Shop Scheduling: A Survey of Simulation research", *OMEGA International Journal of Productions Research*, Vol. 18, No. 1, pp. 43–97.

[90] Ravi, T. (1994), Design and Evaluation of Flexible Manufacturing Systems – An Expert System Approach, Ph. D. Thesis, McMaster University, Hamilton, ON.

[91] Rochette, R. and R. P. Sadowski (1976), "A statistical comparison of the performance of simple dispatching rules for a particular set of job shops", *International Journal of Productions Research*, Vol. 14, pp. 63 – 71.

[92] Rodammer, F. A. and K. P. White, Jr. (1988), "A Recent Survey of Production Scheduling", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 18, No. 6, pp. 841–851.

[93] Sabuncuoglu, I. and D. L. Hommertzheim (1993), "Experimental Investigation of an FMS Due–date Scheduling Problem: Evaluation of Machine and AGV Scheduling Rules", *The International Journal of Flexible Manufacturing Systems*, Vol. 5, pp. 301 – 323.

[94] Schaffer, J. D., R. A. Caruana, L. J. Eshelman and R. Das (1989), "A Study of Control Parameters Affecting Online Performance of Genetic Algorithms for Function Optimization", *Proceedings of the Third International Conference on Genetic Algorithms*, J. D. Schaffer, ed., Morgan Kaufmann Publishers Inc., Palo Alto, CA, pp. 51–60.

[95] Sen, T. and Sushil K. Gupta (1984), "A State–of–Art Survey of Static Scheduling Research Involving Due Dates", *OMEGA The International Journal of Management Science*, Vol. 12, No. 1, pp. 63–76.

[96] Sepehri, M., E. A. Silver and C. New (1986), "A Heuristic for Multiple Lot Sizing For an Order Under Variable Yield", *IIE Transactions*, Vol. 18, pp. 63–69.

[97] Smith, M. L., R. Ramesh, R. A. Dudek and E. L. Blair (1986), "Characteristics of US flexible manufacturing systems – A survey", *Proceedings of the Second ORSA/TIMS Conference on Flexible Manufacturing Systems*, K. E. Stecke and R. Suri (eds.), Elsevier Science Publishers, Amsterdam, pp. 477–486.

[98] Smith, S. F. (1991), *Knowledge–Based Production Management: Approaches, Results and Prospects*, CMU Report # CMU–RI–TR–91–21, Carnegie Mellon University, Pittsburgh, PA.

[99] Smith, Stephen F. (1989), *The OPIS framework for Modeling Manufacturing Systems*, CMU Report # CMU–RI–TR–89–30, Carnegie Mellon University, pp. 1–55.

[100] Sridhar, J. and C. Rajendran (1993), "Scheduling in a cellular manufacturing system: a simulated annealing approach", *International Journal of Productions Research*, Vol. 31, No. 12, pp. 2927–2945.

[101] Starkweather, T., S. McDaniel, K. Mathias, D. Whitley and C. Whitley (1991), "A Comparison of Genetic Sequencing Operators", *Proceedings of the Fourth International Conference on Genetic Algorithms*, San Diego, CA, pp. 69–75.

[102] Stecke, K. E. (1986), "Useful Models to Address FMS Operating Problems", *Advances in Production and Operations Management Systems*, E. Szelke and J. Browne eds., Elsevier Science Publishers, North–Holland.

[103] Stecke, K. E. and James J. Solberg (1981), "Loading and control policies for a flexible manufacturing systems", *International Journal of Productions Research*, Vol. 19, No. 5, pp. 481–490.

[104] Stecke, K. E. (1985), "Design, Planning, Scheduling and Control Problems of Flexible Manufacturing Systems", *Annals of Operations Research*, Vol. 3, pp. 3 – 12.

[105] Steffen, M. S. (1986), "A Survey of Artificial Intelligence – Based Scheduling Systems", *Proceedings of 1986 Fall Industrial Engineering Conference*, Institute of Industrial Engineers, Boston, MA, pp. 395–405.

[106] Stranc, C. (1992), Reactive Planning Environment, M. Eng. Thesis, McMaster University, Hamilton, Canada.

[107] UcKun, S., S. Bagchi, K. Kawamura and Y. Miyabe (1993), "Managing Genetic Search in Job Shop Scheduling", *IEEE Expert*, pp. 15–24.

[108] Vancheeswaran, R., and M. A. Townsend (1993), "Two–Stage Heuristic Procedure for Scheduling Job Shops", *Journal of Manufacturing Systems*, Vol. 12, No. 4, pp. 315–325.

[109] Verbraeck, A. (1991), Developing an Adaptive Scheduling Support Environment, Ph.D. Thesis, Delft University, Delft, Netherlands.

[110] White, K. P. Jr. (1990), "Advances in the Theory and Practice of Production Scheduling", *Control and Dynamic Systems*, Vol. 37, pp. 115–157.

[111] Whitley, Darrel (1989), "The GENITOR Algorithm and Selective Pressure: Why Rank–Based Allocation of Reproductive Trials is Best", *Proceedings of the Third International Conference on Genetic Algorithms*, Morgan Kaufmann Publishers Inc., Palo Alto, CA, pp. 116–121.

[112] Whitley, D. and J. Kauth (1988), "GENITOR: a Different Genetic Algorithm", *Proceedings of the Rocky Mountain Conference on Artificial Intelligence*, Denver, CO, pp. 118–130.

[113] Whitley, D., T. Starkweather and D. Shaner (1991), "The Travelling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination", *Handbook of Genetic Algorithms* Lawrence Davis ed., Van Nostrand Reinhold, New York, NY, pp. 350–372.

[114] Wu, S. D. (1987), An Expert System Approach for the Control and Scheduling of Flexible Manufacturing Cells, Ph.D. Thesis, Pennsylvania State University, University Park, PA.

[115] Wu, S. D. and R. A. Wysk (1989), "Multi–pass Expert Control System – A Control/Scheduling Structure for Flexible Manufacturing Cells", *Journal of Manufacturing Systems*, Vol. 7, No. 2, pp. 107–120.

[116] Yamamoto, M. and S. Y. Nof, (1985), "Scheduling/rescheduling in the manufacturing operating system environment", *International Journal of Productions Research*, Vol. 23, No. 4, pp. 705–722.

[117] Yang, T., J. P. Ignizio and D. E. Deal (1989), "An Exchange Heuristic for Generalized Job Shop Scheduling", *Engineering Optimization*, Vol. 15, pp. 83–96.

[118] Zanakis, S. H., J. R. Evans and A. A. Vazacopoulos (1989), "Heuristic methods and applications: A categorized survey", *European Journal of Operational Research*, Vol. 43, pp. 88–110.

APPENDIX A

EXAMPLES USED IN CHAPTER 4

Jobs	Machines 1	2	3
1	5	8	20
2	6	30	6
3	30	4	5
4	2	5	3
5	3	10	4
6	4	1	4

Figure A.1: Example A1 used for parameter setting (Ref. Chan 1989)

Operation Number	Machine (Processing time)					
	Job 1	Job 2	Job 3	Job 4	Job 5	Job 6
1	3 (1)	2 (8)	3 (5)	2 (5)	3 (9)	2 (3)
2	1 (3)	3 (5)	4 (4)	1 (5)	2 (3)	4 (3)
3	2 (6)	5 (10)	6 (8)	3 (5)	5 (5)	6 (9)
4	4 (7)	6 (10)	1 (9)	4 (3)	6 (4)	1 (10)
5	6 (3)	1 (10)	2 (1)	5 (8)	1 (3)	5 (4)
6	5 (6)	4 (4)	5 (7)	6 (9)	4 (1)	3 (1)

Figure A.2: Example A2 used for parameter setting (Ref. Muth and Thompson 1963)

Jobs	Machines 1	Machines 2
1	3	5
2	4	2

Opn. No.	Machine (Processing Time) Job 1	Job 2
1	1 (4)	3 (2)
2	3 (3)	2 (1)
3	2 (6)	1 (5)

Figure A.3: Processing time for Problem P1 (Ref. Greenberg 1968)

Machines	Jobs 1	2	3	4	5	6	7
1	6	2	4	1	7	4	7
2	3	9	3	8	1	5	6

Figure A.4: Processing time for Problem P2 (Ref. Kusiak 1986)

Jobs	Machine (Processing Time) Operations 1	2
1	1(8)	2(2)
2	1(7)	2(5)
3	1(9)	2(8)
4	1(4)	2(7)
5	2(6)	1(4)
6	2(5)	1(3)
7	1(9)	–
8	2(1)	–
9	2(5)	–

