# Scheduling of Manufacturing Systems Based on Extreme Value Theory and Genetic Algorithms

by

Velusamy Subramaniam

B.Eng. (Mech.) (Hons.), National University of Singapore (1988)

M.Eng., National University of Singapore (1991)

S.M., Massachusetts Institute of Technology (1992)

Submitted to the Department of
Mechanical Engineering
in Partial Fulfillment of the Requirements
for the Degree of

Doctor of Philosophy

at the

Massachusetts Institute of Technology

August 1995



Signature of Author _____
Department of Mechanical Engineering
August 11, 1995

Certified by _____
Professor George Chryssolouris
Thesis Supervisor

Accepted by _____
Professor Ain A. Sonin
Chairman, Department Graduate Committee

**Scheduling of Manufacturing Systems Based on Extreme Value Theory and Genetic Algorithms**

by

Velusamy Subramaniam

Submitted to the Department of Mechanical Engineering on August 11, 1995
in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

## Abstract

Two scheduling methods based on Extreme Value Theory (SEVAT) and Genetic Algorithms (GA) are developed. The SEVAT approach is a schedule building approach that creates a statistical profile of schedules through random sampling and predicts the 'potential' of a schedule alternative. The GA approach, on the other hand, is a schedule permutation approach, in which a population of schedules are initially generated, and through some operations, good traits of these schedules are combined to produce better schedules.

These two scheduling methods were applied to two static benchmark job shop problems (the Muth and Thompson 6x6 and 10x10 problems). The results compare favorably with the optimal solutions, and the solutions obtained using some common dispatch rules and a scheduling approach based on Fuzzy Logic, which is representative of current scheduling research.

A dynamic scheduling problem was designed to reflect a real job shop scheduling environment closely. Two performance measures, viz, Mean Job Tardiness and Mean Job Cost, were used to demonstrate multiple criteria scheduling. Three factors were identified, and varied between two levels each, thereby spanning a varied job shop environment. A factorial design of experiments comprising of 8 experiments were then designed. The SEVAT and GA approaches were applied to these 8 experiments and the results compared with several common dispatching rules and the Fuzzy Logic Approach. The results of this extensive simulation study, overwhelmingly indicate that the SEVAT and GA scheduling approaches produce better scheduling performance than the other methods.

Thesis Supervisor: Professor George Chryssolouris
Title: Associate Professor of Mechanical Engineering

# Dedication

I dedicate this thesis to the

four instrumental pillars

of my life

- Matha
- Pitha
- Guru
- Deivam

# Acknowledgements

I would like to express my heartfelt gratitude to the National University of Singapore, without whose kind and generous support, this thesis will not have been possible.

I am extremely indebted to my advisor, Professor George Chryssolouris for his constant and meticulous supervision of my research. His insatiable desire for creative solutions to the field of manufacturing has most definitely left a permanent mark on me. He has always been available whenever I needed him and most of all, I would like to thank him for being an inspiration. I would also like to thank Professors Patrick Winston and Stephen Graves, members of my thesis committee, for being so patient with me and for providing me with invaluable advice and guidance in my research.

I would also like to thank my 'buddies' in the laboratory for tolerating me and making me feel at home. I had enjoyed many fruitful discussions with them and have gained valuable insights, a privilege accorded by this company of great minds, which only MIT can bring together. Thank you Mike Domroese, Moshin Lee, Paul Sheng, Nick Anastasia, Kristian Dicke, Nick Nassuphis, Andrew Yablon, Andrew Heitner, Alvin Ramsey, Hui Nam, Robert Kallenberg, Don Lee, Jeff Connors, Helen Azrin, Tina Chen, Sean Arnold, Scott Tang, Eric Hopkins, Mike Benco, Joe Lee, Joe Melvin, Joe Lettieri, Rajan Anandan, Adam Chen, Calvin Ying, Jim Derksen, Matthias Siebler, Jeff Stovall, Leo Hsu, Manny Voumvourakis, Tim Fox, Darbie Hailes, Natalie Henry, Reinoud Hermans, German, Ethan Close, John Jacobs, Kendra Borgmann, Jennifer Gilden and my German friends, Thomas Franz, Henning Hannibuth, Nils Tonshoff, Rochus Gold, Dirk Landgrebe, Jens Fassmer, Marcus Mey, Markus Ruwe, Katja Helms, Stefan Leibelt, Johannes Wais, Wolfram Weber, Ulrich Herman, Christian Lischke and Milan Nedeljkovic.

I would like to express my sincere appreciation to Ang Boon Seong and Sung Kah Kay, my fellow countrymen and housemates, for their friendship and moral support which had made life at MIT enjoyable. My special thanks to Donna Jean Kaiser for helping me with my doctoral thesis presentation and for her constant and tireless encouragement, and above all, for being a delightful and wonderful friend.

Last but not least, I would like to thank my family, back in Singapore, for their love, support and encouragement, without which, my academic progress at MIT will have been unfocused.

# Table of Contents

# List of Figures

# Chapter 1
# Introduction

The operation of a manufacturing system is the complex task of planning the material and information flows in the system. Proper material flow is what enables a manufacturing system to produce products on time and in sufficient quantity. It is a direct consequence of the system's information flows: *command* information from human planners or from the planning software prescribes the material flow in the system, while *sensory* information about the status of the system's resources is used to decide on the appropriate commands. The fundamental activity in the operation of a manufacturing system is thus to determine the commands that prescribe the material flow in the system [Chryssolouris, 1992].

Command information flows can be organized into a bi-level hierarchy. High level commands dictate the flow of materials into the manufacturing system. Therefore they determine the system's workload - namely, how many of each type of part to manufacture in each time period, which are typically long, in the order of days or weeks. For this reason, the decision-making activity of generating high level commands is called long-term planning [Chryssolouris, 1992].

Low-level commands dictate the flow of materials within and out of a manufacturing system. They determine which production resources are to be assigned to each operation on each part, and when each operation is to take place. The role of lower-level commands is to resolve contention for the resources of a manufacturing system. This contention occurs among the parts released into the system as a consequence of long term planning decisions. Because low-level commands control individual operations, they must be generated much more frequently than high-level commands. The time between commands is typically on the order of seconds or minutes. For this reason, the decision making activity of generating lower-level commands is called short-term dispatching and is concerned with the detailed assignment of operations to production resources [Chryssolouris, 1992].

## 1.1 The Problem

The practice of scheduling in industry has been dominated by the use of dispatch rules. These rules are easy to apply and are quick in generating schedules. However, there are some shortcomings to using dispatch rules. First, dispatch rules do not consider all of the available resources at the same time. The common practice is to employ an additional rule to first select a resource and then apply the dispatch rule. This contaminates the effectiveness of the dispatch rule and the cumulative effect of these two types of rules (resource selection and dispatch rules) can only be determined through extensive simulation. Second, dispatch rules do not allow for the use of multiple criteria in the scheduling process. Dispatch rules often influence only one performance measure, whereas scheduling in industry usually requires the meeting of several objectives simultaneously. Third, the rigid structure of dispatch rules excludes the use of other useful information that may be available for scheduling. Fourth, there is no single universal dispatch rule. The literature reports hundreds of such rules, and the choice of a suitable dispatch rule depends on the nature of the scheduling problem and the performance measure of interest. Because the scheduling environment itself is dynamic, the nature of the scheduling problem will change over time and therefore, the dispatch rule will also need to change over time.

Academic research in production scheduling has been based primarily on operations research techniques (more specifically, mathematical programming). Recently, many research efforts have employed artificial intelligence techniques, especially rule-based systems. Unfortunately, the use of academic results in industry has been minimal; many academic approaches make assumptions that do not reflect reality. Actual manufacturing environments are extremely variable, and usually cannot be rigidly classified into one of the classical scheduling models often assumed in academic research. In perhaps no other field is the dichotomy between academic research and actual practice more pronounced [Chryssolouris, 1992].

## 1.2 Characterization of the Scheduling Problem

The academic literature contains an abundance of theoretical work on a number of classical scheduling problems. The entities to be scheduled are typically referred to as jobs. Usually, each job corresponds to an individual part and consists of one or more production

operations. Classical scheduling problems can be categorized on the basis of the following dimensions [Graves, 1981].

- Requirements generation
- Processing complexity
- Scheduling criteria

The scheduling problem in most production environments is stochastic and dynamic; however, most models for scheduling problems are deterministic and static [Graves, 1981].

Another perspective in the characterization of the manufacturing scheduling problem is presented in [Parunak, 1991]. Scheduling in a manufacturing environment is viewed as cartesian product of three sets; viz, *'What'* tasks are to be done, *'When'* are these tasks to be done, and *'Where'* (resources) are these tasks to be done.

## 1.3 Research Contribution

The contribution of this thesis to the field of scheduling in manufacturing systems is to bridge the wide gap between academia and industry. This thesis develops scheduling methods to overcome the shortcomings associated with dispatch rules and avoid making the simplistic assumptions associated with academic approaches. The scheduling methods developed will be for the job shop scheduling problem because it is the most difficult scheduling problem amongst manufacturing systems. The methods and tools that are developed and evaluated in this research have the following characteristics:

- *Modeling Ease*
  The theoretical principles underlying each of these approaches are relatively simple and would be very easy to model for a particular manufacturing scheduling problem.

- *Encompassing many problems*
  Each of the methods and tools investigated in this research should also be applicable to most manufacturing problems, be they as varied as Product Design, Production Planning or Process Control.

- *Quality versus Speed*
  The methods and tools should allow the tradeoff between decision quality and computational speed. Academicians have been overly concerned about the accuracy of their methods in predicting the optimum solution and have not placed any constraints on the computational time. On the other hand, in the industry, the accuracy of the solution may not be as crucial. What is crucial, is finding a solution to a problem in some specified time.

Two methods are investigated in this research, and they are based on

- Extreme Value Theory
- Genetic Algorithms

## 1.4 Job Shop Scheduling

The formal job shop problem is defined as [Rodammer and White, 1988]

"N jobs are to be processed on M machines. Each job consists of a set of M operations, one operation uniquely associated with each of the M machines. The processing time for an operation cannot be split. Technological constraints demand that the operations within each job must be processed in a unique order. The scheduling problem involves determining the sequence and timing of each operation on each machine such that some given performance criterion is maximized or minimized."

In this thesis, the formal job shop scheduling problem will be restated, so that it is more general and reflects the actual job shops in industry. Particularly, the constraint that there be M operations for each job (where M is also the number of machines available) will be reformulated. Jobs will be assumed to have some random number of operations each. Furthermore, each operation will not be constrained to have only one available machine to process it. This thesis will explore the job shop where multiple machines could process an operation. Details of the job shop scheduling problem investigated in this thesis will be discussed in the following chapters.

In [Mckay et al, 1988], a survey of job shop schedulers is reported. This survey identifies several common themes that had prevailed, viz,

- Scheduling uses information that is often incomplete, ambiguous, biased, outdated or erroneous.
- The job shop is seldom stable for longer than half an hour.
- Schedulers have various tools at their disposal to deal with variability and constraints, e.g., they can change both the long and short term capacity, alter the long and short term processing logic etc.
- Schedulers use intuition, including sensory data, such as sight, sound and touch, to "fill in the blanks" about what is happening on the shop floor.
- Many constraints or issues can affect the scheduling of different parts of the job shop at different times for different reasons.
- Schedulers use simple dispatching rules, like SPT (Shortest Processing Time) or FCFS (First Come First Served).

The relationship between a job shop and other forms of manufacturing systems is discussed in [MacCarthy and Liu, 1993] and is summarized below in Figure 1.1. In a job shop, each job has its own individual flow pattern or specific route through the machines. In a flow shop, each job has an identical flow pattern, whilst in an open shop, there is no specified flow pattern for any job. In a flow shop, if the order of processing of the jobs are the same, then that system is referred to as the permutation flow shop. When all the machines are identical and viewed as a single stage processing facility, then that manufacturing system is referred to as a parallel machine. Finally, when there is only one machine available, the system is refered to as the single machine (shop).

## 1.5 Static vs Dynamic Scheduling

In Chapter 2, a literature survey of the current job shop scheduling approaches is presented. A common feature that appears in this survey, is that most of the approaches are designed to work with static scheduling problems. In this section, the distinctions between static and dynamic scheduling problems will be explored.

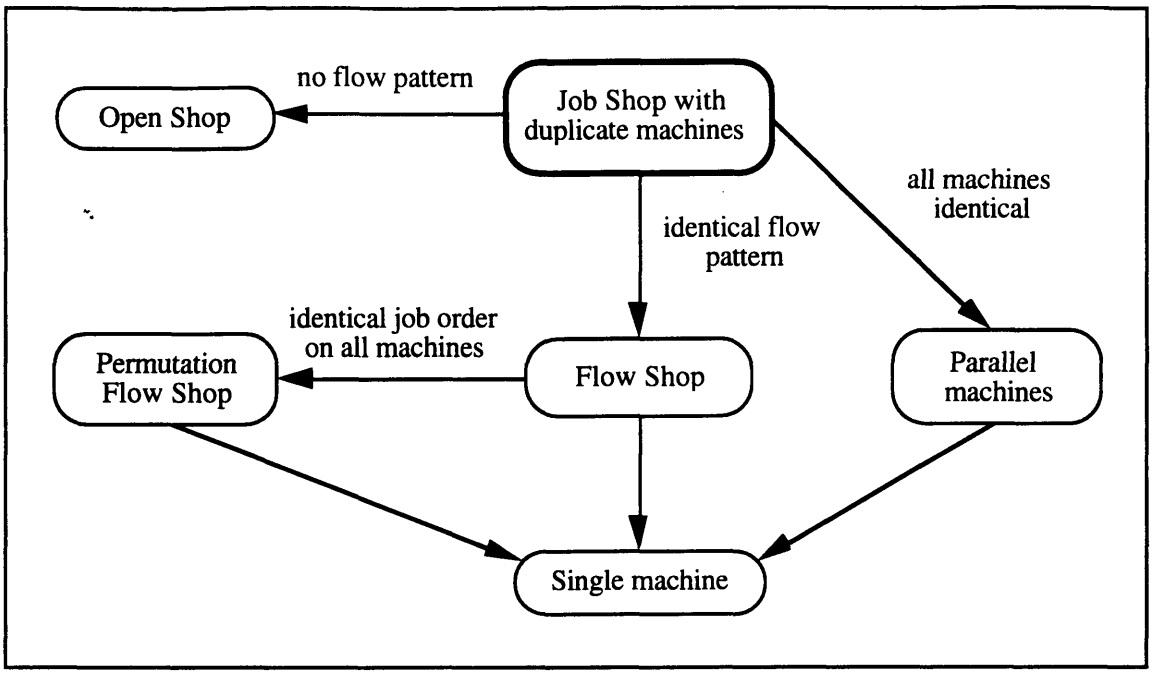


Figure 1.1: Relationships Between Different Machine Environments [MacCarthy and Liu, 1993].