Jobs	Machine (Processing Time) Operations 1	2
1	1(8)	2(7)
2	2(12)	1(14)
3	1(13)	2(9)
4	1(6)	–
5	1(7)	2(13)
6	2(9)	1(8)
7	2(12)	–
8	1(4))	2(10)
9	1(10)	2(6)
10	2(10)	1(11)
11	2(6)	–
12	2(8)	–
13	1(5)	2(13)
14	1(7)	–

Figure A.5: Processing time for Problem P3 (Ref. French 1982)

		Machines		
		1	2	3
Jobs	1	20	0	30
	2	0	10	30
	3	40	50	0
	4	10	20	0
	5	10	0	10

Figure A.6: Processing time for Problem P4 (Ref. Sridhar and Rajendran 1993)

See Figure A.1

Figure A.7: Processing time for Problem P5 (Ref. Gupta 1972 Chan 1989)

		Machines			
		1	2	3	4
Jobs	1	22	11	19	21
	2	9	14	16	2
	3	20	19	4	2
	4	10	18	6	7

Figure A.8: Processing time for Problem P6 (Ref. Chan 1989)

		Arrival Time	Machines				Due Date
			1	2	3	4	
Jobs	1	0	5	6	4	5	31
	2	0	4	7	3	5	30
	3	0	9	5	5	3	35
	4	0	6	8	4	1	40

Figure A.9: Processing time for Problem P7 (Ref. Lee 1987)

		Machines			
		i	2	3	4
Jobs	1	7	25	10	12
	2	13	18	21	10
	3	3	25	3	18
	4	11	9	1	6
	5	6	4	25	11

Figure A.10: Processing time for Problem P8 (Ref. Rajendran and Chaudhuri 1991)

		Batch Size	Sequence of Operations	Due Date
			Machine (Processing time)	
Jobs	1	2	1(1) – 2(1) – 3(1) – 2(1) – 4(1) – 1(1) – 3(1)	18
	2	4	3(2) – 1(3) – 4(1) – 2(1)	30
	3	2	4(2) – 1(1) – 3(1) – 2(3)	24
	4	2	2(1) – 1(1) – 4(3) – 3(2) – 1(1)	20
	5	8	2(1) – 3(1) – 4(1)	32

Figure A.11: Processing time for Problem P9 and P10 (Ref. Lee 1987)

		Arrival Time	Sequence of Operations	Due Date
			Machine (Processing time)	
Jobs	1	0	4(2) – 1(1) – 3(1) – 2(3)	9
	2	0	3(2) – 1(3) – 4(1) – 2(1)	7
	3	0	4(1) – 2(1) – 3(1) – 2(1) – 4(1) – 1(1) – 3(1)	10
	4	0	2(1) – 1(1) – 4(3) – 3(2) – 1(1)	8
	5	0	2(1) – 3(1) – 4(1)	5
	6	4	3(2) – 1(3) – 4(1) – 2(1)	12
	7	4	2(1) – 1(1) – 4(3) – 3(2) – 1(1)	12
	8	6	2(1) – 3(1) – 4(1)	10
	9	6	4(2) – 1(1) – 3(1) – 2(3)	13

Figure A12: Processing time for Problem P11 (Ref. Lee 1987)

		Machines				
		1	2	3	4	5
Jobs	1	3	4	6	8	2
	2	1	3	5	7	6
	3	4	2	1	0	5
	4	2	9	8	7	3

Figure A.13: Processing time for Problem P12 (Ref. Chan 1989)

		Machines					
		1	2	3	4	5	6
Jobs	1	12	45	6	10	18	36
	2	61	17	13	9	63	19
	3	1	20	3	17	2	6
	4	23	19	7	1	19	21

Figure A.14: Processing time for Problem P13 (Ref. Chan 1989)

		Machines						
		M1	M2	M3	M4	M5	M6	M7
Jobs	J1	0.137329	89.160156	73.846436	0	89.978027	59.967041	44.525146
	J2	0	0	55.895996	16.958618	0	38.610840	1.849365
	J3	98.123169	54.409790	90.982056	92.672729	20.318604	88.381958	0
	J4	91.812134	57.730103	21.871948	45.455933	71.533203	18.194580	7.397461
	J5	90.670776	22.924805	25.781250	30.807495	87.591553	13.699341	59.811401

Figure A.15: Processing time for Problem P14 (Ref. Logendran et al. 1991))

Jobs	Machines						
	M1	M2	M3	M4	M5	M6	M7
J1	0	0	0	720	0	0	0
J2	0	630	1260	4200	0	210	0
J3	0	75	0	0	0	0	0
J4	0	1080	0	4320	270	6480	1080
J5	0	0	0	0	210	3360	630
J6	0	0	1800	4500	0	0	0
J7	1785	0	0	37230	0	0	0
J8	0	0	0	0	600	0	0

Figure A.16: Processing time for Problem P15 (Ref. Logendran et al. 1991)

See Figure A.2

Figure A.17: Processing time for Problem P16 (Ref. Muth and Thompson 1963)

Opn. No.	Machine (Processing Times)									
	Job1	Job2	Job3	Job4	Job5	Job6	Job7	Job8	Job9	Job10
1	1(29)	1(43)	2(91)	2(81)	3(14)	3(84)	2(46)	3(31)	1(76)	2(85)
2	2(78)	3(90)	1(85)	3(95)	1(6)	2(2)	1(37)	1(86)	2(69)	1(13)
3	3(9)	5(75)	4(39)	1(71)	2(22)	6(52)	4(61)	2(46)	4(76)	3(61)
4	4(36)	10(11)	3(74)	5(99)	6(61)	4(95)	3(13)	6(74)	6(51)	7(7)
5	5(49)	4(69)	9(90)	7(9)	4(26)	9(48)	7(32)	5(32)	3(85)	9(64)
6	6(11)	2(28)	6(10)	9(52)	5(69)	10(72)	6(21)	7(88)	10(11)	10(76)
7	7(62)	7(46)	8(12)	8(85)	9(21)	1(47)	10(32)	9(19)	7(40)	6(47)
8	8(56)	6(46)	7(89)	4(98)	8(49)	7(65)	9(89)	10(48)	8(89)	4(52)
9	9(44)	8(72)	10(45)	10(22)	10(72)	5(6)	8(30)	8(36)	5(26)	5(90)
10	10(21)	9(30)	5(33)	6(43)	7(53)	8(25)	5(55)	4(79)	9(74)	8(45)

Figure A.18: Processing time for Problem P17 (Ref. Muth and Thompson 1963)

APPENDIX B

EXAMPLES USED IN CHAPTER 5

Data set for Problem B1

Job(i)	Operations (j)	Machines (M_k)					
P_i	O_{ij}	M_1	M_2	M_3	M_4	M_5	M_6
P_1	O_{11}	2	3	4	–	–	–
	O_{12}	–	3	–	2	4	–
	O_{13}	1	4	5	–	–	–
P_2	O_{21}	3	–	5	–	2	–
	O_{22}	4	3	–	–	6	–
	O_{23}	–	–	4	–	7	11
P_3	O_{31}	5	6	–	–	–	–
	O_{32}	–	4	–	3	5	–
	O_{33}	–	–	13	–	9	12
P_4	O_{41}	9	–	7	9	–	–
	O_{42}	–	6	–	4	–	5
	O_{43}	1	–	3	–	–	3

Figure B.1: Processing times for Problem B1 (Ref. Nasr and Elsayed 1990)

Data set for Problem B2

Table B.1: Task precedence and order quantity

	Task–1	Task–2	Task–3	Task–4	Task–5	Quantity
Job Type 1	Turn	Review	Drill	Inspection	Wash	20
Job Type 2	Bore	Wash	Screw	Inspection	–	20
Job Type 3	Turn	–	–	–	–	40
Job Type 4	Mill	Wash	Inspection	Drill	Review	20
Job Type 5	Turn	Wash	–	–	–	40
Job Type 6	Mill	Wash	Inspection	Review	–	40
Job Type 7	Face	Wash	Inspection	Drill	Wash	20
Job Type 8	Mill	Review	–	–	–	40
Job Type 9	Drill	Review	Wash	Turn	–	20
Job Type 10	Mill	Wash	Inspection	–	–	40

Table B.2: Machines available in Workcells

	Machining Workcell	Review Workcell	Inspection Workcell	Wash Workcell
Machines	M1, M2, M3, M4, M5	M6, M7	M8	M9
Tasks	Turning, Facing, Drilling, Milling, Boring, Screwing	Review	Inspection	Wash

Table B.3: Processing Time (Setup Time) Information

Job–Task No.	Resources								
	M1	M2	M3	M4	M5	M6	M7	M8	M9
1–1	97(196)	88(176)	95(190)	86(172)	90(180)	–	–	–	–
1–2	–	–	–	–	–	6(12)	5(10)	–	–
1–3	34(68)	34(68)	33(66)	31(62)	32(64)	–	–	–	–
1–4	–	–	–	–	–	–	–	11(22)	–
1–5	–	–	–	–	–	–	–	–	7(14)
2–1	35(70)	36(74)	32(68)	32(64)	36(72)	–	–	–	–
2–2	–	–	–	–	–	–	–	–	10(20)
2–3	65(130)	59(118)	62(124)	60(120)	62(124)	–	–	–	–
2–4	–	–	–	–	–	–	–	15(30)	–
3–1	54(108)	48(96)	48(98)	50(100)	51(102)	–	–	–	–
4–1	41(82)	43(86)	39(78)	43(80)	38(76)	–	–	–	–
4–2	–	–	–	–	–	–	–	–	9(18)
4–3	–	–	–	–	–	–	–	8(16)	–
4–4	86(172)	93(186)	84(168)	84(180)	88(176)	–	–	–	–
4–5	–	–	–	–	–	6(10)	6(12)	–	–
5–1	85(170)	82(164)	89(178)	87(174)	89(178)	–	–	–	–
5–2	–	–	–	–	–	–	–	–	6(12)
6–1	60(120)	66(132)	60(124)	65(130)	66(128)	–	–	–	–
6–2	–	–	–	–	–	–	–	–	7(14)
6–3	–	–	–	–	–	–	–	20(40)	–
6–4	–	–	–	–	–	6(12)	6(10)	–	–
7–1	78(156)	84(168)	80(160)	86(172)	86(164)	–	–	–	–
7–2	–	–	–	–	–	–	–	–	9(18)
7–3	–	–	–	–	–	–	–	19(38)	–
7–4	66(132)	63(124)	59(118)	63(126)	63(122)		–	–	–
7–5	–	–	–	–	–	–	–	–	10(20)
8–1	23(46)	24(44)	24(48)	24(50)	23(47)	–	–	–	–
8–2	–	–	–	–	–	10(18)	10(20)	–	–
9–1	62(124)	62(128)	71(142)	67(134)	69(138)	–	–	–	–
9–2	–	–	–	–	–	8(16)	8(16)	–	–
9–3	–	–	–	–	–	–	–	–	10(20)
9–4	48(96)	44(88)	45(90)	48(94)	44(86)	–	–	–	–
10–1	61(132)	65(130)	60(120)	59(118)	63(126)	–	–	–	–
10–2	–	–	–	–	–	–	–	–	8(16)
10–3	–	–	–	–	–	–	–	12(24)	–

Machine Specifications		
MC#	Operation	Setup Time
M1	Op1	2
M1	Op2	2
M1	Op3	4
M2	Op4	2
M2	Op5	4
M3	Op6	6
M3	Op7	2
M4	Op8	4
M4	Op13	4
M5	Op2	4
M5	Op3	2
M6	Op9	6
M6	Op6	4
M6	Op14	2
M6	Op15	4
M7	Op10	2
M7	Op11	2
M7	Op12	4

Job/Task Specifications			
Job/Task No.	Task Type	Task Duration	Job. Priority
T1,1	Op4	10	4
T1,2	Op3	8	
T1,3	Op13	8	
T2,1	Op4	6	3
T2,2	Op5	10	
T2,3	Op7	10	
T3,1	Op8	10	5
T3,2	Op6	10	
T3,3	Op4	6	
T4,1	Op7	6	2
T4,2	Op1	8	
T4,3	Op2	8	
T4,4	Op13	14	
T5,1	Op9	24	1
T5,2	Op5	10	
T5,3	Op14	8	
T6,1	Op10	8	6
T6,2	Op15	8	
T6,3	Op12	10	
T6,4	Op2	8	
T6,5	Op11	14	

Figure B.2: Data for Problem B2 in Chapter 5 (Ref. Dutta 1990)

APPENDIX C

INPUT/OUTPUT SESSION

Part 1

Input Session with the User Interface

```
**************************************************************************   **
*                                                                            **
*MANUFACTURING SCHEDULING WITH GENETIC ALGORITHMS                            **
*                                                                            **
**************************************************************************   **

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++       +
+                                                                            +
+              GENERAL INFORMATION                                           +
+              ===================                                           +
+                                                                            +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++       +
```

Do you wish to enter new data or modify existing data ?

1. Enter entirely new scheduling data.
2. Modify existing scheduling data.

Enter your choice :-> 1

How many Machines are currently available in the system (1 to 30) :-> 5

==> Number of Machines = 5
IS THIS CORRECT (y or n) ? :->y

How many different Part types (Orders) you want to schedule (1 to 50) :-> 4

==> Number of Parts (Orders) = 4
IS THIS CORRECT (y or n) ? :->y

Enter Maximum number of batches allowed in any order (1 to 10) :-> 1

==> Number of Batches per Order = 1
IS THIS CORRECT (y or n) ? :->y

Enter Minimum number of parts in any sub-batch (4 to 10) :-> 4

==> Minimum allowed quantity per batch = 4
IS THIS CORRECT (y or n) ? :->y

Enter Waiting time limit to be used for truncared SI and SPT dispatching rules (0 to 100) :-> 0

==> waiting Limit is = 0
IS THIS CORRECT (y or n) ? :->y

ORDER SIZE (DEMAND) INFORMATION:
==========

Enter Order Size of Part type 1:-> 4
Enter Order Size of Part type 2:-> 4
Enter Order Size of Part type 3:-> 4
Enter Order Size of Part type 4:-> 4

Order Number :	1	2	3	4
Order Size :	4	4	4	4

Do you wish to modify any value (y or n) ? :-> n

THE FINAL ACCEPTED VALUES FOR ORDER SIZE ARE AS FOLLOWS:

Order Number :	1	2	3	4
Order Size :	4	4	4	4

TASK (OPERATION) INFORMATION:
==========

Enter Number of Tasks in Part type 1:-> 3
Enter Number of Tasks in Part type 2:-> 3
Enter Number of Tasks in Part type 3:-> 3
Enter Number of Tasks in Part type 4:-> 3

Order Number :	1	2	3	4
Number of Tasks :	3	3	3	3

Do you wish to modify any value (y or n) ? :-> n

THE FINAL ACCEPTED VALUES FOR OPERATIONS ARE AS FOLLOWS:

Order Number :	1	2	3	4
Number of Tasks :	3	3	3	3

ALTERNATE MACHINE CHOICE INFORMATION:

Enter number of alternate machines available to process Part 1/Task 1:-> 2
Enter number of alternate machines available to process Part 1/Task 2:-> 2
Enter number of alternate machines available to process Part 1/Task 3:-> 2

Enter number of alternate machines available to process Part 2/Task 1:-> 1
Enter number of alternate machines available to process Part 2/Task 2:-> 2
Enter number of alternate machines available to process Part 2/Task 3:-> 2

Enter number of alternate machines available to process Part 3/Task 1:-> 3
Enter number of alternate machines available to process Part 3/Task 2:-> 2
Enter number of alternate machines available to process Part 3/Task 3:-> 2

Enter number of alternate machines available to process Part 4/Task 1:-> 3
Enter number of alternate machines available to process Part 4/Task 2:-> 2
Enter number of alternate machines available to process Part 4/Task 3:-> 2

Order Number	Task Number	Alternate Choices
1	1	2
1	2	2
1	3	2
2	1	1
2	2	2
2	3	2
3	1	3
3	2	2
3	3	2
4	1	3
4	2	2
4	3	2

Do you wish to modify the values (y or n) ? :-> n

THE FINAL VALUES FOR ALTERNATE CHOICE ARE AS FOLLOWS:

Order Number	Task Number	Machine Choices
1	1	2
1	2	2
1	3	2

2	1	1
2	2	2
2	3	2
3	1	3
3	2	2
3	3	2
4	1	3
4	2	2
4	3	2

TASK – MACHINE REQUIREMENT INFORMATION:

Enter Machine choice 1 on which Part 1/Task 1 is possible:-> 1
Enter Machine choice 2 on which Part 1/Task 1 is possible:-> 3

Enter Machine choice 1 on which Part 1/Task 2 is possible:-> 4
Enter Machine choice 2 on which Part 1/Task 2 is possible:-> 5

Enter Machine choice 1 on which Part 1/Task 3 is possible:-> 1
Enter Machine choice 2 on which Part 1/Task 3 is possible:-> 3

Enter Machine choice 1 on which Part 2/Task 1 is possible:-> 1

Enter Machine choice 1 on which Part 2/Task 2 is possible:-> 2
Enter Machine choice 2 on which Part 2/Task 2 is possible:-> 3

Enter Machine choice 1 on which Part 2/Task 3 is possible:-> 1
Enter Machine choice 2 on which Part 2/Task 3 is possible:-> 3

Enter Machine choice 1 on which Part 3/Task 1 is possible:-> 1
Enter Machine choice 2 on which Part 3/Task 1 is possible:-> 2
Enter Machine choice 3 on which Part 3/Task 1 is possible:-> 5