In most static problems, the job shop scheduling problem is one where all the jobs arrive at time zero, i.e., all the jobs that need to be scheduled are already available at the outset of the scheduling problem. The problem also assumes that there are no machine breakdowns, and no operations are pre-empted (i.e. once an operation is started, it may not be interrupted). The processing times of all the operations of all jobs are known a priori and the set up times for the various operations are assumed included in the processing times.

A variation of the static job shop problem accommodates arrival of jobs at non-zero time. However, the arrival times are deterministic and are known a priori. Scheduling decisions take into account the future arrivals.

The dynamic scheduling problem, on the other hand, tend to reflect more of the actual job shop conditions. In this problem, the job arrivals are not known a priori. Scheduling decisions cannot make use of future arrival information and are further complicated by the breakdown and the repair of resources (machines). The arrivals of jobs and, the breakdown and repair of resources are assumed to be described by statistical distributions. These details will be discussed in greater detail in Chapter 5. Figure 1.2, shows the distinctions between static and dynamic job shop scheduling problems.

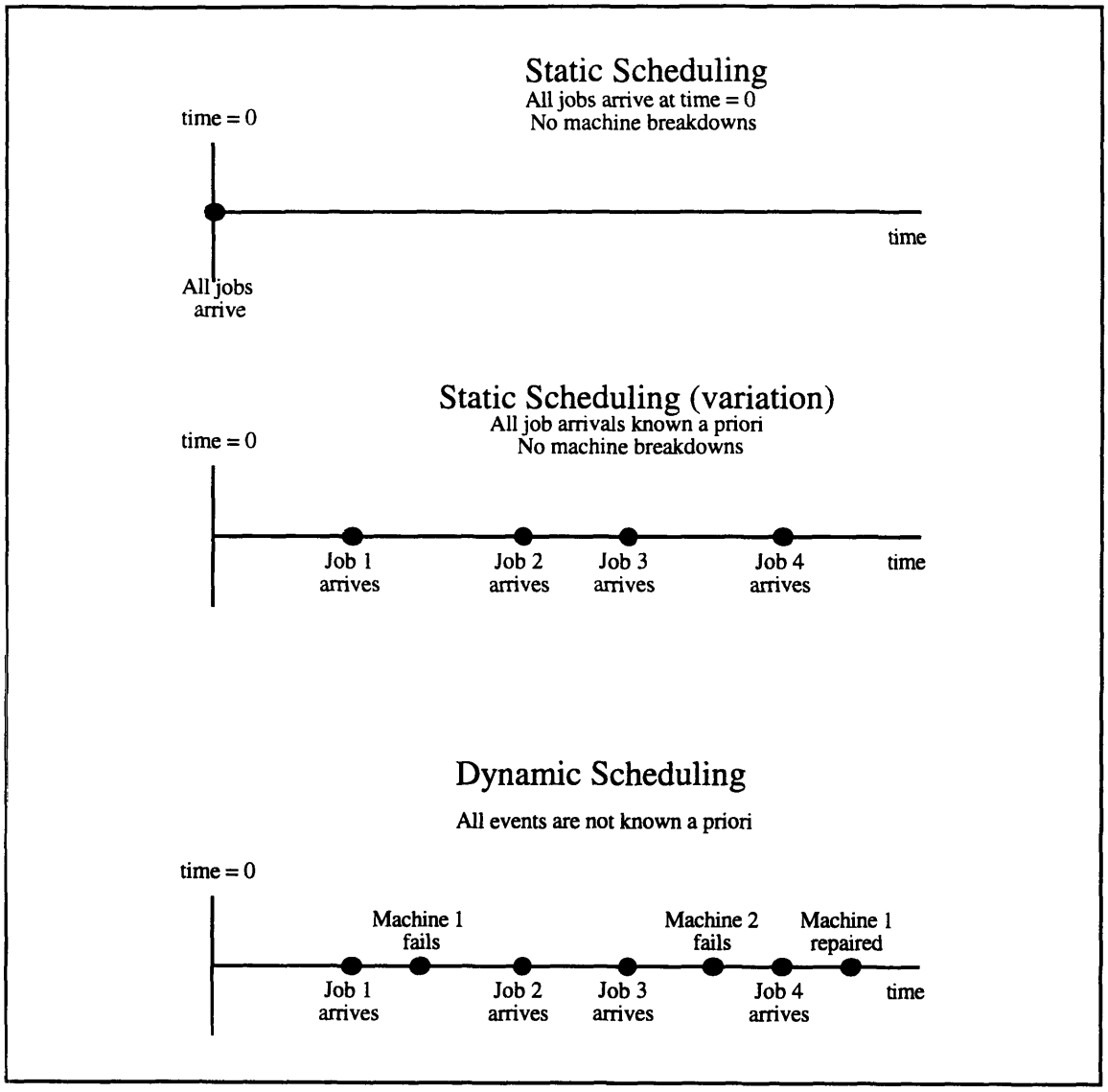


Figure 1.2: Distinctions between Static and Dynamic Job Shop Scheduling

## 1.6 Scope of Thesis

This thesis began with a brief introduction into the manufacturing scheduling problem and discussed the gap that exists between academia and industry. The objectives of this thesis were then stated as an attempt to bridge this gap, by developing scheduling techniques and applying them to the job shop scheduling problem. In the next chapter, a survey of the current job shop scheduling techniques in the last five years will be presented. This survey is not exhaustive and the categorization of the scheduling methods

is not necessarily unique. Following the survey, the research framework will be laid out. A suitable methodology from the literature will be identified to serve as a basis of comparison of the performance of the scheduling methods developed in this thesis. In Chapters 3 and 4, the Extreme Value Theory and the Genetic Algorithms scheduling approaches will be discussed in detail and applied to a static problem. The performances of these approaches will be analyzed and compared. In Chapter 5, a dynamic scheduling problem will be designed and the various scheduling approaches will be applied to this problem and their performances will be discussed. This thesis will then conclude with a summary of the significant results.

# Chapter 2
# Current Job Shop Scheduling Approaches

This chapter presents a survey of the current job shop scheduling approaches that have been reported in the literature in the last five years. These approaches have been organized broadly into five categories (Figure 2.1). Following the survey, a research framework is formulated. A suitable job shop scheduling methodology from the survey is selected and used as a basis of comparison for the performance of the two scheduling methods developed in this thesis.

## 2.1 Heuristic Rules

The most common approach to job shop scheduling in industry is to use dispatching rules. In the literature, the terms priority rules, scheduling rules and heuristic rules are used synonymously for dispatching rules. These terms are used to determine the ranking of the order in which the jobs waiting in machine queues are to be processed when the machines become available. [Ramasesh, 1990] makes a distinction between the terms 'priority rules', 'scheduling rules' and 'heuristics'. A priority rule represents the technique by which a number is assigned to each job in the queue; a heuristic represents rules of thumb; and scheduling rules represent a combination of one or more priority rules and heuristics [Ramasesh, 1990]. In this chapter, the term heuristics or dispatching rules will be used synonymously and the above distinction can be drawn from the context of the scheduling problem.

Dispatching rules were first proposed in the 1950s and were attractive in terms of their simplicity. However, with the advance in computer technology, these simple rules have been modified or combined to make use of other available information from the job shop floor. This has continued to be the current focus of scheduling research in the area of

heuristic rules. The broad classification of 'heuristic rules' can be further categorized (Figure 2.2).

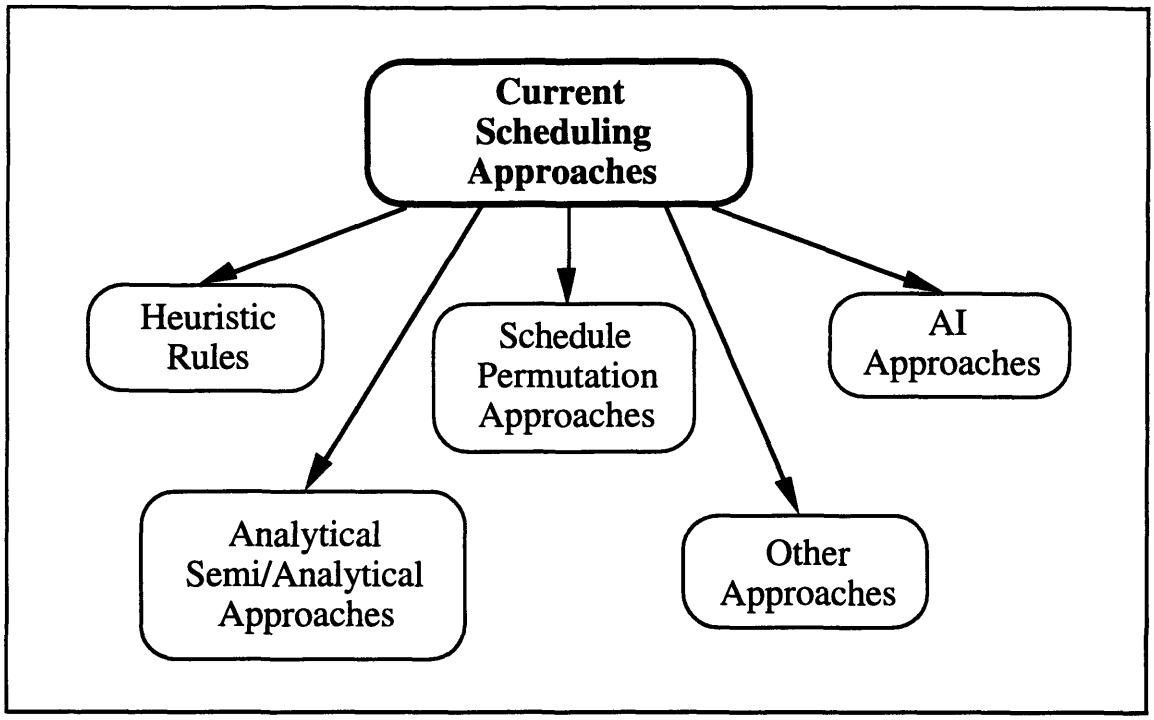


Figure 2.1: Characterization of Current Scheduling Approaches

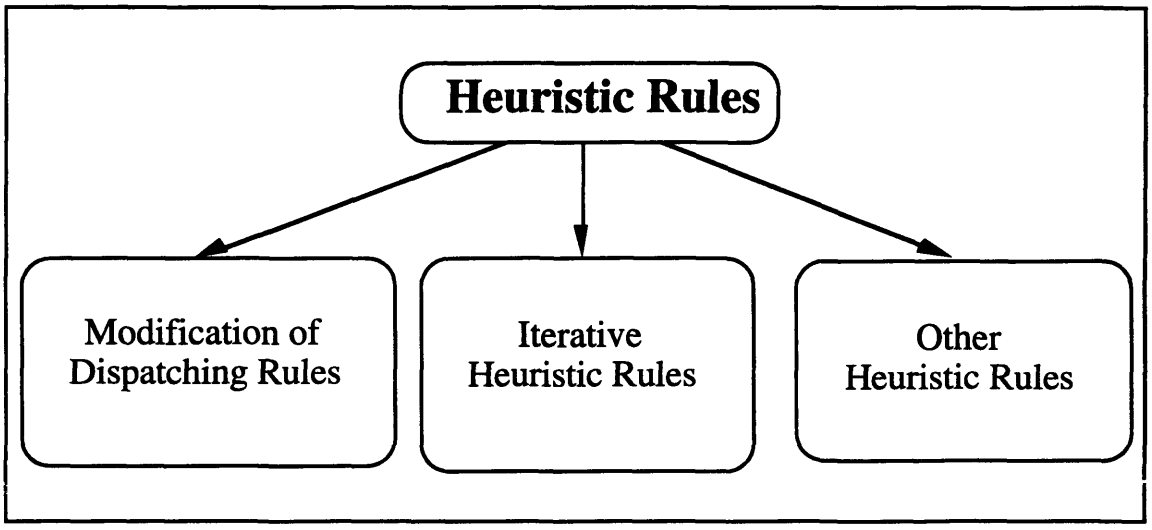


Figure 2.2: Categorization of Heuristic Rules

Dispatching rules can also be classified based on whether the priority value changes over time or not. A dispatching rule is referred to as a dynamic rule if the priority value calculated at a particular instant differs from the value calculated at a later time, e.g. the least slack rule. If the priority value once calculated remains the same throughout, it is referred to as a static rule, e.g. the earliest due date (EDD) rule [Raghu and Rajendran, 1993].

Another dimension of categorizing dispatching rules, is the type of information that is required to calculate the priority of jobs. Local rules require information about jobs currently waiting for the machine under consideration, whereas global rules make use of information about other machines in the shop floor in addition to the machine under consideration [Raghu and Rajendran, 1993].

### 2.1.1 Modification of Dispatching Rules

This section discusses the research attempts to modify existing dispatching rules to enhance performance. The modification attempts can be categorized as:

- combination of dispatching rules.
- expediting and truncating dispatching rules.

Good performance of individual dispatching rules is limited to certain shop load conditions. For example, the earliest due date (EDD), minimum slack time (MST) and the slack per remaining work (S/RPT) rules perform reasonably well with light load levels but deteriorate in congested job shops, whereas SPT performs well in congested shops with tight due dates but fails with low load levels and loose due dates. The failure of simple rules to be effective over a wide range of due date tightness and shop utilization, have been recognized for a long time, and has motivated researchers to combine two or more of these simple dispatching rules into a single rule in order to harness their individually excellent performance characteristics [Anderson and Nyirenda, 1990].

Two combination dispatching rules for minimizing the tardiness (positive difference between the completion time and due date of a job) in a job shop are presented in [Anderson and Nyirenda, 1990]. Both rules are closely related to the modified due date (MOD) rule. The first rule is a combination of the shortest processing time (SPT) rule and

the critical ratio (CR) rule, and the second is a combination of the the SPT rule and the slack per remaining work (S/RPT) rule. These rules do not require any parameter estimation. These composite rules have been shown to work well over other dispatching rules and over various job shop conditions. However, these rules are only effective when minimizing the tardiness performance measure.

In [Benton, 1993], a composite rule called the modified value added composite rule (VMOD) is proposed. This rule is defined as the ratio of the value added to a job to the total value of the job. This work is implemented for a specific purpose, namely to solve scheduling problems with cost and time based performance measures. Despite its attractiveness as a composite rule, this approach will suffer from all the shortcomings of dispatch rules discussed earlier in Chapter 1.

A dynamic dispatching rule is proposed in [Raghu and Rajendran, 1993], where the processing time and due date components of a job are weighted as a function of shop utilization level. This dynamic rule is attractive, because it takes into account the shop floor conditions when dispatching jobs to machines. However, this rule was designed for flow time and tardiness performance measures, it will not be effective in situations where there are performance measures that are not related to flowtime or tardiness. Furthermore, one of the implicit assumptions of this approach is that it does not consider jobs with alternative machine routings.