Enter Machine choice 1 on which Part 3/Task 2 is possible:-> 4
Enter Machine choice 2 on which Part 3/Task 2 is possible:-> 5

Enter Machine choice 1 on which Part 3/Task 3 is possible:-> 1
Enter Machine choice 2 on which Part 3/Task 3 is possible:-> 5

Enter Machine choice 1 on which Part 4/Task 1 is possible:-> 1
Enter Machine choice 2 on which Part 4/Task 1 is possible:-> 3
Enter Machine choice 3 on which Part 4/Task 1 is possible:-> 4

Enter Machine choice 1 on which Part 4/Task 2 is possible:-> 2
Enter Machine choice 2 on which Part 4/Task 2 is possible:-> 4

Enter Machine choice 1 on which Part 4/Task 3 is possible:-> 1
Enter Machine choice 2 on which Part 4/Task 3 is possible:-> 2

Order Number	Task Number	Alternate Choice	Machine Number

1	1	1	1
1	1	2	3
1	2	1	4
1	2	2	5
1	3	1	1
1	3	2	3
2	1	1	1
2	2	1	2
2	2	2	3
2	3	1	1
2	3	2	3
3	1	1	1
3	1	2	2
3	1	3	5
3	2	1	4
3	2	2	5
3	3	1	1
3	3	2	5
4	1	1	1
4	1	2	3
4	1	3	4
4	2	1	2
4	2	2	4
4	3	1	1
4	3	2	2

Do you wish to modify the values ? :-> n

THE FINAL VALUES ARE AS FOLLOWS:

Order Number	Task Number	Alternate Choice	Machine Number
1	1	1	1
1	1	2	3
1	2	1	4
1	2	2	5
1	3	1	1
1	3	2	3
2	1	1	1
2	2	1	2
2	2	2	3
2	3	1	1
2	3	2	3
3	1	1	1
3	1	2	2
3	1	3	5

3	2	1	4
3	2	2	5
3	3	1	1
3	3	2	5
4	1	1	1
4	1	2	3
4	1	3	4
4	2	1	2
4	2	2	4
4	3	1	1
4	3	2	2

TASK – MACHINE – SETUP TIME REQUIREMENT INFORMATION:

Enter Setup time required if Part 1/Task 1 goes to Machine 1 (seconds):–> 10
Enter Setup time required if Part 1/Task 1 goes to Machine 3 (seconds):–> 12

Enter Setup time required if Part 1/Task 2 goes to Machine 4 (seconds):–> 8
Enter Setup time required if Part 1/Task 2 goes to Machine 5 (seconds):–> 12

Enter Setup time required if Part 1/Task 3 goes to Machine 1 (seconds):–> 5
Enter Setup time required if Part 1/Task 3 goes to Machine 3 (seconds):–> 6

Enter Setup time required if Part 2/Task 1 goes to Machine 1 (seconds):–> 8

Enter Setup time required if Part 2/Task 2 goes to Machine 2 (seconds):–> 8
Enter Setup time required if Part 2/Task 2 goes to Machine 3 (seconds):–> 10

Enter Setup time required if Part 2/Task 3 goes to Machine 1 (seconds):–> 7
Enter Setup time required if Part 2/Task 3 goes to Machine 3 (seconds):–> 10

Enter Setup time required if Part 3/Task 1 goes to Machine 1 (seconds):–> 12
Enter Setup time required if Part 3/Task 1 goes to Machine 2 (seconds):–> 12
Enter Setup time required if Part 3/Task 1 goes to Machine 5 (seconds):–> 14

Enter Setup time required if Part 3/Task 2 goes to Machine 4 (seconds):–> 9
Enter Setup time required if Part 3/Task 2 goes to Machine 5 (seconds):–> 7

Enter Setup time required if Part 3/Task 3 goes to Machine 1 (seconds):–> 21
Enter Setup time required if Part 3/Task 3 goes to Machine 5 (seconds):–> 23

Enter Setup time required if Part 4/Task 1 goes to Machine 1 (seconds):–> 20
Enter Setup time required if Part 4/Task 1 goes to Machine 3 (seconds):–> 18
Enter Setup time required if Part 4/Task 1 goes to Machine 4 (seconds):–> 22

Enter Setup time required if Part 4/Task 2 goes to Machine 2 (seconds):–> 9
Enter Setup time required if Part 4/Task 2 goes to Machine 4 (seconds):–> 12

Enter Setup time required if Part 4/Task 3 goes to Machine 1 (seconds):–> 6
Enter Setup time required if Part 4/Task 3 goes to Machine 2 (seconds):–> 8

Order Number	Task Number	Alternate Choice	Machine Number	Setup Time
1	1	1	1	10
1	1	2	3	12
1	2	1	4	8
1	2	2	5	12
1	3	1	1	5
1	3	2	3	6
2	1	1	1	8
2	2	1	2	8
2	2	2	3	10
2	3	1	1	7
2	3	2	3	10
3	1	1	1	12
3	1	2	2	12
3	1	3	5	14
3	2	1	4	9
3	2	2	5	7
3	3	1	1	21
3	3	2	5	23
4	1	1	1	20
4	1	2	3	18
4	1	3	4	22
4	2	1	2	9
4	2	2	4	12
4	3	1	1	6
4	3	2	2	8

Do you wish to modify the values ? :-> n

THE FINAL VALUES ARE AS FOLLOWS:

Order Number	Task Number	Alternate Choice	Machine Number	Setup Time
1	1	1	1	10
1	1	2	3	12
1	2	1	4	8
1	2	2	5	12
1	3	1	1	5
1	3	2	3	6
2	1	1	1	8
2	2	1	2	8
2	2	2	3	10
2	3	1	1	7
2	3	2	3	10
3	1	1	1	12
3	1	2	2	12

3	1	3	5	14
3	2	1	4	9
3	2	2	5	7
3	3	1	1	21
3	3	2	5	23
4	1	1	1	20
4	1	2	3	18
4	1	3	4	22
4	2	1	2	9
4	2	2	4	12
4	3	1	1	6
4	3	2	2	8

TASK – MACHINE – PROCESS TIME REQUIREMENT INFORMATION:

Enter Processing time required if Part 1/Task 1 goes to Machine 1 (seconds):-> 20
Enter Processing time required if Part 1/Task 1 goes to Machine 3 (seconds):-> 12

Enter Processing time required if Part 1/Task 2 goes to Machine 4 (seconds):-> 22
Enter Processing time required if Part 1/Task 2 goes to Machine 5 (seconds):-> 20

Enter Processing time required if Part 1/Task 3 goes to Machine 1 (seconds):-> 15
Enter Processing time required if Part 1/Task 3 goes to Machine 3 (seconds):-> 12

Enter Processing time required if Part 2/Task 1 goes to Machine 1 (seconds):-> 14

Enter Processing time required if Part 2/Task 2 goes to Machine 2 (seconds):-> 22
Enter Processing time required if Part 2/Task 2 goes to Machine 3 (seconds):-> 18

Enter Processing time required if Part 2/Task 3 goes to Machine 1 (seconds):-> 29
Enter Processing time required if Part 2/Task 3 goes to Machine 3 (seconds):-> 30

Enter Processing time required if Part 3/Task 1 goes to Machine 1 (seconds):-> 38
Enter Processing time required if Part 3/Task 1 goes to Machine 2 (seconds):-> 40
Enter Processing time required if Part 3/Task 1 goes to Machine 5 (seconds):-> 36

Enter Processing time required if Part 3/Task 2 goes to Machine 4 (seconds):-> 21
Enter Processing time required if Part 3/Task 2 goes to Machine 5 (seconds):-> 23

Enter Processing time required if Part 3/Task 3 goes to Machine 1 (seconds):-> 41
Enter Processing time required if Part 3/Task 3 goes to Machine 5 (seconds):-> 39

Enter Processing time required if Part 4/Task 1 goes to Machine 1 (seconds):-> 53
Enter Processing time required if Part 4/Task 1 goes to Machine 3 (seconds):-> 52
Enter Processing time required if Part 4/Task 1 goes to Machine 4 (seconds):-> 50

Enter Processing time required if Part 4/Task 2 goes to Machine 2 (seconds):-> 33
Enter Processing time required if Part 4/Task 2 goes to Machine 4 (seconds):-> 28

Enter Processing time required if Part 4/Task 3 goes to Machine 1 (seconds):-> 22
Enter Processing time required if Part 4/Task 3 goes to Machine 2 (seconds):-> 23

Order Number	Task Number	Alternate Choice	Machine Number	Setup Time	Process Time
1	1	1	1	10	20
1	1	2	3	12	12
1	2	1	4	8	22
1	2	2	5	12	20
1	3	1	1	5	15
1	3	2	3	6	12
2	1	1	1	8	14
2	2	1	2	8	22
2	2	2	3	10	18
2	3	1	1	7	29
2	3	2	3	10	30
3	1	1	1	12	38
3	1	2	2	12	40
3	1	3	5	14	36
3	2	1	4	9	21
3	2	2	5	7	23
3	3	1	1	21	41
3	3	2	5	23	39
4	1	1	1	20	53
4	1	2	3	18	52
4	1	3	4	22	50
4	2	1	2	9	33
4	2	2	4	12	28
4	3	1	1	6	22
4	3	2	2	8	23

Do you want to modify the values ? :-> n

THE FINAL VALUES ARE AS FOLLOWS:

Order Number	Task Number	Alternate Choice	Machine Number	Setup Time	Process Time
1	1	1	1	10	20
1	1	2	3	12	12
1	2	1	4	8	22
1	2	2	5	12	20
1	3	1	1	5	15
1	3	2	3	6	12
2	1	1	1	8	14
2	2	1	2	8	22
2	2	2	3	10	18
2	3	1	1	7	29

2	3	2	3	10	30
3	1	1	1	12	38
3	1	2	2	12	40
3	1	3	5	14	36
3	2	1	4	9	21
3	2	2	5	7	23
3	3	1	1	21	41
3	3	2	5	23	39
4	1	1	1	20	53
4	1	2	3	18	52
4	1	3	4	22	50
4	2	1	2	9	33
4	2	2	4	12	28
4	3	1	1	6	22
4	3	2	2	8	23

INPUT SUMMARY

Number of Parts to be scheduled : 4
Number of Machines in the system : 5

Part Number :	1	2	3	4
Demand :	4	4	4	4
# of Tasks :	3	3	3	3

^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^

Setup/Processing time Information

vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

Task	M1	M2	M3	M4	M5
1_1	10/20	0/0	12/12	0/0	0/0
1_2	0/0	0/0	0/0	8/22	12/20
1_3	5/15	0/0	6/12	0/0	0/0
2_1	8/14	0/0	0/0	0/0	0/0
2_2	0/0	8/22	10/18	0/0	0/0
2_3	7/29	0/0	10/30	0/0	0/0
3_1	12/38	12/40	0/0	0/0	14/36
3_2	0/0	0/0	0/0	9/21	7/23
3_3	21/41	0/0	0/0	0/0	23/39
4_1	20/53	0/0	18/52	22/50	0/0
4_2	0/0	9/33	0/0	12/28	0/0
4_3	6/22	8/23	0/0	0/0	0/0

Do you want to update all the related input files (y or n) ?:-> y

All the above information are now written in files as follows:

Data	File name
Task – Machine Requirement	: No_split_option_machine.dat
Task – Setup time Requirement	: No_split_option_setup.dat
Task – Process time Requirement	: No_split_option_time.dat

PART 2

Interaction with the Scheduler

```
**********************************************************************

    MANUFACTURING SCHEDULING WITH GENETIC ALGORITHMS

**********************************************************************

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  +
+                                                                    +
+              GENERAL INFORMATION                                   +
+              -------------------                                   +
+                                                                    +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++  +

              SCHEDULING INPUT FROM PREVIOUS SESSION

                1. Number of Parts (Orders)                 = 10
                2. Number of Machines                       = 9
                3. Minimum number of parts in any sub batch = 4

Part Number :        1   2   3   4   5   6   7   8   9   10
Demand      :       20  20  40  20  40  40  20  40  20  40
# of Tasks  :        5   4   1   5   2   4   5   2   4   3

                ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
                Setup/Processing time Information
                vvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvvv

Task  M1      M2      M3      M4      M5      M6      M7     M8     M9

1_1   194/97  176/88  190/95  172/86  180/90  0/0     0/0    0/0    0/0
1_2   0/0     0/0     0/0     0/0     0/0     12/6    10/5   0/0    0/0
1_3   68/34   68/34   66/33   62/31   64/32   0/0     0/0    0/0    0/0
1_4   0/0     0/0     0/0     0/0     0/0     0/0     0/0    22/11  0/0
1_5   0/0     0/0     0/0     0/0     0/0     0/0     0/0    0/0    14/7

2_1   70/35   74/36   68/32   64/32   72/36   0/0     0/0    0/0    0/0
2_2   0/0     0/0     0/0     0/0     0/0     0/0     0/0    0/0    20/10
2_3   130/65  118/59  124/62  120/60  124/62  0/0     0/0    0/0    0/0
2_4   0/0     0/0     0/0     0/0     0/0     0/0     0/0    30/15  0/0

3_1   108/54  96/48   98/48   100/50  102/51  0/0     0/0    0/0    0/0
```

```
4_1   82/41  86/43  78/39   80/43   76/38  0/0   0/0   0/0   0/0
4_2   0/0    0/0    0/0     0/0     0/0    0/0   0/0   0/0   18/9
4_3   0/0    0/0    0/0     0/0     0/0    0/0   0/0   16/8  0/0
4_4   172/86 186/93 168/84  180/84  176/88 0/0   0/0   0/0   0/0
4_5   0/0    0/0    0/0     0/0     0/0    10/6  12/6  0/0   0/0

5_1   170/85 164/82 178/89  174/87  178/89 0/0   0/0   0/0   0/0
5_2   0/0    0/0    0/0     0/0     0/0    0/0   0/0   0/0   12/6

6_1   120/60 132/66 124/60  130/65  128/66 0/0   0/0   0/0   0/0
6_2   0/0    0/0    0/0     0/0     0/0    0/0   0/0   0/0   14/7
6_3   0/0    0/0    0/0     0/0     0/0    0/0   0/0   40/20 0/0
6_4   0/0    0/0    0/0     0/0     0/0    12/6  10/6  0/0   0/0

7_1   156/78 168/84 160/80  172/86  164/86 0/0   0/0   0/0   0/0
7_2   0/0    0/0    0/0     0/0     0/0    0/0   0/0   0/0   18/9
7_3   0/0    0/0    0/0     0/0     0/0    0/0   0/0   38/19 0/0
7_4   132/66 124/63 118/59  126/63  122/63 0/0   0/0   0/0   0/0
7_5   0/0    0/0    0/0     0/0     0/0    0/0   0/0   0/0   20/10

8_1   46/23  44/24  48/24   50/24   47/23  0/0   0/0   0/0   0/0
8_2   0/0    0/0    0/0     0/0     0/0    18/10 20/10 0/0   0/0

9_1   124/62 128/62 142/71  134/67  138/69 0/0   0/0   0/0   0/0
9_2   0/0    0/0    0/0     0/0     0/0    16/8  16/8  0/0   0/0
9_3   0/0    0/0    0/0     0/0     0/0    0/0   0/0   0/0   20/10
9_4   96/48  88/44  90/45   94/48   86/44  0/0   0/0   0/0   0/0

10_1  122/61 130/65 120/60  118/59  126/63 0/0   0/0   0/0   0/0
10_2  0/0    0/0    0/0     0/0     0/0    0/0   0/0   0/0   16/8
10_3  0/0    0/0    0/0     0/0     0/0    0/0   0/0   24/12 0/0
```

Enter name of the project :-> For_dissertation
Project : For_dissertation

Enter version number of the project :-> Final
Project version: Final

Enter your name :-> Ajay Jain
Analyst : Ajay Jain

The output will be stored in RESULT-For_dissertation-Final file.

Minimum allowed quantity per batch specified is = 4, Is this correct (y or n) :-> y