In [Kannan and Ghosh, 1993], the interaction between truncation schemes and dispatching rules is studied. Truncation is a term that is applied to reducing the set of jobs in a machine queue to a set of (fewer) priority jobs on which the dispatching rules are then applied. For example, the truncation rule, SPTT, truncates jobs that have been waiting in the machine queue for a time $\alpha_1$ and puts these jobs in a priority queue. A certain amount of truncation is beneficial, but too much or too little truncation can compromise performance, depending on the performance measure.

[Ye and Williams, 1991] discuss a SPT-related dispatching rule, referred to as the CEXSPT rule. The CEXSPT rule partitions the jobs waiting at a queue into three queues. All the jobs that are late are placed in the first queue. All the jobs that are not late, but with the next operation being late, is placed in the second queue. In the last queue, all the jobs that are ahead of schedule are placed. The selection of which job to process next is as follows: The job with the smallest processing time in the first queue is processed next,

unless doing so creates at least one new late job, in which case, the job with the smallest processing time in the second queue goes next, provided it does not create a new operationally late job. If it does, the job with the smallest processing time in the third queue is processed next.

The CEXSPT rule was modified to control the scheduling of jobs with long processing times, and its performance is reported to be better than the pure SPT or other due-date based rules. This is due to the fact that late jobs are given priority over jobs that are behind schedule, and jobs that are behind schedule are always given priority over jobs that are ahead of schedule [Ye and Williams, 1991]. However, it is only after the jobs with the long processing times become late or operationally late that these jobs have a chance to be paced through the job shop. There is no guarantee that these jobs can be completed on time [Ye and Williams, 1991].

### 2.1.2 Iterative Heuristic Rules

This section examines heuristic rules that adopt an iterative scheme for job shop scheduling. These approaches may be classified as:

- multiple-pass heuristic rules
- look-ahead heuristic rules
- rescheduling heuristic rules.

In the previous section on the modification of dispatching rules, the rules were all single-pass, i.e., jobs in a machine queue are prioritized according to some criteria and the rule with the highest priority is scheduled. In multiple pass heuristics, two or more passes are used. The first pass is usually one in which a preliminary schedule is created using a dispatching rule and in the second pass, this preliminary schedule is then altered to ensure schedule improvement.

One such approach is proposed in [Vancheeswaran and Townsend, 1993], which consists of two independent stages consisting of a heuristic and followed by a non-enumerative schedule improvement method. This approach was designed to minimize the makespan (total production time) of the schedule. The first stage, consists of a priority rule, termed the modified urgency criterion and is applied to all jobs that will be worked on

the machine of interest, including those that have not reached the machine queue. The modified urgency criterion is defined as the ratio of the total remaining processing time of a job (inclusive of the processing time of the present operation) to the psuedo processing time (PPT) of the job. The PPT is defined as the sum of the processing time of the present operation and the time required to complete work on previous machines.

The second stage of this approach uses the critical job perturbation procedure (CJPP) to improve on the schedule. The critical job is defined as the job that finishes last in the shop and therefore determines the makespan. The CJPP reviews the queues of all the operations of the critical job. If an operation of the critical job was prioritized late in one of the machine queues, this operation is then exchanged with an operation of a non-critical job to improve on the schedule. During the perturbation, the schedule performance may or may not improve or a new critical job may arise. If the schedule performance does not improve, then the previous schedule is kept and a different perturbation is considered on a different machine queue. If the schedule performance is improved and a new critical job arises, then this new job is replaced with the previous critical job and the CJPP is once again applied. The CJPP is terminated when all of the operations on the critical job have the highest priority or if an iteration limit is exceeded [Vancheeswaran and Townsend, 1993].

The drawback of this approach is that it is designed for the minimum makespan performance measure only. Neither the modified urgency criterion, nor the CJPP will work for multi-criteria scheduling problem.

Another multiple-pass heuristic algorithm is proposed in [He et al, 1993], where the performance measure is total job tardiness. This heuristic consists of two phases; in phase 1, a dispatching rule is used to generate a feasible schedule, and in phase 2, operations are selected from a group of predetermined promising operations. The initial schedule is then improved by left-shifting the start times of these predetermined operations and rearranging the other operations in the schedule.

The advantage of this approach is that it has a very simple formulation. However, there are several drawbacks with this approach. First, this heuristic works for only one scheduling criterion, namely, to minimize total job tardiness, and therefore, may not be suitable for problems with multiple performance criteria. Second, in a general job shop, many operations can be worked on other alternative machines. This algorithm does not

address this issue. Finally, the time taken to improve on the schedule is large compared to the initial schedule generation.

One of the serious drawbacks of using dispatching rules, is that they do not consider the additional information that is available on the shop floor. Look-ahead strategies overcome this shortcoming by considering possible future occurrences in the current dispatching decisions.

A two-fold look ahead search method is proposed in [Itoh et al, 1993]. This method essentially creates a decision tree. There are two stages to this search technique. In the first stage, referred to as the local-look-ahead, the tree is completely expanded so that all short-term future states (say, a horizon of two operations) are considered. For each expanded node of the tree, the next stage of the search method, known as the global-look-ahead is applied. In the global-look-ahead stage, a rough evaluation strategy, which is essentially a dispatching rule, is applied to the scheduling problem, to obtain a global performance measure of each of the nodes. The node with the most superior global performance measure, is kept and the rest of the tree is then pruned. The two-fold look ahead search is then once again applied, with the node that was found to be the most superior in the previous stage, as the starting node. This process is then continued until all the operations are scheduled.

This method is an improvement over dispatching rules, in that it considers the additional information that is available by taking into account the future consequences of a scheduling decision. However, there are several disadvantages to this approach. First, this approach is described as a multi-criteria approach. This approach is not truly a multi-criteria approach. This method incorporates an additional criterion by labeling it a secondary criterion. The reason why the the decision criteria are labelled as primary and secondary criteria is because of the exclusive use of dispatching rules in arriving at the global-look ahead values. The secondary criterion is only used in breaking ties when making a scheduling decision. Moreover, this approach assumes that there is no routing flexibility and does not handle the situation where operations may be worked on several alternative machines. The next shortcoming of this approach is that it considers a node as a single machine-operation assignment. In the case of Job Shop Scheduling, there will arise situations where several machine-operation assignments can be made simultaneously. Finally, this approach uses a dispatch rule to compute a measure of the global look ahead

performance. Similar approaches, such as MADEMA, have used random sampling instead of dispatch rules, thereby to considering decisions that are affected by multiple criteria.

Another look ahead dispatching procedure is proposed in [Zeestraten, 1990]. This approach takes into account scheduling problems with routing flexibility (i.e. when an operation can have more than one alternative machines to be worked upon) and deals with the makespan performance measure.

This approach performs a search in the space of possible states that the system can reach within approximately, twice the average cycle time. The actual state of the system at that point in time is used as a starting point for the search. Possible future system states can be rated using an evaluation function. On the basis of the evaluation function the system selects one of these states and the partial schedule that leads to that state The partial schedule specifies only a few operations for each machine in the production system. During the next phase, the production control system implements the first part of the partial schedule. At most, one operation will be implemented on each machine. Whenever the need for yet another operation arises, the state space search will be repeated. Any disturbances that may have occured in the meantime will be reflected in the current system state, and thus can be taken into account when a next partial schedule is produced [Zeestraten, 1990].

The approach formulated works best for the makespan performance measure. However, most job shop scheduling problems are multi-criteria and modifying this approach for such criteria may not be straightforward, or the advantage obtained in looking forward to only, twice the average cycle time will be sufficient. Expanding the search space to large search horizons will cause an explosion of states that need to be considered.

The job shop, is a dynamic environment which is constantly affected by factors that makes the change of existing schedules inevitable. Rescheduling is a term that is often used in the literature to describe the constant change of schedules. A rescheduling heuristic is proposed in [Li et al, 1993]. Factors that necessitate revisions to existing schedules, are termed as rescheduling factors and include the following:

a) machine breakdown
b) interruption due to the arrival of 'very urgent' jobs.
c) shortage of materials

d) quality problems
e) over or under estimating process times
f) cancellation of jobs
g) changes in the due dates of jobs

An encounter with a rescheduling factor results in two kinds of disrupted operations. First, there are the directly affected operations (e.g. the operation that is being currently worked on a machine when it breaks down). Second, there are the indirectly affected operations (e.g. when a machine breaks down, the next operation to be operated on the machine and the subsequent operation of the job that was on the machine when it broke down are indirectly affected operations) [Li et al, 1993].

The rescheduling heuristic creates a "scheduling binary tree" beginning with a directly affected operation and creating branches of indirectly affected operations. These operations are then rescheduled, by computing the new start times and end times of the affected operations. The indirectly affected operations may in turn, affect other operations in the schedule. As a result the branches of the previous tree may grow and add on newer branches. The heuristic continues to reschedule all the operations that are part of the tree. The rescheduling terminates when all the branches of the scheduling tree have been traversed [Li et al, 1993].

The advantage of this heuristic is that it provides a 'quick-fix' to existing schedules. By creating a scheduling tree, it does not tamper with unaffected operations and only concentrates on operations that have been affected. On the other hand, the heuristic assumes that the operation sequence of each machine remains unchanged. This may not be the case in real job shop situations, where the operation sequence of machines may change due to the arrival of new jobs, and when operations may be worked on several alternative machines. In such situations, it may still be necessary to regenerate the schedule, instead of rescheduling.

### 2.1.3 Other Heuristic Rules

In this section two approaches, viz, constrained based heuristics and efficient solution heuristics will be discussed.

The dominant constraints in a job shop are the temporal activity precedence and the resource capacity constraints. The activity precedence constraints, along with the job's release date and due date, restrict the set of acceptable start times for each activity. The capacity constraints restrict the number of activities that can use a resource at any particular point in time and create conflicts among activities that are competing for the use of the same resource at overlapping time intervals [Miyashita and Sycara, 1994 and Sycara et al, 1991].

The constraint-based scheduling approach views each activity as a variable. A variable's value corresponds to a reservation for an activity. A reservation consists of a start time and the set of resources needed by the activity. A schedule is built by opportunistically selecting an activity to be scheduled, new constraints that reflect the activity reservation are added to the initial scheduling constraints. These new constraints are then propagated. If an inconsistency (violation of constraints) is detected during propagation, the system backtracks. Otherwise, the scheduler moves on and looks for a new activity to schedule and selects its corresponding reservation. The process goes on until all activities have been scheduled successfully [Miyashita and Sycara, 1994].

A scheduling algorithm making use of the concept of efficient solutions for a job shop with parallel machines is developed in [Cenna and Tabucanon, 1991]. An efficient solution is one in which no increase can be obtained in any of the objectives without causing a simultaneous decrease in at least one of the objectives. This approach was applied to a job shop using two performance measures, viz, minimizing total flowtime and minimizing maximum tardiness.

## 2.2 Schedule Permutation Methods

As the name of this category suggests, these scheduling methods first generate a feasible schedule, often by using a dispatch rule. These methods systematically permutate this initial schedule and after a period of time, return the best schedule found to date. The manner in which the permutation is performed, lends these methods their name. Generally, these approaches may be classified as

- Genetic Algorithms
- Simulated Annealing
- Taboo Search.

Some of these approaches may also be classified as AI approaches. However, their unique problem formulation that requires the permutation of schedules, makes it appropriate to classify them as above.

### 2.2.1 Genetic Algorithms

Genetic Algorithms are adaptive search techniques imitating natural selection and genetics. The key idea here, is to maintain a set of candidate schedules, called population. Members of the population are usually represented as strings (chromosomes) and evaluated according to a fitness function (performance measure). The population is initialized at random and evolves in cycles (generations). In each generation, the population is affected by genetic operators and selection mechanism. Genetic operators such as crossover and mutation, provide information flow among individuals, while selection promotes survival of the fittest population members. Through recombination and selection, progressively better strings are constructed from the best building blocks from past generations. The evolution process thus converges to highly fit population members representing good (near-optimal) schedules [Filipic, 1992 and Uckun et al, 1993]. The Genetic Algorithms will be discussed in greater detail in Chapter 4.

### 2.2.2 Simulated Annealing

Simulated Annealing is a search procedure based on the analogy between the process of annealing in solids, as described by the models of statistical mechanics, and the process of combinatorial optimization. In general, when a solid is annealed, the positions of the molecules are gradually evolved into a configuration of very low strain energy in the solid. If we make an analogy between the molecular positions and internal strain energy of a solid to the schedule and the quality of the schedule (measured by some performance measure), then the statistical mechanical models which describe solid annealing can provide a recipe for the generation of a good schedule [Chryssolouris, 1992].

A simulated annealing approach to job shop scheduling is discussed in [Musser et al, 1993]. This approach considers both routing and the sequencing of operations. A "two-level" stochastic descent approach, based on simulated annealing is proposed - a

"higher" level method for routing and a lower level for sequencing. The simulated annealing approach treats each schedule as a state in a discrete time finite state Markov chain. The transition probabilities of the Markov Chain are selected so that the state drifts towards the lower cost states. The simulated annealing algorithm uses a Markov process, whereby an initial schedule is chosen at random. Next, a nearby trial schedule is chosen and the two schedules are compared. If the trial schedule has lower cost, it is accepted as the new state of the Markov chain. Otherwise, it is accepted with a lower probability determined by the relative costs, which is expressed in terms of a parameter called temperature. The schedule cost is calculated as the weighted average of the sum of three cost components, viz, work in progress, tardiness and inventory cost. If the trial schedule is only accepted when it has lower cost, the algorithm will remain in any local minimum it may find. This is the case when the temperature is zero. A non-zero temperature, however, gives the Markov chain the opportunity to escape from local minima. In the simulated annealing process, the temperature starts of at a very high value and tends to zero. In this approach, the trial schedules are generated by swapping adjacent operations on a single machine.

### 2.2.3 Taboo Search

The taboo search procedure, is a global iterative optimization method. The search moves from one solution to another, in order to improve the quality of the solutions visited. When the search arrives at a local optimum, it does not terminate, but moves beyond the local optimum by choosing the best possible neighbouring solution. A move to a neighbouring solution involves permutating two successive operations that use the same machine. In order to avoid cycling, the move that leads back to the local optimum (just found) is forbidden. This is achieved in a short-term memory framework by keeping the forbidden (taboo) moves in a taboo list. For a taboo list of a given size, when an element is added to the taboo list, another (the oldest move) is removed. The size of the taboo list must be large enough to avoid cycling, but small enough not to forbid too many moves. However, it may happen that an interesting move is taboo (such a move that improves the best solution already found). In order to perform such moves, an aspiration level is defined (depending, for example on the current solution and the best solution found). To ensure diversification of the search, a long-term memory mechanism is also implemented [Taillard, 1994].