Enter Due Date Tightness (DDT) factor to be used (1 to 8) :-> 1.5

DDT = 1.500000

++

```
+                                                                    +
+            SCHEDULING PARAMETERS                                   +
+            ---------------------                                   +
+                                                                    +
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

The choices for the different Performance Measures are as follows :
1. Minimize the Mean flow time
2. Minimize the Maximum flow time
3. Minimize the Mean tardiness
4. Minimize the Maximum tardiness
5. Minimize the Mean waiting time
6. Minimize the Maximum waiting time
7. Maximize the Average machine utilization

Enter Primary Performance Criteria to be used (? for help) :-> ?

****

HELP ON PERFORMANCE MEASURES

The different performance criterion you can select are as follows:

1. Minimize the Mean flow time:
   Flow time is defined to be the time that part spends in the shop. Mean flow time is the flow time average of all the jobs present in the shop.

2. Minimize the Maximum flow time:
   This performance criteria minimizes the maximum time that a part spends in the shop. This is also called the makespan.

3. Minimize the Mean tardiness:
   The positive difference between the completion time of a part and its due date is called tardiness. This criteria minimizes the tardiness of all the jobs.

4. Minimize the Maximum Tardiness:
   This performance criteria minimizes the maximum of all the tardy times.

5. Minimize the Mean waiting time:
   This criteria minimizes the time a job has to wait for before the machine. This is averaged over all the jobs to minimize the mean waiting time.

6. Minimize the Maximum waiting time:
   This performance criteria minimizes the individual maximum waiting time of all the jobs.

7. Maximize the Average machine utilization:
   This criteria maximizes the average utilization of all the resources available in the system.

You can select upto two performance measures (Primary performance & Secondary performance and the software will give you the schedule that best satisfies both performance measures.

****

Enter Primary Performance Criteria to be used (? for help) :-> 1

Performance Criteria selected is Mean flow time.

Do you wish to use Secondary Performance Criteria ? (y or n) :-> y
1. Minimize the Mean flow time
2. Minimize the Maximum flow time
3. Minimize the Mean tardiness
4. Minimize the Maximum tardiness
5. Minimize the Mean waiting time
6. Minimize the Maximum waiting time
7. Maximize the Average machine utilization

Enter Secondary Performance Criteria to be used (? for help) :-> 3

Secondary Performance Criteria selected is Mean tardiness.

You have two choices as follows:

1. Specify your own Scheduling Parameters.
2. Search the knowledge base for Scheduling Parameters.

Enter your Choice (? for help) :-> ?

****

HELP SESSION

Option 1: You can specify your own scheduling parameters such as batching policy, job regeneration methods, and scheduling rules (SI, SPT, etc.).

Option 2: You can search the knowledge base for the best parameter combination that should be used in the case of desired performance criteria.

****

Enter your Choice (? for help) :-> 1

You have opted to specify your own Scheduling Parameters.

The different Batch Splitting policies are:
1. None (None)
2. Split the batches in equal sizes (Equal)
3. Split the batches in half sizes (Half)
4. Split the batches according to the number of machines in the system (Machines)
5. Split the batches according to the number of operations of jobs (Operations)
6. Split the batches according to the processing time of jobs (Time)

Enter the Batching policy to be used (? for help) :-> ?

****

HELP ON BATCHING POLICIES

The different batch sizing policies and as follows:

1. None :
   In this case, the original orders are not splitted into sub orders or sub batches.
   The scheduling is performed on the basis of given order sizes.

2. Split the batches into equal sizes :
   In this case, all the order are divided into sub batches such that there are equal
   number of parts in each sub batch.

3. Split the batches in half sizes:
   In this case, all the orders are divided into half the size of original order.

4. Split according to machine requirement:
   In this case, the batches are formed based on individual part machine requirements.

5. Split according to total number of operations per job:
   In this case, the batches are formed based on the total number of operations of each part.

6. Split according to the processing time:
   In this case, the batches are formed based on the individual parts total processing time.

****

Enter the Batching policy to be used (? for help) :-> 5

Batch Splitting policy selected is Operations.

The different job Regeneration Methods are:
1. Static regeneration
2. Dynamic Constant regeneration
3. Dynamic Continuous regeneration

Enter the Regeneration Method to be used (? for help) :-> ?

****

HELP ON REGENERATION METHODS

Three different job (order) regeneration method that can be used are:

1. Static Regeneration:
   In static regeneration the jobs or orders are released into the shop floor after all the job
   orders in the previous set are completed.

2. Dynamic Constant Regeneration:
   In this case, a set of new job orders are released after every period (for ex. a shift).

3. Dynamic Continuous Regeneration:
   In this case, a new job order is released after completion of the previous order.

****

Enter the Regeneration Method to be used (? for help) :-> 3

Regeneration Method selected is Dynamic Continuous.

The different Dispatching Rules are :

1. Shortest Imminent processing time	(SI)
2. Shortest Processing Time	(SPT)
3. Longest Imminent processing time	(LI)
4. Longest Processing Time	(LPT)
5. Earliest Due Date	(EDD)
6. earliest OPeratioN Due Date	(OPNDD)
7. Least Slack Time per job	(LST)
8. Least Slack time per Operation	(LSO)
9. Slack per Remaining Processing time	(S/RP)
10. Slack per Remaining number of Operations	(S/RO)
11. Shortest Remaining Processing Time	(SRPT)
12. First Come First Serve	(FCFS)

Enter the Dispatching Rule to be used (? for rules) :-> ?

****

HELP ON DISPATCHING RULES

Definition: A dispatching rule is used to select the next job to be processed from a set of jobs waiting in machine queue for service. These are also known as scheduling rule, priority rule or heuristics. Thw tweleve dispatching rules used in this software and their are as follows:

1. Shotest Imminent Processing Time (SI): (or SIPT)
   Select the job with the 'Shortest Imminent operation time'.

2. Shortest Processing Time (SPT):
   Select the job with least total operation time.

3. Longest Imminent Processing Time (LI):
   Select the job with the 'Longest Imminent operation time'.

4. Longest Processing Time (LPT);
   Select the job with most total operation time.

5. Earliest Due Date (EDD):
   Select the job with the earliest due date.

6. Earliest Operation Due Date (OPNDD):
   Select the operation from the queue with the earliest operation due date.

7. Least Slack Time per Job (LST):
   Select the job with the least amount of slack (available time before due date time for remaining operations.

8. Least Slack Time per Operation (LSO):
   Select the job with the least ratio of slack time to the number of total operations.

9. Slack per Remaining Processing Time (S/RP):
   Select the job with the least ratio of slack time to the remaining processing time.

10. Slack per Remaining number of Operations (S/RO):
    Select the job with the least ratio of slack time to the number of remaining operations.

11. Shortest Remaining Processing Time (SRPT):
    Select the job with the least remaining processing time.

12. First Come First Serve (FCFS):
    Select the job that arrived first in the queue.

13. Truncated Shortest Imminent Processing Time (T–SI):
    Select job based on SI rule. If there is job whose waiting time is more than a specified value, use FCFS rule.

14. Truncated Shortest Processing Time (T–SPT):
    Select job based on SPT rule. If there is job whose waiting time is more than a specified value, use FCFS rule.

****

Enter the Dispatching Rule to be used (? for rules) :-> 5

Dispatching Rule selected is Earliest Due Date.

The optimization will be performed by using following selected parameters:
1. Batch Splitting policy is (5) = Operations
2. Regeneration Method is (3) = Dynamic Continuous
3. Dispatching Rule is (5) = Earliest Due Date

++
+ +
+ GENETIC ALGORITHMS PARAMETERS +
+ ========== +
+ +
++

Enter Population Size to be used in Genetic Algorithms (50 to 200) :-> 80

PoolSize = 80
Length_String = 140
Enter Number of Generations or Trials to be used in GA (10 to 10000) :-> 50

Trials = 50
Enter Selection Bias to be used in GA (1.0 to 2.0) :-> 1.8

Bias = 1.800000
Enter Mutation Rate to be used in GA (0.0 to 1.0) :-> 0.1

MutateRate = 0.100000
Enter Interval for the Status Report :-> 10

StatusInterval = 10
Enter name of the FinalPoolFile :-> out thesis disser.out

++

SUMMARY : DATA INPUT FOR SCHEDULING

SCHEDULING PARAMETERS :

Number of Orders to be scheduled	= 10
Number of Machines in the System	= 9
Due date tightness (DDT) factor used	= 1.500000
Waiting time limit in any queue	= 30

Orders	1	2	3	4	5	6	7	8	9	10
Number of Tasks	5	4	1	5	2	4	5	2	4	3
Order Size	20	20	40	20	40	40	20	40	20	40
Order Due date	3100	2520	2160	3180	3800	3960	3800	1360	2740	3400

Primary Performance Criteria	= Mean flow time
Secondary Performance Criteria	= Mean tardiness
Batch Splitting Policy	= Operations
Job Regeneration Method	= Dynamic Continuous
Dispatching Rule	= Earliest Due Date

BASED ON THE ABOVE INFORMATION, THE SUB-BATCH SIZES ARE AS FOLLOWS:

Order	Lot Sizes
1	8 8 4 0
2	10 10 0 0
3	40 0 0 0
4	8 8 4 0
5	20 20 0 0

6	10 10 10 10
7	8 8 4 0
8	20 20 0 0
9	10 10 0 0
10	13 13 14 0

GENETIC ALGORITHM PARAMETERS :

Population Size	= 80
Length of the String	= 140
Number of Generations	= 50
Selection Bias	= 1.800000
Mutation Rate	= 0.100000
Status Interval	= 10
Final Population File	= disser.out

PART 3

System Output

++

(An Integrated Scheduling Approach for Discrete Manufacturing Systems)

RESULT SUMMARY

Analyst : Ajay
Project : For_thesis
Version : Final
Date : Thursday, 24 November 1994

Output File : RESULT-For_thesis-Final

++

INPUT DATA FOR SCHEDULING

SCHEDULING PARAMETERS :

Number of Orders to be scheduled = 10
Number of Machines in the System = 9
Due date tightness (DDT) factor used = 1.500000
Waiting time limit in any queue = 30

Orders	1	2	3	4	5	6	7	8	9	10
Number of Operations	5	4	1	5	2	4	5	2	4	3
Order Size	20	20	40	20	40	40	20	40	20	40
Order Due date	3100	2520	2160	3180	3800	3960	3800	1360	2740	3400

Primary Performance Criteria = Mean flow time
Secondary Performance Criteria = Mean tardiness
Batch Splitting Policy = Operations
Job Regeneration Method = Dynamic Continuous
Dispatching Rule = Earliest Due Date

++

LOT SIZES :

Order	Lot Sizes			
1	8	8	4	0
2	10	10	0	0
3	40	0	0	0
4	8	8	4	0
5	20	20	0	0
6	10	10	10	10
7	8	8	4	0
8	20	20	0	0
9	10	10	0	0
10	13	13	14	0

++

GENETIC ALGORITHM PARAMETERS :

Population Size = 80
Length of the String = 140
Number of Generations = 50
Selection Bias = 1.800000
Mutation Rate = 0.100000
Status Interval = 10
Final Population File = the_final.out

++

TASK'S SETUP TIME/PROCESSING TIME:

Task	M1	M2	M3	M4	M5	M6	M7	M8	M9
1_1	194/97	176/88	190/95	172/86	180/90	0/0	0/0	0/0	0/0
1_2	0/0	0/0	0/0	0/0	0/0	12/6	10/5	0/0	0/0
1_3	68/34	68/34	66/33	62/31	64/32	0/0	0/0	0/0	0/0
1_4	0/0	0/0	0/0	0/0	0/0	0/0	0/0	22/11	0/0
1_5	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	14/7
2_1	70/35	74/36	68/32	64/32	72/36	0/0	0/0	0/0	0/0
2_2	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	20/10
2_3	130/65	118/59	124/62	120/60	124/62	0/0	0/0	0/0	0/0
2_4	0/0	0/0	0/0	0/0	0/0	0/0	0/0	30/15	0/0
3_1	108/54	96/48	98/48	100/50	102/51	0/0	0/0	0/0	0/0
4_1	82/41	86/43	78/39	80/43	76/38	0/0	0/0	0/0	0/0
4_2	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	18/9

4_3	0/0	0/0	0/0	0/0	0/0	0/0	0/0	16/8	0/0
4_4	172/86	186/93	168/84	180/84	176/88	0/0	0/0	0/0	0/0
4_5	0/0	0/0	0/0	0/0	0/0	10/6	12/6	0/0	0/0
5_1	170/85	164/82	178/89	174/87	178/89	0/0	0/0	0/0	0/0
5_2	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	12/6
6_1	120/60	132/66	124/60	130/65	128/66	0/0	0/0	0/0	0/0
6_2	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	14/7
6_3	0/0	0/0	0/0	0/0	0/0	0/0	0/0	40/20	0/0
6_4	0/0	0/0	0/0	0/0	0/0	12/6	10/6	0/0	0/0
7_1	156/78	168/84	160/80	172/86	164/86	0/0	0/0	0/0	0/0
7_2	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	18/9
7_3	0/0	0/0	0/0	0/0	0/0	0/0	0/0	38/19	0/0
7_4	132/66	124/63	118/59	126/63	122/63	0/0	0/0	0/0	0/0
7_5	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	20/10
8_1	46/23	44/24	48/24	50/24	47/23	0/0	0/0	0/0	0/0
8_2	0/0	0/0	0/0	0/0	0/0	18/10	20/10	0/0	0/0
9_1	124/62	128/62	142/71	134/67	138/69	0/0	0/0	0/0	0/0
9_2	0/0	0/0	0/0	0/0	0/0	16/8	16/8	0/0	0/0
9_3	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	20/10
9_4	96/48	88/44	90/45	94/48	86/44	0/0	0/0	0/0	0/0
10_1	122/61	130/65	120/60	118/59	126/63	0/0	0/0	0/0	0/0
10_2	0/0	0/0	0/0	0/0	0/0	0/0	0/0	0/0	16/8
10_3	0/0	0/0	0/0	0/0	0/0	0/0	0/0	24/12	0/0

++

PERFORMANCE MEASURES FOR BEST 2 SCHEDULES:

Schedule Number	Mean Flowtime	Max Flowtime	Mean Tardiness	Max Tardiness	Mean Waittime	Max Waittime	Average Machine Util.
1	2575.43	8336.00	311.83	2606.00	438.62	3408.00	3849.00
2	2578.07	9582.00	336.83	4794.00	432.48	4040.00	3865.22

INDIVIDUAL MACHINE UTILIZATIONS FOR BEST 2 SCHEDULES:

Schedule Number	Machine1 Util.	Machine2 Util.	Machine3 Util.	Machine4 Util.	Machine5 Util.	Machine6 Util.	Machine7 Util.	Machine8 Util.	Machine9 Util.
1	6068.00	4529.00	7274.00	5749.00	4581.00	770.00	452.00	2860.00	2358.00
2	5950.00	8712.00	4470.00	5486.00	3739.00	710.00	502.00	2860.00	2358.00

++

Primary Performance Criteria weightage = 0.900000

Secondary Performance Criteria weightage = 0.100000

Schedule Number	Primary Performance	Secondary Performance	Average Performance
1	2575.43	311.83	2349.06
2	2578.07	336.83	2353.95

++

THE BEST SCHEDULE IS : 1

1. Performance Measure Values for schedule 1 are:

Minimum Mean Flow Time is : 2575.42505
Minimum Maximum Flow Time is : 8336.00000
Minimum Mean Tardiness is : 311.82501
Minimum Maximum Tardiness is : 2606.00000
Minimum Mean Waiting Time is : 438.62500
Minimum Maximum Waiting Time is : 3408.00000
Maximum Mean Machine Utilization is : 0.46173

2. Machine Utilizations for schedule 1 are:

Machine Number	Busy Time	Percent Utilization
1	6068	0.72793
2	4529	0.54331
3	7274	0.87260
4	5749	0.68966
5	4581	0.54954
6	770	0.09237
7	452	0.05422
8	2860	0.34309
9	2358	0.28287

++

FOR THE CURRENT SCHEDULE :

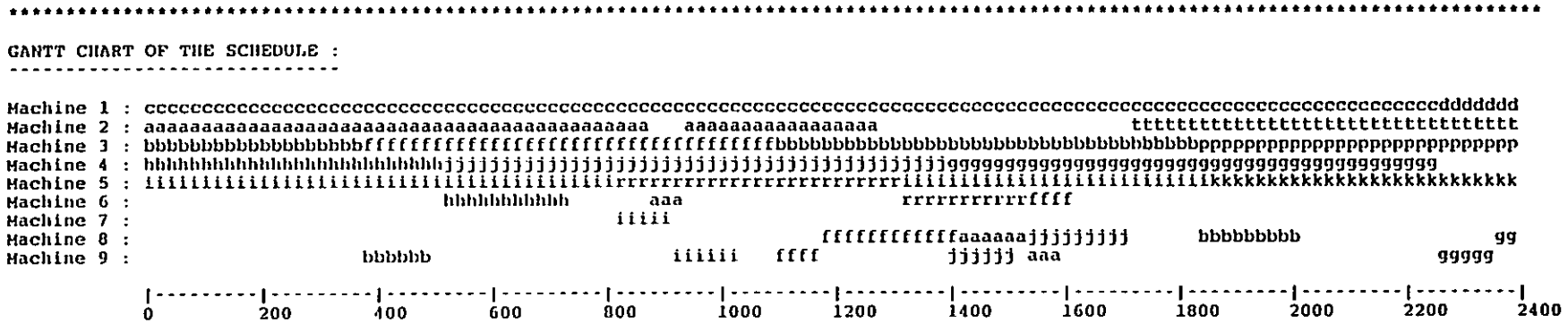
Task	Order Name	Machine	Start Time	Finish Time	Processing Time	NewStart Time	NewFinish Time
1/1	a	2	0	880	880	0	44
1/2	a	6	880	940	60	44	47
1/3	a	2	940	1280	340	47	64

1/4	a	8	1436	1546	110	71	77
1/5	a	9	1546	1616	70	77	80
2/1	b	3	0	308	308	0	19
2/2	b	9	308	508	120	19	25
2/3	b	3	1112	1856	744	55	92
2/4	b	8	1856	2036	180	92	101
3/1	c	1	0	2268	2268	0	113
4/1	d	1	2260	2670	410	113	133
4/2	d	9	2678	2768	90	133	138
4/3	d	8	2904	2984	80	145	149
4/4	d	5	3193	4073	880	159	203
4/5	d	7	4073	4133	60	203	206
5/1	e	1	2670	4540	1870	133	227
5/2	e	9	4548	4680	132	227	234
6/1	f	3	308	1112	724	19	55
6/2	f	9	1112	1196	84	55	59
6/3	f	8	1196	1436	240	59	71
6/4	f	6	1553	1625	72	77	81
7/1	g	4	1415	2275	860	70	113
7/2	g	9	2275	2365	90	113	118
7/3	g	8	2365	2555	190	118	127
7/4	g	1	4548	5208	660	227	260
7/5	g	9	5208	5308	100	260	265
8/1	h	4	0	530	530	0	26
8/2	h	6	530	748	218	26	37
9/1	i	5	0	828	828	0	41
9/2	i	7	828	924	96	41	46
9/3	i	9	924	1044	120	46	52
9/4	i	5	1335	1861	526	66	93
10/1	j	4	530	1415	885	26	70
10/2	j	9	1415	1535	120	70	76
10/3	j	8	1546	1726	180	77	86
11/1	k	5	1861	2761	900	93	138
11/2	k	6	2761	2821	60	138	141
11/3	k	3	2821	3151	330	141	157
11/4	k	8	3164	3274	110	158	163
11/5	k	9	3313	3383	70	165	169
12/1	l	5	2761	3193	432	138	159
12/2	l	9	3193	3313	120	159	165
12/3	l	3	4124	4868	744	206	243
12/4	l	8	4868	5048	180	243	252
13/1	m	0	0	0	0	0	0
14/1	n	2	4193	4623	430	209	231
14/2	n	9	4680	4770	90	234	238
14/3	n	8	4770	4850	80	238	242
14/4	n	1	5208	6068	860	260	303
14/5	n	7	6068	6128	60	303	306
15/1	o	4	4983	6897	1914	249	344
15/2	o	9	6897	7029	132	344	351
16/1	p	3	1856	2580	724	92	129

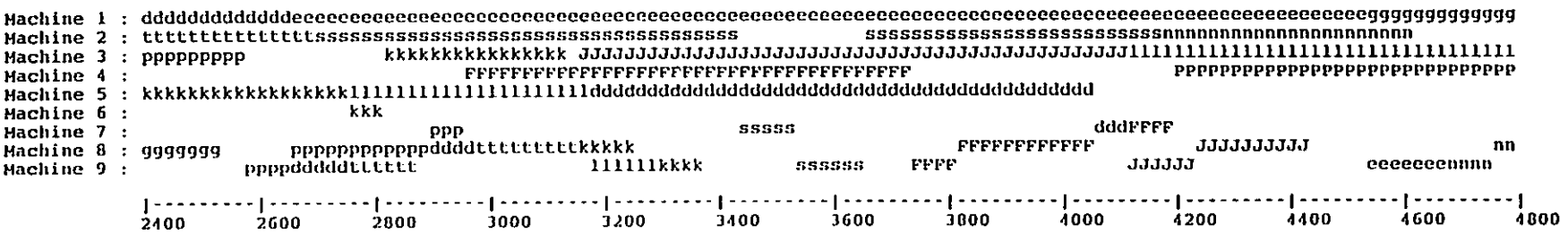
16/2	p	9	2580	2664	84	129	133
16/3	p	8	2664	2904	240	133	145
16/4	p	7	2904	2974	70	145	148
17/1	q	3	5438	6238	800	271	311
17/2	q	9	6238	6328	90	311	316
17/3	q	8	6328	6518	190	316	325
17/4	q	2	6518	7146	628	325	357
17/5	q	9	7146	7246	100	357	362
18/1	r	5	828	1335	507	41	66
18/2	r	6	1335	1553	218	66	77
19/1	s	2	2701	3449	748	135	172
19/2	s	7	3449	3545	96	172	177
19/3	s	9	3545	3665	120	177	183
19/4	s	2	3665	4193	528	183	209
20/1	t	2	1726	2701	975	86	135
20/2	t	9	2760	2888	128	138	144
20/3	t	8	2984	3164	180	149	158
21/1	A	3	4868	5438	570	243	271
21/2	A	6	5438	5474	36	271	273
21/3	A	3	6238	6436	198	311	321
21/4	A	8	6518	6584	66	325	329
21/5	A	9	6584	6626	42	329	331
22/1	B	0	0	0	0	0	0
22/2	B	0	0	0	0	0	0
22/3	B	0	0	0	0	0	0
22/4	B	0	0	0	0	0	0
23/1	C	0	0	0	0	0	0
24/1	D	3	6436	6670	234	321	333
24/2	D	9	6670	6724	54	333	336
24/3	D	8	6724	6772	48	336	338
24/4	D	3	6772	7276	504	338	363
24/5	D	6	7276	7310	34	363	365
25/1	E	0	0	0	0	0	0
25/2	E	0	0	0	0	0	0
26/1	F	4	2974	3754	780	148	187
26/2	F	9	3754	3838	84	187	191
26/3	F	8	3838	4078	240	191	203
26/4	F	7	4133	4203	70	206	210
27/1	G	5	7246	7754	508	362	387
27/2	G	9	7754	7808	54	387	390
27/3	G	8	7808	7922	114	390	396
27/4	G	3	7922	8276	354	396	413
27/5	G	9	8276	8336	60	413	416
28/1	H	0	0	0	0	0	0
28/2	H	0	0	0	0	0	0
29/1	I	0	0	0	0	0	0
29/2	I	0	0	0	0	0	0
29/3	I	0	0	0	0	0	0
29/4	I	0	0	0	0	0	0
30/1	J	3	3164	4124	960	158	206
30/2	J	9	4124	4252	128	206	212

30/3	J	8	4252	4444	192	212	222
31/1	K	0	0	0	0	0	0
31/2	K	0	0	0	0	0	0
31/3	K	0	0	0	0	0	0
31/4	K	0	0	0	0	0	0
31/5	K	0	0	0	0	0	0
32/1	L	0	0	0	0	0	0
32/2	L	0	0	0	0	0	0
32/3	L	0	0	0	0	0	0
32/4	L	0	0	0	0	0	0
33/1	M	0	0	0	0	0	0
34/1	N	0	0	0	0	0	0
34/2	N	0	0	0	0	0	0
34/3	N	0	0	0	0	0	0
34/4	N	0	0	0	0	0	0
34/5	N	0	0	0	0	0	0
35/1	O	0	0	0	0	0	0
35/2	O	0	0	0	0	0	0
36/1	P	4	4203	4983	780	210	249
36/2	P	9	4983	5067	84	249	253
36/3	P	8	5067	5307	240	253	265
36/4	P	6	5307	5379	72	265	268
37/1	Q	0	0	0	0	0	0
37/2	Q	0	0	0	0	0	0
37/3	Q	0	0	0	0	0	0
37/4	Q	0	0	0	0	0	0
37/5	Q	0	0	0	0	0	0
38/1	R	0	0	0	0	0	0
38/2	R	0	0	0	0	0	0
39/1	S	0	0	0	0	0	0
39/2	S	0	0	0	0	0	0
39/3	S	0	0	0	0	0	0
39/4	S	0	0	0	0	0	0
40/1	T	0	0	0	0	0	0
40/2	T	0	0	0	0	0	0
40/3	T	0	0	0	0	0	0

GANTT CHART OF THE SCHEDULE :



GANTT CHART CONTD... :