## 2.3 AI Approaches

The field of Artificial Intelligence involves ideas and methods that make computers "intelligent". Its main goals are to make computers more useful and to understand the principles that make intelligence possible [Chryssolouris, 1992]. In the following subsection, three artificial tools that have found increased used in the field of job shop scheduling will be described. These tools include

- Search
- Neural Networks
- Fuzzy Logic

### 2.3.1 Search

Search methods find solutions by exploring paths. What distinguishes one search method from another is the heuristics which decide how the exploring is to be done. In the field of job shop scheduling, this is often done by starting with a feasible schedule. The search process explores the schedule space, moving from schedule to schedule, evaluating each as it is explored. The search usually uses the information from the previous schedules to determine the path of future moves through the schedule space. In the end, the search would have arrived at a final schedule, either because an "optimal" schedule was found or because some limit on the computational limit has been exceeded. In any case, the final schedule is usually better than the initial starting schedule [Chryssolouris, 1992]. There are various search methods reported in the literature, and in this section, the application of one such search procedure to a job shop scheduling problem will be discussed.

In the approach described in [Chang et al, 1989], the flexible job shop problem is investigated. A flexible job shop, as opposed to the general job shop, is one in which there is routing flexibility. A procedure is developed to identify the potential bottleneck machine and a beam search technique is subsequently formulated to solve the job shop scheduling problem. In [Aarts et al, 1994], the computational performance of several search algorithms are investigated.

### 2.3.2 Neural Networks

Neural Networks can perform two basic functions: they can be trained to remember some information and they can be used to perform constraint satisfaction and optimisation tasks [Willems and Rooda, 1994]. Neural Networks have been proposed in the literature, to solve job shop scheduling problems that are viewed as either one of the two perspectives mentioned above.

An example of using neural networks that are trained to remember information in the job shop scheduling context is described in [Sim et al, 1994]. A feed-forward back propagation neural network is designed and trained to recognize the individual contributions of traditional dispatch rules. The neural networks are expected to learn and store in their structure, the relative factors that influence the various considerations for dynamic job shop scheduling. Given sufficient realistic examples, the network can be trained to recognize how these considerations cooperate or compete in the assignment of jobs as they mutually reinforce or nullify their influences on meeting the scheduling objective. The network is incorporated into an expert system which activates the network according to the prevailing shop environment.

The expert system and the neural network approaches are integrated to form an expert neural network in order to overcome the weakness associated with each methodology. The expert system will reduce the amount of time required for training the artificial neural network by allowing the sub-networks to be trained separately. The neural network will learn about and deal with the complex interactions of the scheduling considerations without the need for the long knowledge-acquisition and development time of expert systems.

The approach cited in [Sim et al, 1994] is an attempt to make use of the existing dynamic conditions of the job shop in making scheduling decisions. There are several shortcomings of this implementation. First, the jobs arrival rate in a real job shop environment cannot be measured accurately. Perhaps, this information could have been represented using linguistic variables such as "high", "low" or "medium" to describe the existing workload conditions instead of a Poisson value. Second, this approach does not use multiple criteria to make scheduling decisions. Finally, this approach may not be able to handle the situation where there is routing flexibility, where there are more than one alternate machines to perform an operation.

Another example of using neural networks that are trained to remember information is described in [Watanabe et al, 1993]. In this approach, neural networks are used to improve the SLACK dispatch rule. The SLACK rule is used in scheduling to meet due dates. This rule assigns the job in a queue with the shortest margin to its due date. The margin is calculated by subtracting the operation time of the job from the remaining time to the due date. This margin is referred to as the theoretical margin. However, the actual margin of the job is shorter, due to the 'interference' of other jobs. Neural networks are proposed to estimate the actual margins to the due date.

This approach is an attempt to improve a single dispatch rule with a neural network. This method is basically a single criterion approach. The shortcomings due to dispatch rules, discussed earlier, also apply to this approach, except that this approach uses additional information existing or prevailing in the job shop.

Examples of neural networks used in job shop scheduling from the constraint satisfaction and optimisation perspective are described in [Zhou et al, 1991, Foo et al, 1994, and Willems and Rooda, 1994]. In job shop scheduling, the search for a schedule is bound by two types of constraints. The first constraint states that the mandatory precedence sequence of operations must be guaranteed and the second constraint states that not more than one job at a time can be processed by a machine at the same time [Willems and Rooda, 1994]. The job shop problem is modelled as a mixed integer linear programming problem. The set of equations describing the linear program is exactly identical in all these approaches, however, the structure of the network is different.

The approaches cited in [Zhou et al, 1991, and Foo et al, 1994], use the Hopfield and Tank linear programming network, consisting of electronic elements such as diodes, amplifiers, resisters and capacitors, to model the scheduling problem. The approach provided by [Zhou et al, 1991] as compared to the approach of [Foo et al, 1994], has less network complexity, however, the computations required of each neural processing element is more extensive. The neural network in [Willems and Rooda, 1994] is not modelled as electronic components, but rather as neurons, as in the more traditional neural networks. This approach unlike [Zhou et al, 1991, and Foo et al, 1994] does not make use of an energy function.

The neural network approaches cited in [Zhou et al, 1991, Foo et al, 1994, and Willems and Rooda, 1994] were modelled for the minimum makespan problem. In order to accommodate multiple criteria scheduling, the cost function in the mixed integer linear program, needs to be reformulated. Moreover, this approach works for the case where there is routing inflexibility (i.e. operations can be performed on only one machine). The structure that has been proposed in these approaches are best suited for the static scheduling problem. This is not a serious disadvantage, because, the dynamic scheduling problem could be viewed as a series of quasi-static problems and can be solved accordingly using these approaches.

### 2.3.3 Fuzzy Logic

Several scheduling methodologies have been proposed in the literature that use fuzzy logic to build up aggregated rules that achieve satisfaction of several criteria [Turksen et al, 1993, Grabot and Geneste, 1994, and Custodio et al, 1994]. In the approach reported by [Turksen et al, 1993], the SPT rule is selected to be one of the linguistic variables in the fuzzy rule base, because of its favorable properties in terms of the flow time and tardiness criteria. Another dispatch rule that is included in the rule base is the slack time rule. This rule is complementary to the SPT rule. The slack time rule is inherently dynamic in that it is computed based on the current time and the remaining operation time. This approach makes two types of scheduling decisions. The first decision, is the job release decision and this decision is made using the fuzzy variables of "priority", "slack time" and "requested start date." The second decision is that of the job dispatch decision and it depends on the fuzzy variables of "priority", "slack time" and "remaining processing time." Each of these variables has linguistic values associated with them, for example "high, low, medium etc."

In a similar approach reported in [Grabot and Geneste, 1994], two or more dispatching rules can be combined into an aggregated rule using Fuzzy Logic, with intermediate performance, between those of the elementary dispatch rules that compose the aggregated dispatch rule.

In [Custodio et al, 1994], a production system with a hierarchical structure that consists of three levels (higher, middle and lower) is proposed. These levels are each responsible for a different production problem with a different time scale. The

methodology approaches the tasks associated with each level, using a heuristic formulation and solves the short-range planning and scheduling problems with a non-stationary policy. The higher decision level determines safety stock levels used to compensate for future resource failures. At the middle level, loading rates are computed. This is achieved through a fuzzy controller that tends to minimize the error between the cumulative production and the cumulative demand while keeping the work in process below acceptable values. Finally the lower level controls the flow of parts among the resources using a fuzzy decision method, which has the ability to use several criteria to generate a decision.

## 2.4 Analytical / Semi-analytical Approaches

These approaches formulate the job shop scheduling problem, in terms of mathematical models that are described using differential or difference equations, often highly coupled and nonlinear. Assumptions pertaining to the nature of the job shop scheduling problem allow for simplifications to be made to these equations, so that they become more tractable. In this section, seven such approaches (Figure 2.3) will be discussed.

### 2.4.1 Control Theoretic Formulation

In classical formulations of scheduling problems, all events are assumed controllable, and subject to constraints. However, some events such as machine failures and material supply disruptions are not controllable and can be modelled as random events. The goals of the hierarchical flow control formulation of scheduling problems proposed by [Van Ryzin et al, 1991 and Gershwin, 1994] are:

- to reduce computational complexity of scheduling problems by dealing with rates of events rather than individual events, wherever possible.
- to include random events (e.g. machine failures).

The machine failures are assumed to be much less frequent and of much greater duration than the operations. This assumption allows for the hierarchical decomposition of the scheduling problem. The hierarchy consists of three levels.

At the highest level, the flow control strategy is determined. The flow control strategy involves the setting of short term production rates as a function of the machine repair states and production surplus. The flow control strategy is then implemented in the middle level which operates on a long time scale.

Instead of viewing parts as discrete heterogeneous jobs, the middle level considers a homogeneous material flow which is subject to random interruptions due to machine failures. The dynamics of the material flow are represented by simple, first order differential equations, and failures and repairs are modelled as jump Markov disruptions in the material flow. The objective is to control the flow of parts over a long time horizon so that demand is satisfied as closely as possible, and inventories, surpluses and backlogs are kept low, while keeping the system within production rate constraints. At the lowest level, the short term production rate is used as a target for the loading of parts. The actual loading of individual jobs into machines is left to low level controllers which work at a shorter time scale. The low level attempts to fulfil the production goals determined by the higher level controllers.

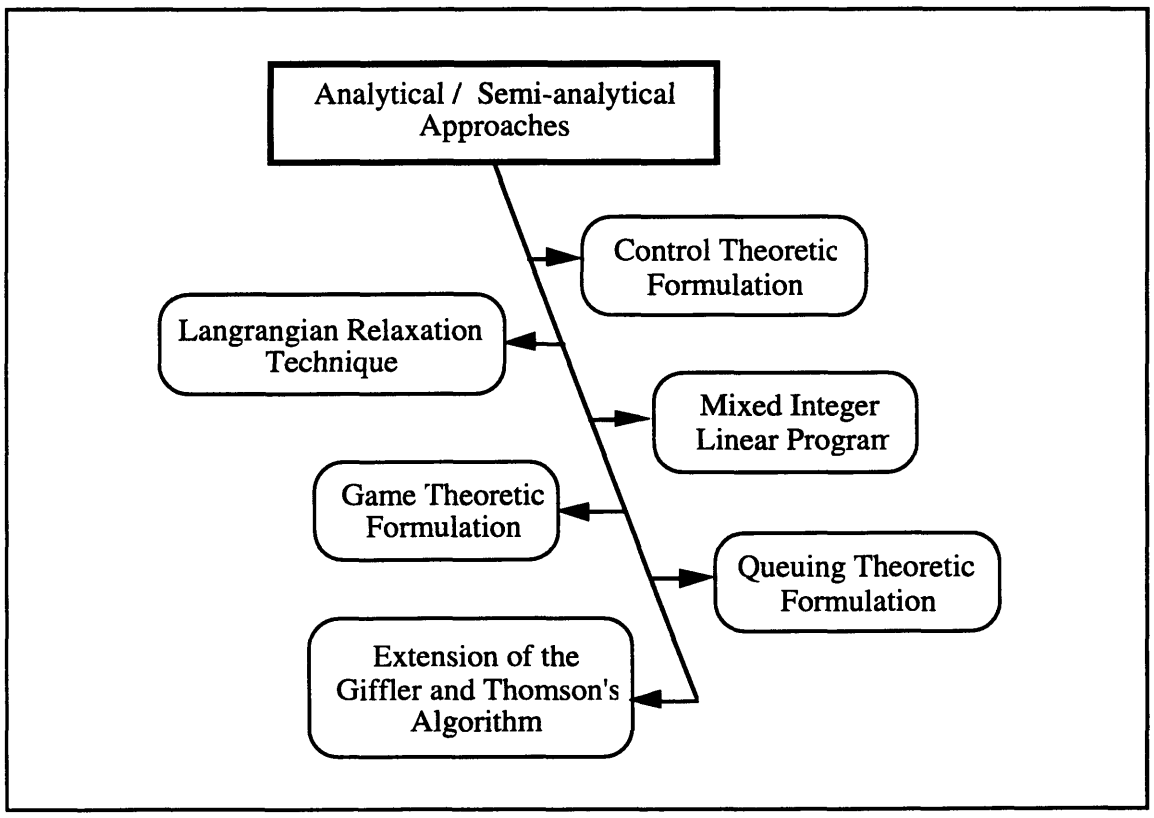


Figure 2.3: Summary of the Analytical / Semi-analytical Approaches.

This approach may not be suited for scheduling the production of a high variety of jobs, where the demand for each job type may be very small and the continuous variable approximation may be poor. In order to consider individual jobs, the discrete nature of the job flow must be retained [Luh and Hoitomt, 1993].

### 2.4.2 Lagrangian Relaxation Technique

Scheduling, using the lagrangian relaxation technique, is viewed as a decision problem with discrete variables, where an objective function is optimized subject to relevant constraints. The decision variables are the beginning times of the various operations required to produce jobs [Luh and Hoitomt, 1993 and Hoitomt et al, 1993].

Lagrangian relaxation is a mathematical programming technique for performing constrained optimization. This technique is used to decompose a scheduling problem into job or operation level subproblems, and the performance measure considered is tardiness. The lagrange multipliers act as prices to regulate the use of machines.

This approach is quite complex and is not practical if the precedence structure involves many operations. This method is effective for jobs with fewer than three or four operations.

### 2.4.3 Mixed Integer Linear Program

In the approach discussed in [Nasr and Elsayed, 1990], the job shop scheduling problem with alternate machine tool routings are investigated with the objective of minimizing the mean flow time. The problem is analytically formulated as a mixed integer program. An efficient algorithm is developed, based on this formulation, by decomposing the original problem into smaller subproblems, that are easier to solve.

The scheduling process is viewed as a decision making process in which a decision is made, when an operation is completed. The decision could be to either schedule the successor of the finished operation, if the machine is available, or simply wait and not schedule operations.

### 2.4.4 Game Theoretic Formulation

The approach considered in [Leon et al, 1994], utilizes information about future disturbances to control the schedule execution. This approach allows the use of static schedules in an uncertain environment. This approach looks at control strategies to guide recovery of the system from disruptions to a pre-computed schedule. The scheduling problem is formulated as a resequencing problem to minimize the expected value of a bicriterion function, where one criterion is the makespan and the other is the deviation from the static, precomputed schedule.

This online control problem is treated as a game, where the controller plays against nature. The controller proposes alternative ways to schedule and nature places stochastic events on the schedule, such as machine disruptions. The controller takes action in two cases, determined when significant new information becomes available. First, when a disruption occurs and second, when the system reliability falls below a certain threshold.

This approach assumes that the interarrival time distributions of disruptions of each machine and the disruption duration are available. These parameters are used to compute the probability of having the next disruption on a machine in a specific time interval. Subsequently as time proceeds, the inter arrival time marginals and the distribution for the arrival of the next disruption are updated.

### 2.4.5 Queuing Theoretic Formulation

The approach reported in [Enns, 1993], investigates analytical methods of due-date setting in a job shop under conditions where Jackson's decomposition principle can be applied, using queueing analysis. Sufficient conditions under which the Jackson's decomposition principle may be applied are as follows:

- Job arrivals are randomly generated by a Poisson process with a stationary mean.
- Operation processing times are independent and generated from negative exponential distributions with the same mean.

- The routing of jobs is specified by probabilities in a fixed transition matrix.

- Jobs in queues at each machine are processed on a first come first served basis.

Previous work in the literature has lead to results which minimize squared deviations between due-dates and actual completion times. [Enns, 1993], extend these previous works by considering the distribution of prediction error. If the distribution of errors associated with completion time predictions can also be determined, statistical analysis can be used to develop a due-date setting model which allows management to control expected delivery performance.

The use of analytical techniques such as queueing theory and other statistical methods, usually results in some probabilistic distribution functions in the limit. The close form solution of this functions exist, only because of the restrictive assumptions that are necessary to make the problem tractable. For example, the assumption that Jobs in queues at each machine are processed on a first come first served basis is a sufficient condition for the Jackson decomposition principle to apply, but results in the literature indicate that the FCFS dispatch rule is usually outperformed by other dispatch rules for most performance measures. Moreover, the probabilistic distribution functions do not provide with a detail schedule, but rather indicate the overall performance of the scheduling system in the limit, if the restrictive assumptions are applied in practice. The analytical techniques are useful in understanding the mechanics of the scheduling process (as per the assumptions) in the long run, but do not provide the Job Shop with any useful schedule in the short run.

### 2.4.6 Extension of the Giffler and Thomson's Algorithm

The scheduling problem discussed in [Chang and Sullivan, 1990] involves operations that can be performed on alternate machines with different operation processing times, and therefore involves both sequencing of operations as well as routing them. This scheduling formulation is an extension of the Giffler and Thomson's scheduling algorithm.

A *feasible* schedule is defined as one where the machine must be available for an assigned operation and that the start time of the next operation of the same job does not begin before the earlier operation is completed. *Global Left-shift* is an operation that moves an operation within the schedule so that it starts as early as possible without delaying other

operations. An *active* schedule is a feasible schedule where no Global Left-shift is possible.

The Giffler and Thomson algorithm generates all active schedules for a particular Job Shop problem. The number of active schedules is extremely large, even for a small scheduling problem. This paper proposes an extension of the Giffler and Thomson algorithm that reduces the number of active schedules significantly. A search method is then employed within the reduced schedule space to find an optimal or near optimal schedule.

This approach works best for cases where the makespan or other performance measures that require the active schedules. There may be instances, based on other performance measures such as cost or quality, where the delay of a particular operation may be beneficial. By eliminating all non-active schedules, this approach may not result in good schedules for multi-criteria scheduling.

## 2.5 Other Approaches

This subsection explores job shop scheduling methods that have been reported in the literature and cannot be classified into the previous four categories. Two such approaches will be discussed in this subsection.

The first approach deals with the problems of dynamic scheduling and integration of a job shop and its environment [Sun and Lin, 1994]. A dynamic scheduling framework is presented in which dynamic scheduling is carried out through a series of static backward scheduling problems.

The backward approach is a common methodology being practiced in the field of production planning and control. MRP systems carry out backward scheduling to determine the order release times. JIT systems use the backward approach in the sense that the operations are controlled by the signals from downstream workstations [Sun and Lin, 1994].

The framework facilitates the integration by establishing the relationship between scheduling, due-date assignment and job release times through backward scheduling.

When a job shop scheduling problem is presented as a backward scheduling problem, due date performance improvement and inventory cost reduction, can be represented in a more straightforward way [Sun and Lin, 1994].

The jobs' finishing times in a backward schedule are the times when the jobs actually start their first operations in the forward time frame. In backward scheduling, if the jobs' available times are set to be the backward times corresponding to their due-dates in the forward time frame, the jobs' backward finishing times in the resulting backward schedule will become the latest forward release times of jobs in the forward time frame, i.e., if the jobs are actually released to the shop floor by those latest release times, all due-dates can be met. As a result, the jobs' finishing times in a backward schedule can be conveniently used to control the jobs' actual release times [Sun and Lin, 1994].

In the second approach [Chu et al, 1992], the spatial splitting up approach, in which the job shop is split into several small and almost independent job shops (also referred to as subsystems) is considered. The initial scheduling problem is then replaced by a set of smaller scheduling subproblems.

The splitting up approach reported in the literature, aims at partitioning the set of machines into a set of subsystems, the set of parts into a set of part families and, simultaneously, at establishing a one-to-one relationship between the set of subsystems and the set of part families in order to maximize the number of operations performed on the parts outside the related subsystem [Chu et al, 1992].

When none of the operations on the parts is performed outside the related subsystems (i.e. when there is no traffic between the subsystems) the partition of the machines is said to be perfect. In that particular case, every job is manufactured using only one subsystem. Thus the optimal schedule of the entire problem is the concatenation of the optimal schedules of the subsystems. In general, however, some of the jobs have to visit machines belonging to different subsystems. A heuristic algorithm is developed to manage such links between subsystems [Chu et al, 1992].

## 2.6 Research Framework

To illustrate the usefulness of the methods developed in this thesis, these methods will be applied to two static problems that are often cited as testbeds in the literature, viz, the Muth and Thompson 6x6 and 10x10 problems [Muth and Thomson, 1963]. The following framework was developed for the scheduling problem:

1 Check for available machines.
2 Check for operations that can be assigned to the available machines.
3 Form alternatives that assign available operations to available machines.
4 Evaluate these alternatives (using one of the two methods developed in this research).
5 Implement the best alternative.
6 Goto step 1, if all the operations are not assigned.

This framework is generic and will also be applied to a dynamic scheduling problem where machines are liable to breakdown and job arrivals are random. The arrival of jobs and machine breakdowns could be assimilated into the framework described above and this will be discussed in greater detail in Chapter 5.

## 2.7 An Approach for Comparison

Based on the survey presented in this Chapter, we have seen the reasons for the choice of dispatch rules in the industry. To summarize, dispatch rules are popular, because they are easy to use and implement on any shop floor. The shortcomings of dispatch rules are that there is no one single universal rule for all requirements and that dispatch rules do not make use of the information that is available on the shop floor to improve on the schedule quality.

We have also seen that because of the shortcomings in dispatching rules and the advances in computer technology, research in dispatching rules has continued to grow and the current trends in this research are to develop dynamic rules [Raghu and Rajendran, 1993] or to combine two or more simple dispatching rules into a single composite rule [Anderson and Nyirendra, 1990]. Whilst these approaches have made improvements over the simple dispatching rules, these new rules still suffer the same disadvantage as dispatch

rules, perhaps to a lesser degree. These rules were developed with specific performance measures in mind. If these performance measures were not the ones that a scheduler may be interested in, than these new rules become ineffective.

The concept of combining simple rules into a composite one is a very attractive one. In order to apply this concept to a general job shop environment, the mechanism of combining dispatching rules must be flexible. One such flexible approach to combining dispatching rules was discussed in the Fuzzy Logic Approach [Grabot and Geneste, 1994].

The Fuzzy Logic Approach is a good approach to use as a basis of comparison of the performances of the scheduling strategies that are being proposed in this thesis for the following reasons:

- First, the Fuzzy Logic Approach uses dispatch rules. This is the inherent bias of industries towards dispatching. One of the aims of this thesis is to show that the methods develop in this thesis are a better alternative to dispatching rules.

- Second, it embodies the trends of current research in job shop scheduling.

- Third, it is a dynamic rule and is also flexible in accommodating various performance measures, just like the approaches that will be developed in this thesis and a fair and equitable comparison of these approaches can therefore be made.

- Finally, the prime objective of this thesis is to develop job shop scheduling methods that would work in a dynamic environment. The Fuzzy Logic Approach is also an approach that is well suited for the real dynamic job shop environment.

The initial process of the Fuzzy Logic Scheduling Approach is to identify the decision making criteria. For example, in [Grabot and Geneste, 1994], the decision making criteria were the processing times of the operation and the slack time of the job. In the following discussion on the Fuzzy Logic Scheduling Approach, it will be assumed that there are two decision making criteria, namely the processing time and the slack. These criteria can be expanded to include any other decision making criteria that may be relevant to the scheduling problem.

To recapitulate, the scheduling problem as formulated using the Fuzzy Logic Approach, is exactly the same problem as formulated using dispatching rules. That is, one has to decide which of the operations that are pending at the job shop is the most suitable one to be loaded on to the available resource. The same decision has to be continuously applied, whenever a resource becomes available. The term *alternative*, is used to describe each pending task that can be processed on the resource that has become available.

Therefore, at each decision point, one needs to compute the slack time of the jobs that are pending at the resource and also the processing times of the operations. These values are usually assumed known and their calculations are very trivial. The next step is to normalize the processing times, and the slack of all the jobs that are pending at the machine. Let the subscripts, 's' and 'p' denote variables associated with the slack time and processing times respectively. The equations for computing the normalized slack and processing times are

$$C_{sj} = \frac{z_{sj} - \min_s}{\max_s - \min_s} \tag{2.1}$$

$$C_{pj} = \frac{z_{pj} - \min_p}{\max_p - \min_p} \tag{2.2}$$

where the subscript j denotes the jth job pending at the machine, and max and min refer to the maximum and minimum (slack or processing time) of all the jobs pending at the machine.

Each of the decision variable is expressed using linguistic values contained in a set, which will be termed the Process Term Set (PTS). For example, the PTS may include linguistic values, such as [Mamdani et al, 1984],

$$\{PTS\} = \{\text{low (L), medium (M), high (H)}\} \tag{2.3}$$

These linguistic values are represented by triangular membership functions, $\mu_{PTS}$, in a normalized universe of discourse, C = [0,1] (Figure 2.4). Note that there is no particular

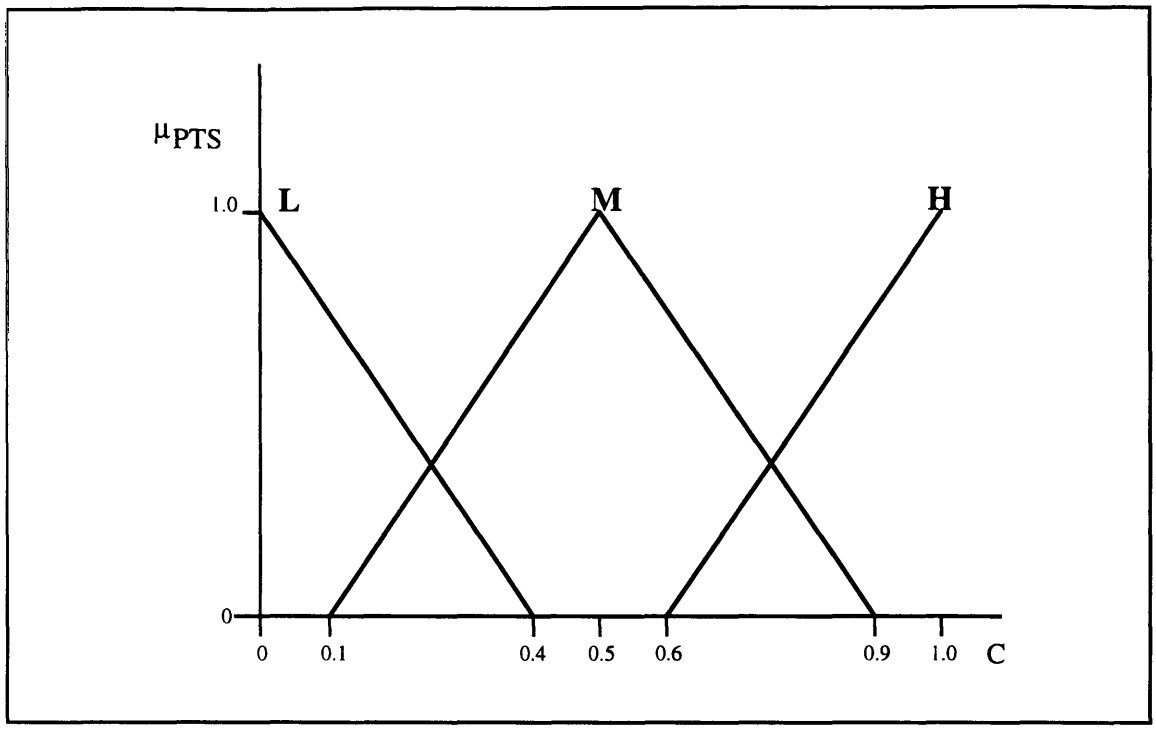


Figure 2.4: Membership Functions of the Linguistic Values Contained in the PTS

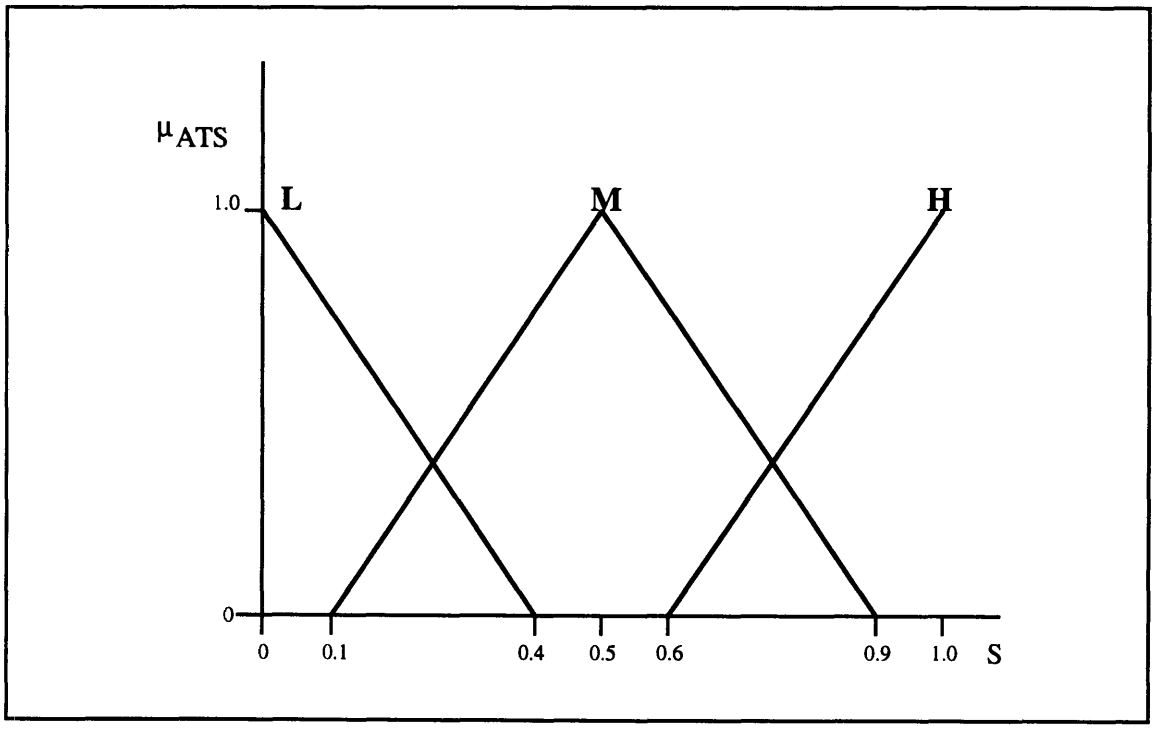


Figure 2.5: Membership Functions of the Linguistic Values Contained in the ATS

reason why the membership function has to be triangular, except that it makes the analysis much simpler.