```
**********************************************************************************************************************************

GANTT CHART CONTD... :
----------------------

Machine 1 : gggggggggggggggggggnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
Machine 2 :                                                                                                    qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
Machine 3 : lllAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqAAAAAAAAAAADDDDDDDDDDDDD      DDDDDDDDDDDDDDDDDDDDDDD
Machine 4 : PPPPPPPPPoooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
Machine 5 :
Machine 6 :                            PPP    AA
Machine 7 :                                                                  nnn
Machine 8 : nn lllllllll PPPPPPPPPPPP                                                                qqqqqqqqqAAAA       DD
Machine 9 :           PPPP       ggggg                                                        qqqqq              AA  DDD         ooooooo       qqq

            |---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
            4800      5000      5200      5400      5600      5800      6000      6200      6400      6600      6800      7000      7200


**********************************************************************************************************************************

GANTT CHART CONTD... :
----------------------

Machine 1 :
Machine 2 :
Machine 3 : DDD                                    GGGGGGGGGGGGGGGGG
Machine 4 :
Machine 5 :   GGGGGGGGGGGGGGGGGGGGGGGGG
Machine 6 :    DD
Machine 7 :
Machine 8 :                                  GGGGGG
Machine 9 : qq                           GGG                          GGG

            |---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
            4800      5000      5200      5400      5600      5800      6000      6200      6400      6600      6800      7000      7200

**********************************************************************************************************************************
```