These decision variables expressed as linguistic values using fuzzy logic will be mapped onto a set of action variables contained in the set, termed as the Action Term Set (ATS). Like the PTS, the ATS is also expressed using the linguistic values that are represented by triangular membership functions, $\mu_{ATS}$, in a normalized universe of discourse, S = [0,1] (Figure 2.5). Once again the triangular membership functions were used to make the analysis simpler. The ATS is simply the degree of suitability of loading the job under consideration onto the available resource.

The next stage of the Fuzzy Logic Scheduling Approach requires the development of decision rules. This is the process of transforming expert knowledge into coded knowledge and through these decision rules, one can map the decision variables to the action variable. The structure of a decision rule is

**If** {decision variable 1 is $PTS_i$ and decision variable 2 is $PTS_j$} **THEN** action variable is $ATS_k$

In our example the decision variables are slack and processing time, and the action variable is scheduling suitability, one plausible set of decision rules is as summarized in Figure 2.6. The partial matching attribute and the overlapping preconditions of the decision rules, usually results in more than one decision rule being fired at any one time. This situation is referred to as "fuzzy conflict" [Lee, 1990, and Qiao et al, 1992]. A methodology, based on the Max-Min Compositional rule, may be used to choose the precise suitability value for each job pending at an available resource.

To illustrate this methodology, assume that the normalized values of the slack and processing time for a particular job were calculated to be 0.25 and 0.8 respectively (Figure 2.7). Using these values, the membership or truth values of each rule can be calculated (only those rules that are fired and have non-zero truth values are shown in Figure 2.7). The strength of each rule is the minimum of the truth values and is mapped onto the membership function of the action variable (suitability). To determine the precise value of the suitability, the centroid of the aggregate shape is used. The centroid is computed as [Kosko, 1992].

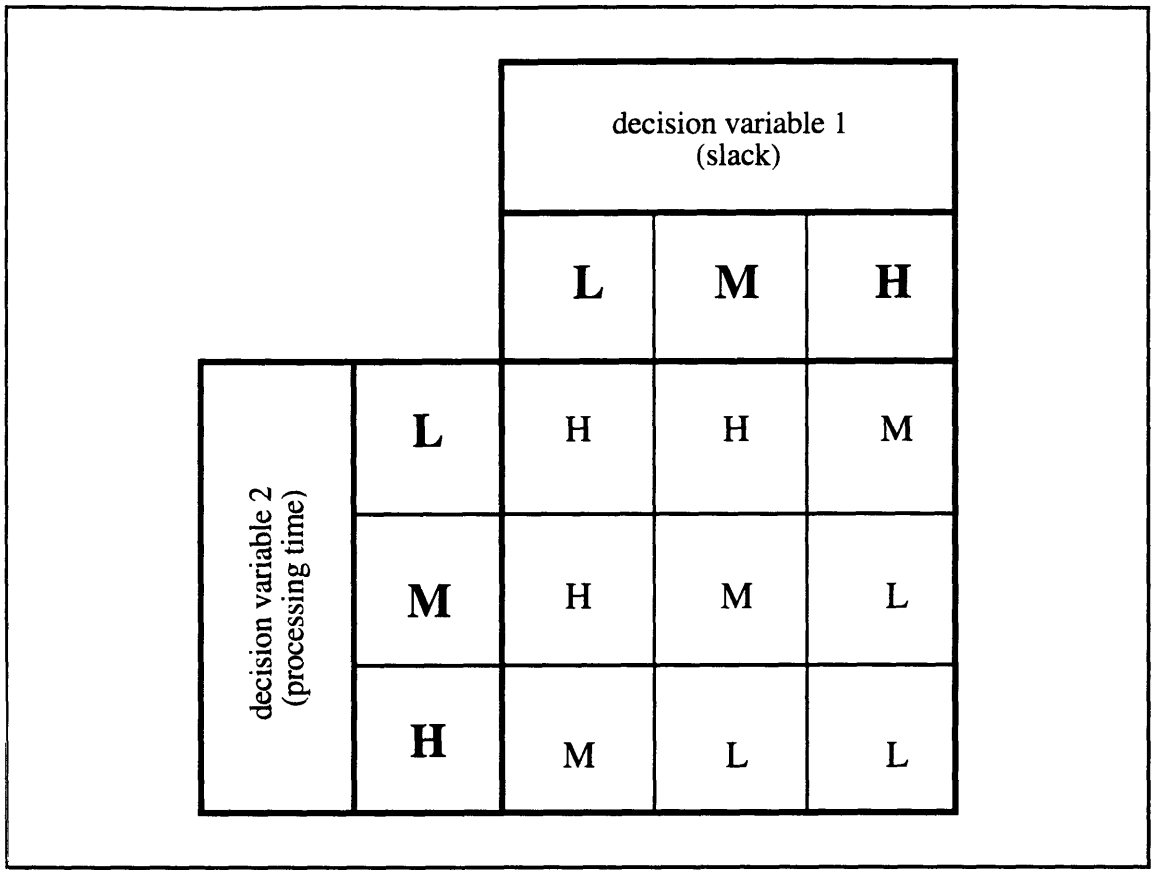


Figure 2.6: Fuzzy Decision Rules

$$\text{cent} = \frac{\sum_i w_i c_i I_i}{\sum_i w_i I_i} \qquad (2.4)$$

where $I_i$ is the area of shape i, $c_i$ is the centroid of shape i and $w_i$ is the truth value associated with the shape i. The choice of triangular membership functions makes the computation of the area, centroid and truth values easy.

The Fuzzy Logic Scheduling Procedure may be summarized as follows:

Begin

A set of Fuzzy rules linking decision variables (in terms of linguistic variables) to action variable (suitability, also expressed as linguistic variables) must be developed a priori.

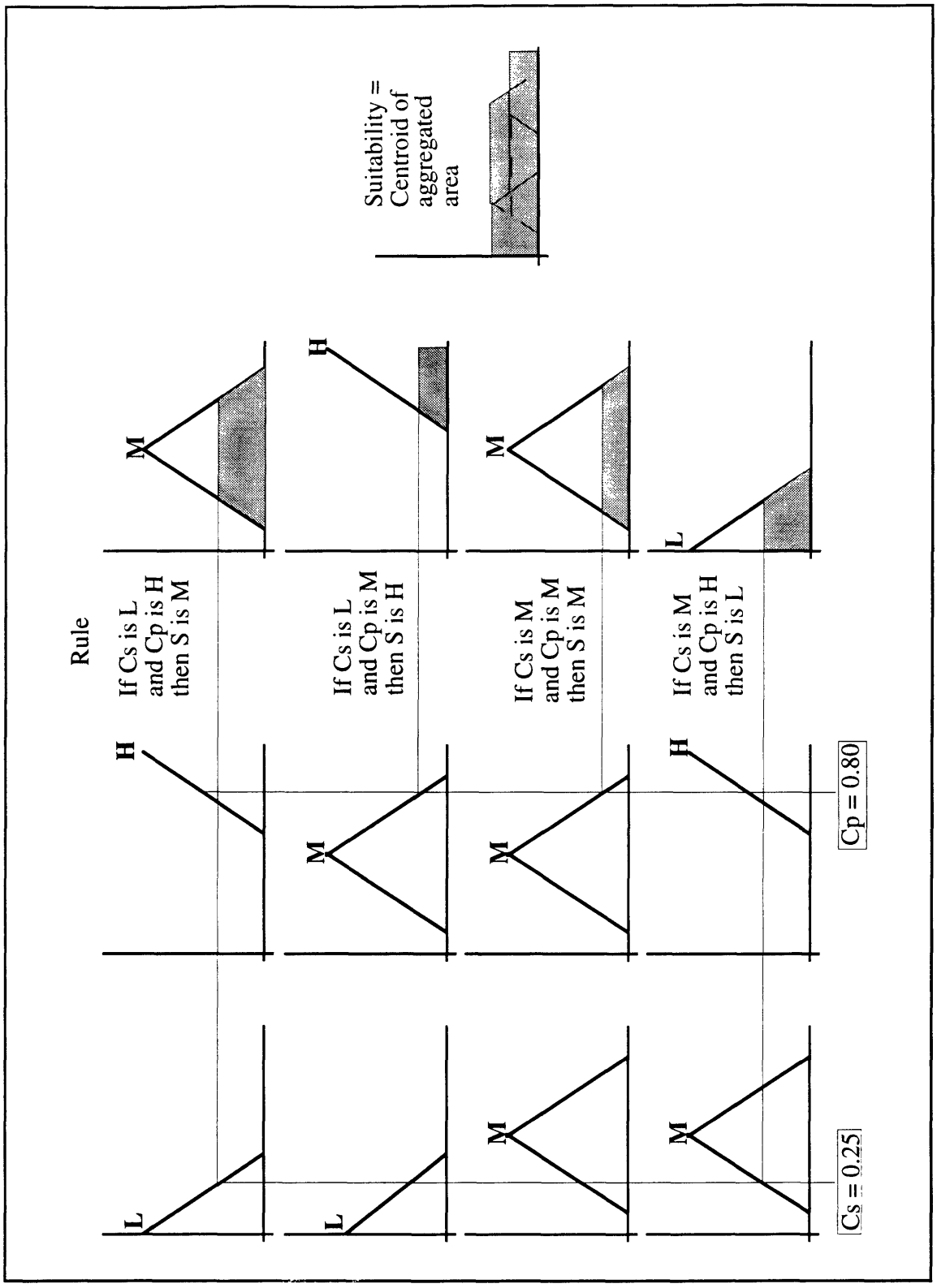


Figure 2.7: Resolution of Fuzzy Conflicts

Step 1 Each time a resource becomes available, compute the slack and processing times of all the jobs that are pending at the resource. After which, normalize the slack and processing times of each job.

Step 2 Using the normalized values for slack and processing times for each job, apply the Max-Min Compositional rule and compute the fuzzy centroid of the aggregated shape. The fuzzy centroid is the precise or crisp value of the suitability of loading the job of interest on the available machine.

Step 3 Load the job on the resource that has the largest suitability value.

Step 4 Go back to step 1, if there are jobs pending at the job shop.

# Chapter 3
# Job Shop Scheduling using Extreme Value Theory

In this chapter, a scheduling method based on Extreme Value Theory will be developed. This method looks at the extrema of the distribution of feasible schedules to evaluate the quality of candidate machine-job operation assignments. This method will be tested on the Muth and Thompson 6x6 and 10x10 static job shop scheduling problems where the performance measure is the makespan. The makespan obtained for these problems using the Extreme Value approach will be compared with those obtained using some of the common dispatching rules and the Fuzzy Logic approach (discussed in Chapter 2), which may be considered as a representative of current scheduling approaches. The application of the Extreme Value approach to dynamic job shop scheduling will be discussed in Chapter 5.

## 3.1 Introduction

Extreme Value Theory has been widely used to model the extremes of natural phenomena and other rare events, such as in flood analysis [Smith, 1987], air quality analysis [Chock and Sluchak, 1986], reliability analysis [Smith, 1986] and corrosion analysis [Laycock et al, 1990]. In these applications the focus is not in the main body of data, but rather in the probability distribution tails of the data [Guida and Longo, 1988]. A number of techniques have been developed to predict the probability tails (which reflect the occurrences of extrema) based on historical data [Castillo, 1988, and Gumbel, 1958]. Some of these techniques are used to develop a decision theoretic approach for solving job shop scheduling problems. The author is not aware of any prior applications of Extreme Value Theory to job shop scheduling problems. For the reader who may not be familiar with Extreme Value Theory, Appendix A offers a brief overview of the main concepts involved.

In decision theoretic approaches, a decision point arises whenever one or more production resources become available after completing previously assigned operations or after repair. A resource can be any individual production unit such as a single machine, an operator, or a manufacturing cell of machines grouped together with auxiliary devices (e.g. robots). Resources can be grouped into work centers according to common manufacturing functions. The decision to be made in each work center (at each decision point) is which of the pending operations (operations that are ready to be performed) should be performed next on each of the available resources. The partitioning of this decision making problem into work centers renders it more computationally manageable while still allowing the consideration of all feasible resource-operation assignments. A feasible decision alternative is a list of resource-operation assignments in which the resources are the ones that are available in a work center at the decision point and the operations are selected from those pending at the work center at the decision point. For example, if two resources, R1 and R2 are available and three operations, T1, T2 and T3, are pending at a decision point, and assuming that all operations can be performed on either of the two resources, then the six possible alternatives are (R1-T1, R2-T2), (R1-T2, R2-T1), (R1-T1, R2-T3), (R1-T3, R2-T1), (R1-T2, R2-T3) and (R1-T3, R2-T2) (Figure 3.1). The schedule for the work center can be generated by deciding on alternatives at consecutive decision points over time: the schedule for an entire facility can be generated by combining the schedules of its constituent work centers.

## 3.2 Mechanics

One simple approach to generate a schedule is to use a random procedure. In this procedure, whenever a machine becomes available, a suitable operation (i.e. one that can be performed on the available resource) is selected randomly and assigned from the list of pending operations at the work center. In this manner, a schedule is obtained when all operations are assigned to the machines. An expansion of this approach is to produce several and the best schedule, as per some schedule measure (e.g. makespan), is then implemented.

A similar procedure was the basis in the development of the scheduling approach based on Extreme Value Theory. This procedure shall be referred to as SEVAT. Instead of choosing operations randomly from queues, SEVAT uses a more systematic way of generating the final schedule.

The SEVAT method will be illustrated with the aid of a generic example (Figure 3.1), where there are 6 resources (R1, R2, R3, ..., R6) and 7 operations, (T1, T2, T3, ..., T7) that can be immediately assigned and are pending at the work center. Without the loss of generality, let us assume that only resources R1 and R2 are currently available and of the pending operations, only operations T1, T2 and T3 can be assigned to the available resources. Therefore, as per our discussion earlier, there are six possible alternatives (as illustrated in Figure 3.1). The question then arises, "Which one of the six alternatives should be implemented?" The SEVAT procedure addresses this question by evaluating the potential of each of the alternatives.

The potential of an alternative is determined with the aid of simulation, by generating random schedules that originate from that alternative. The generation of random schedules could be

- **exhaustive.**

  In this case, the alternative under consideration is assigned to the available resources in a simulation. This assignment will trigger the release of further operations that can be immediately assigned into the workcenter. Since the processing time of each operation-resource allocation is assumed to be known a priori, the simulation can step through future decision points. At each of these simulated future decision points, new alternatives are formed and one alternative is randomly chosen and assigned, and this will further trigger other operations to be released into the simulation. This form of random assignment is continued until no more operations can be released and all operations are assigned. The total sum of assignments in the simulation will constitute one random schedule and the quality of this schedule is evaluated by some schedule measure (such as cost, workpiece quality, flowtime or makespan). Several such random schedules are required to evaluate the potential of an alternative.

- **partially exhaustive.**

  In this case, the schedules are generated by neglecting the release of further operations in to the simulation. Only the operations that are currently pending at the work center and that can be immediately assigned are considered in the simulation. As soon as an alternative is assigned in the simulation, new alternatives are formed from the remaining pending operations at the work center. At simulated future

decision points, these alternatives are randomly assigned as before. A partial schedule is generated when all the operations that are currently pending have been assigned. Several of such partial schedules are required to evaluate the potential of an alternative.

The mechanism in generating simulated random schedules for the two cases are illustrated in Figure 3.2. The advantage of the partially exhaustive schedule is that it is computationally less expensive than the exhaustive approach.

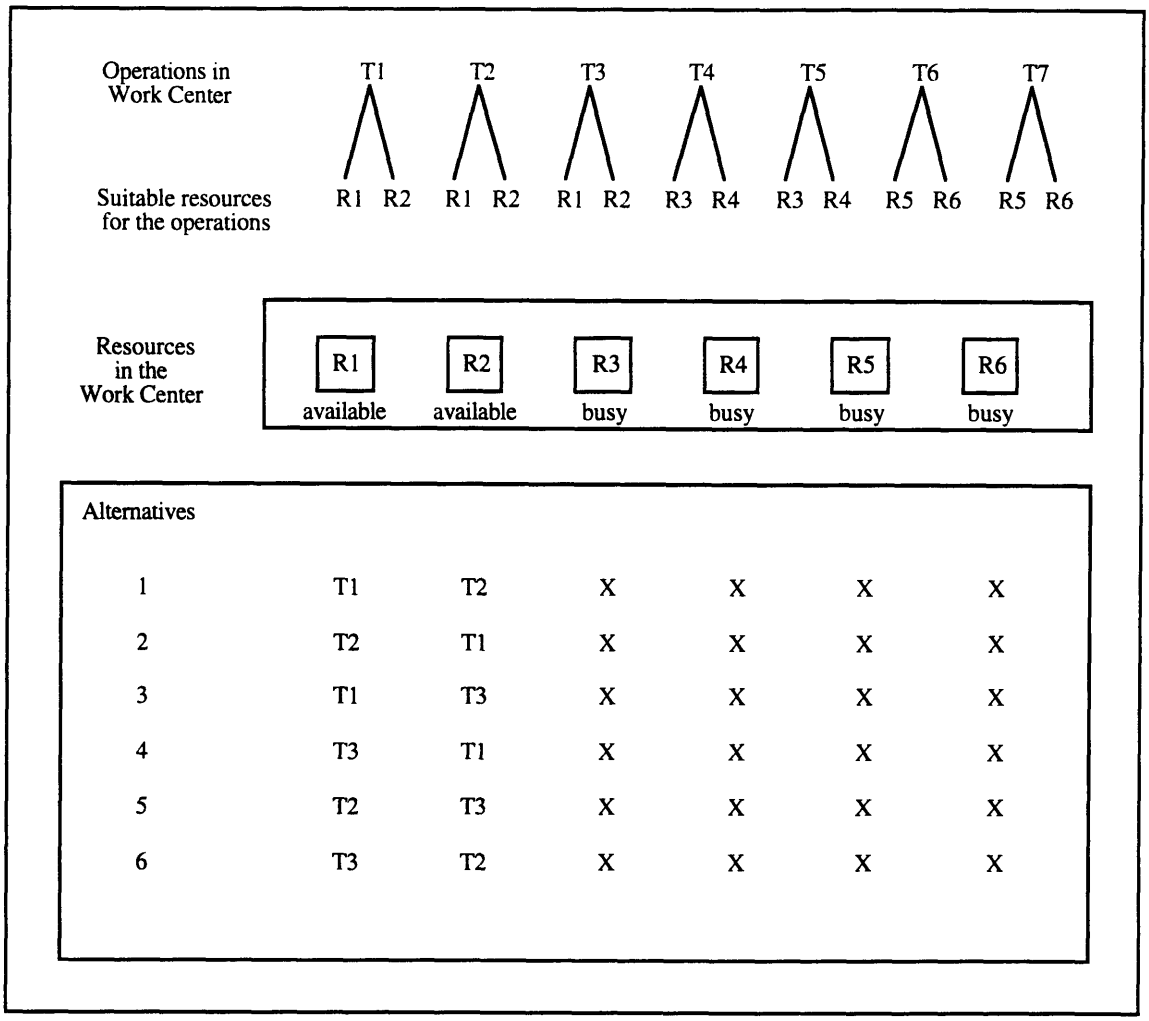


Figure 3.1: A Generic Scheduling Example

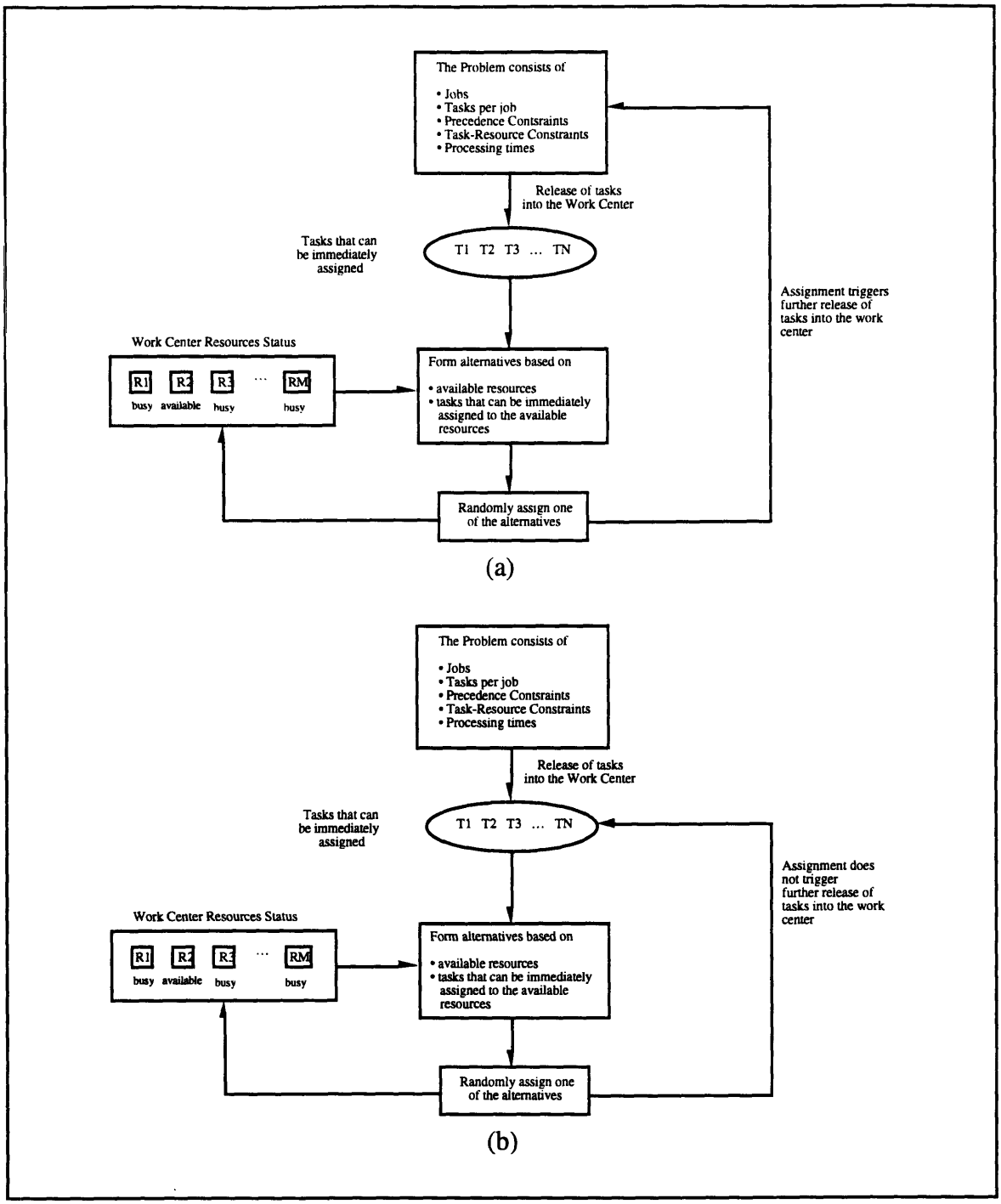


Figure 3.2: (a) generation of random schedules using the exhaustive approach
(b) generation of random schedules using the partially exhaustive approach

Now, if the potential of each alternative is perfectly determined, the resulting final schedule will be the optimal schedule. In practice, the potential cannot be perfectly determined. However, it can be estimated by utilizing the information in the distribution of

the quality of the simulated random schedules. The estimation procedure is based on the principles of statistics and Extreme Value Theory.

The Extreme Value Theory uses observed extremes from samples to forecast the extremes that are expected to occur in a future sample. The forecast is made based on the assumption that the initial distribution from which the extremes have been drawn remain constant from one sample to the next. Extreme value analysis is applicable to the tails of the distribution. The tail data of a distribution can be obtained in two ways. In the classical approach, sample extremes (maxima or minima) are used. In an approach proposed in [Castillo, 1988], only data at the tails of the distribution are used. In this presentation, we will limit ourselves to the classical approach.

The distribution of the tail can be described by several limit distributions (see Appendix A). In a broad sense, extreme value analysis involves the estimation of the parameters that describe the limit distribution [Guida and Longo, 1988, Hosking et al, 1985, and Csorgo and Mason, 1987]. The parameters could be estimated using two approaches. The first approach involves using system identification techniques, such as the maximum likelihood algorithm on the data. The other approach involves using graphical techniques which are simple to implement and result in quick answers. Graphical techniques involve plotting the probability against the extreme sample values of the random variable. A simple parametric graphical estimation technique is to locate r points on a curve visualized as passing through the data. The value of r corresponds to the number of unknowns [Bardsley, 1989, and Arnell et al, 1986]. Substituting the parameter estimates into the cumulative density function, gives a curve that passes through the specified points.

Consider an independent and identically distributed random variable, X (in the subsequent analysis, X is analogous to the quality of schedules). Assume that four extremes (maxima or minima) are obtained from random samples of size y, which are drawn from X and are ranked in ascending order, i.e., $x_1<x_2<x_3<x_4$. The probability of a value of X in the next draw from a sample of size y, being smaller than $x_1$, i.e. $P(X<x_1)$ is 1/5. Similarly one can show that $P(X<x_2)=2/5$, $P(X<x_3)=3/5$, $P(X<x_4)=4/5$ and $P(X>x_4)=1/5$. In general, if n extremes of an independent and identically distributed random variable is drawn from samples of size y, then $P(X<x_m)=m/n+1$ and $P(X>x_m)=(n+1-m)/n+1$, where m is the rank of the extreme drawn from X.

In production scheduling, one is generally interested in the extremes. Consider the case where the minimum value of X (e.g. cost) is of interest (the procedure described below can be easily modified to accommodate the case where the maximum solution is of interest).

As mentioned earlier, the distribution of the tails can be described by several limit distributions. Of these, the Gumbel distribution is the simplest and is used in the analysis described below. The Extreme Value Analysis basically consists of the following three steps:

Step 1: Rank the extremes. Plot the probability, p, against the ranked extremes. An example of such a plot is given in Figure 3.3. The probability is given as,

$$p = \frac{m}{n+1} \tag{3.1}$$

where m = rank of extreme
n = the number of extremes in the analysis.

Step 2: Transform the plot of Step 1 into a straight line using the Gumbel model. This involves transforming the probability using two logarithmic functions, as described in Eqn. 3.2. The example plot of Step 1 (Figure 3.3) is transformed using the Gumbel model and is presented in Figure 3.4.

$$\log(-\log(1-p)) \tag{3.2}$$

Step 3: Use the straight line of Step 2 to answer one of the following two questions:

*i* *how many additional drawings of X are required such that the minimum value of $x_1$ is reduced by an amount $\Delta$ ?*

The number of additional drawings is obtained by using the estimated line to predict the ordinate value $P^*_{\Delta}$, for an abscissa value of $x_1-\Delta$ (Figure 3.5). The ordinate value is then transformed back to the probability, $P_{\Delta}$, using the Gumbel model, where

$$P_\Delta = 1 - e^{-e^{-\beta^*\Delta}} \qquad (3.3)$$

The number of additional drawings required is then computed from the estimated probability,

$$P_\Delta = P(X<x_1-\Delta) = 1/(\alpha+n+1) \qquad (3.4)$$

where $\alpha$ is the additional number of drawings required for the minimum value of $x_1$ to be reduced to $x_1-\Delta$.

*ii* *what is the minimum value of X, if the total possible outcomes of X are finite?*

Similarly, the minimum value of X could also be estimated. The probability of achieving the minimum value of X, $P_{min}$, may be computed based on the number extremes (Eqn. 3.1). This probability is transformed using Eqn 3.2 to yield $P^*_{min}$. Using $P^*_{min}$ and the straight line of Step 2, an estimate of the minimum value of X is obtained (Figure 3.5).

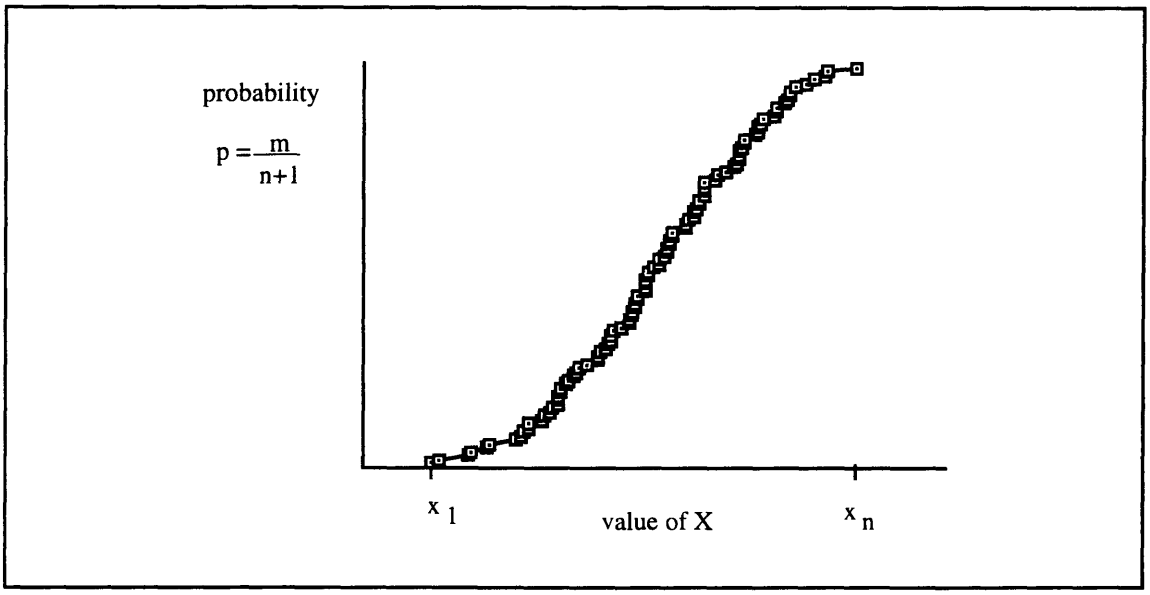


Figure 3.3: Plot of probability versus value of random variable X

The termination of the SEVAT procedure depends on the available computational resources. We have assumed that the computational resources required for the evaluation of a single alternative (a single drawing from X) is known. Based on this assumption, the resources required to evaluate the additional number of drawings to improve the current solution by $\Delta$, is computed. We have also assumed that the decision maker is influenced by a preference curve that determines the percentage of available resources that he/she is willing to expend for an expected improvement in the quality of the solution. For the purposes of our analysis, we have assumed that the preference function is linear (or risk neutral). To construct the preference function, we require the decision maker to state the value of the solution that he expects to obtain if he were to expend all his computational resources. The preference curve (or line) is then obtained as shown in Figure 3.6. We could also superimpose on this plot the estimated curve representing the variation in the improvement of the solution with computational resources. The intersection of these two plots indicates the point beyond which additional sampling would not be desirable. To the left of this point, the improvement in the solution is larger than the assumed preference of the decision maker and therefore further sampling should continue. To the right of the intersection point, the expected improvement in the solution is smaller than the desired improvement of the decision maker and this indicates that further sampling should be discontinued.

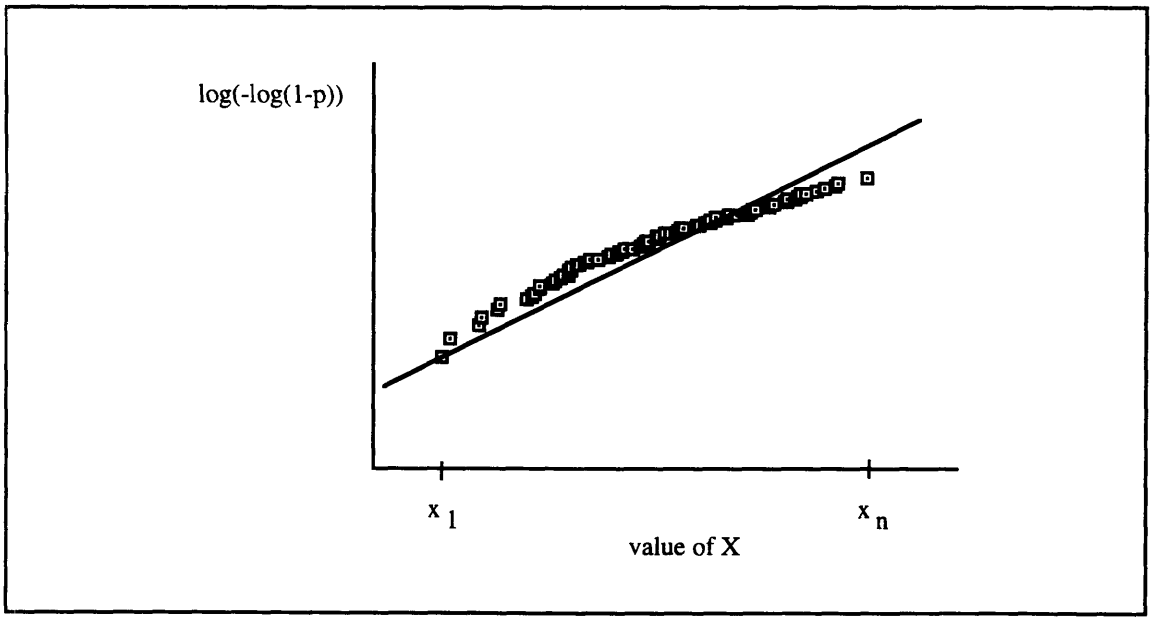


Figure 3.4: The plot of Figure 3.3 is fitted to a Gumbel distribution

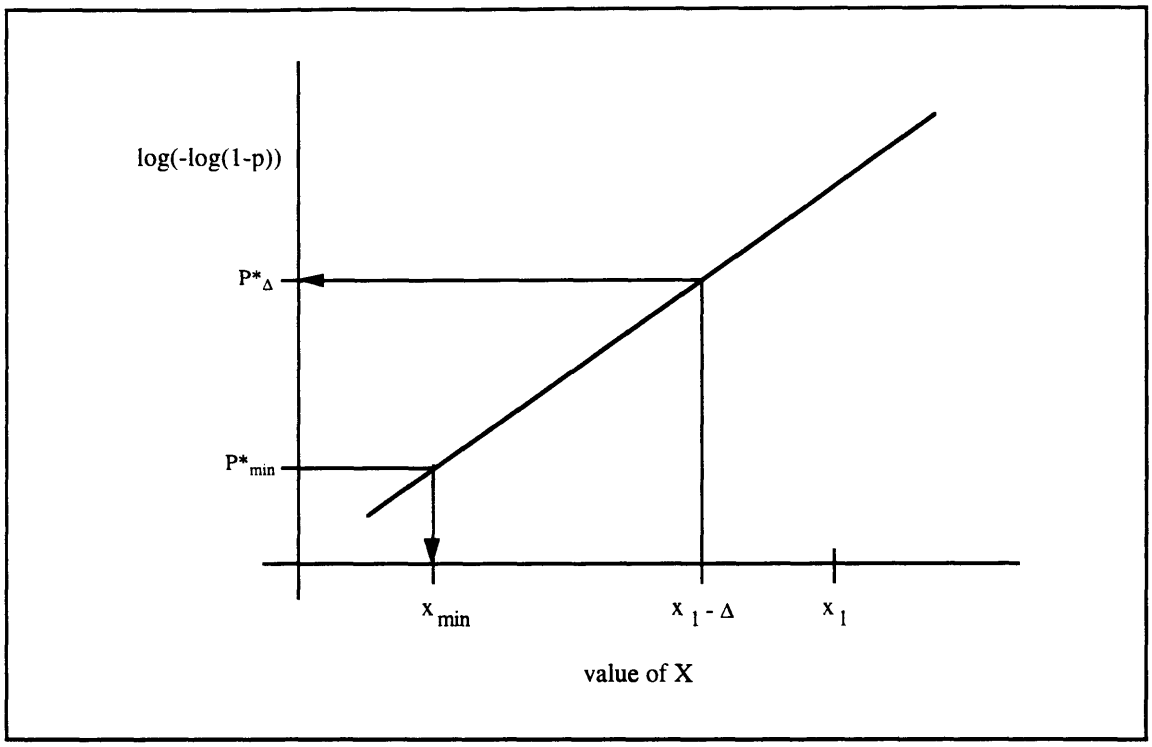


Figure 3.5: Predicting the i) the additional number of drawings required to reduce the minimum value by an amount Δ and ii) the minimum value of X

## 3.3 A Job Shop Problem

To illustrate the usefulness of the SEVAT procedure, this procedure is applied to standard testbed problems, commonly cited in the literature. These problems are referred to as the Muth and Thompson 6x6 and 10x10 problems. The 6x6 problem is a static problem in which the job shop has 6 machines and initially (at time=0), 6 jobs are pending at the job shop. Each job has 6 operations, and each operation can be processed by a single machine only. The precedence constraints, machine-operation requirements and the processing time for each operation are summarized in Figures B.1 and B.2 in Appendix B. Similarly, the 10x10 problem involves 10 machines, and has 10 jobs pending initially at the job shop, and each job has 10 operations. The relevant data for the 10x10 problem are summarized in Figures B.3 and B.4 in Appendix B.

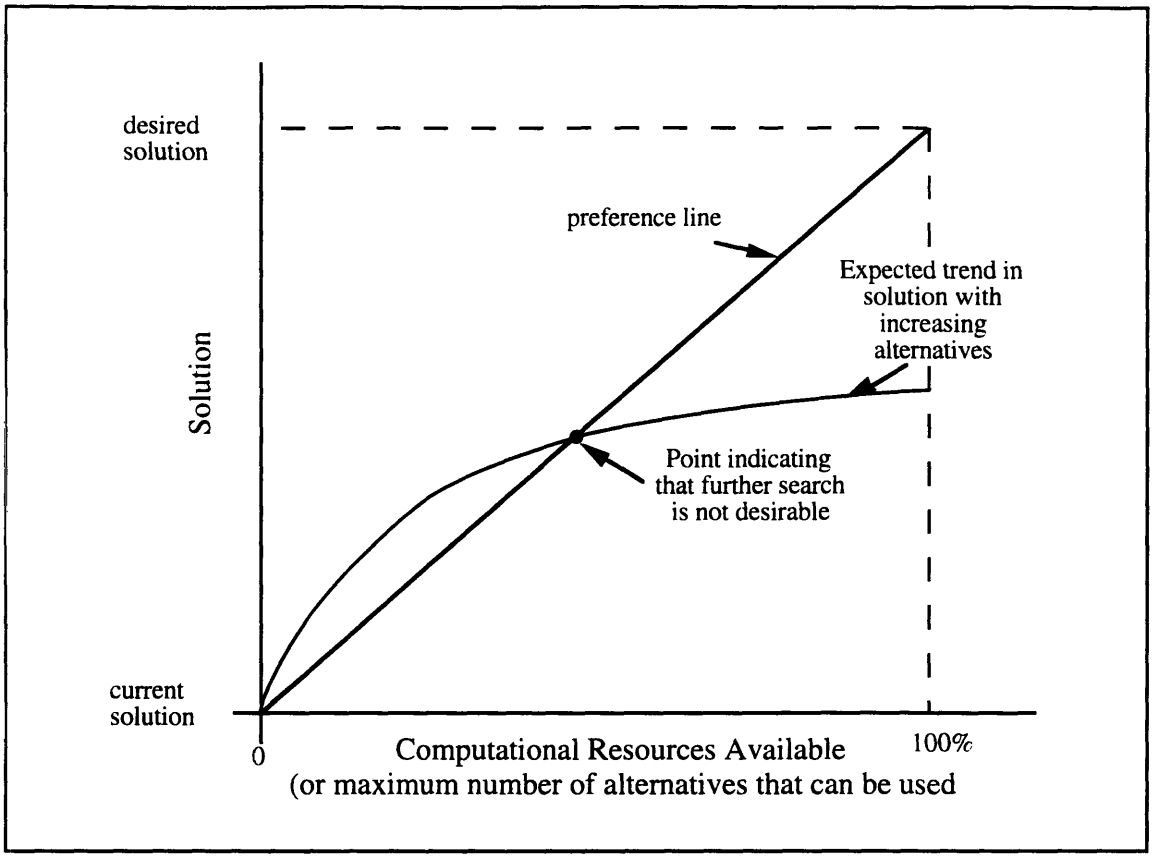


Figure 3.6: Decision Maker's trade-off between computational resources and quality of solution

The following are more specific assumptions regarding the Job Shop problem.

- There are N jobs, each with multiple operations that are available for processing at the start of the schedule (i.e. at time zero).

- Each resource is assumed to be operational at all times, i.e., delays such as resource breakdowns will be neglected.

- The setup times for all operations are not dependant on the sequence of the jobs and are assumed included in the processing time of the operation.

- Each operation can be performed on one or several alternative resources.

- Once a operation is performed on a resource, that operation must be completed before the next operation is to be performed.

- Non-delay: no resource is deliberately kept idle in the presence of a waiting operation.

As stated earlier, a decision point occurs when a resource or several resources becomes available. An alternative at a decision point is the possible assignment of pending operations to the available resources. Typically, there are usually a large number of pending operations and available resources, in which case, the number of alternatives becomes extremely large. For example, in the hypothetical case where there are 20 pending operations and 10 available resources, and assuming that all the operations can be performed on all the resources, then there are $6.7 \times 10^{11}$ , (20!/(20-10)!) alternatives. In the general case, however, not all the pending operations can be performed on all the machines, and therefore the number of alternatives calculated above would be an upper bound. Nevertheless, the number of alternatives grow exponentially as the number of pending operations increase.

The evaluation of all possible alternatives in a decision theoretic framework, will therefore become computationally burdensome. Since, the objective of the scheduling strategy is not to seek the optimal schedule, but rather to seek a "good" schedule, the evaluation of all the possible alternatives at a decision point is therefore not necessary. An alternative approach, would be to adopt a "simple random sampling" strategy. In this strategy, unbiased samples are drawn from the population of alternatives at each decision point. The question then arises, "How many alternatives should one draw at each decision point?".

An approximate answer to this question is obtained by incorporating the extreme value analysis into the alternative selection process. This process requires the estimation of the "potential" of an alternative (Figure 3.7) and begins by randomly sampling a small number of alternatives, say 10. The potential of these alternatives are estimated (the estimation process will be discussed later). Using these potential values, the alternatives are ranked and the extreme value analysis is applied to the ranked data. The further number of sample alternatives expected to improve the "current best" alternative's potential by say 10% is estimated from extreme value analysis. If the number of further alternatives required is larger than a preset limit, then, the process will terminate as adequate alternatives have been sampled. The preset limit can be varied according to the computational time that is available. If the number of further alternatives required is smaller

than the preset limit, these additional alternatives are drawn from the population of alternatives and the process as described earlier is continued (Figure 3.7).

The next part of the problem is to examine how the potential of the alternatives should be estimated. Once again, extreme value analysis could be used for this purpose and the process is described in Figure 3.8.

At the current decision point, the alternative whose potential is to be estimated, is tentatively assigned to the available resources. The scheduling process is simulated, and at each future decision point in the simulation, new alternatives are formed based on the pending operations and the available resources. One of these alternatives is randomly assigned to available resources. As discussed earlier, one could allow the simulation to progress until a complete schedule of all currently pending jobs and their respective operations are assigned to the resources (exhaustive schedule). Alternatively, one can generate a partially exhaustive schedule in which the simulation is progressed without triggering further release of operations into the work center, and generating a schedule until all the currently pending operations at the work center have been assigned. The performance of the randomly simulated schedule (be it exhaustive or partially exhaustive) is calculated and this calculation depends on the performance measures of interest, i.e, tardiness, quality, cost or flowtime.

Several such random schedules, say 100, are generated and divided into groups of say, size 10. The schedule with the best performance from each group is identified and Extreme Value Analysis is applied to these performance values. From the Extreme Value Analysis, one can estimate the expected performance measure, assuming that the number of random schedules generated were doubled. This expected performance measure is the potential of the alternative.

The overall scheduling strategy using Extreme Value Analysis is summarized in Figure 3.9. The strategy may be viewed as having two decision loops. In the outer loop, Extreme Value Analysis is used to ensure adequate alternatives are sampled and in the inner loop, Extreme Value Analysis is used to evaluate the potential of each alternative.

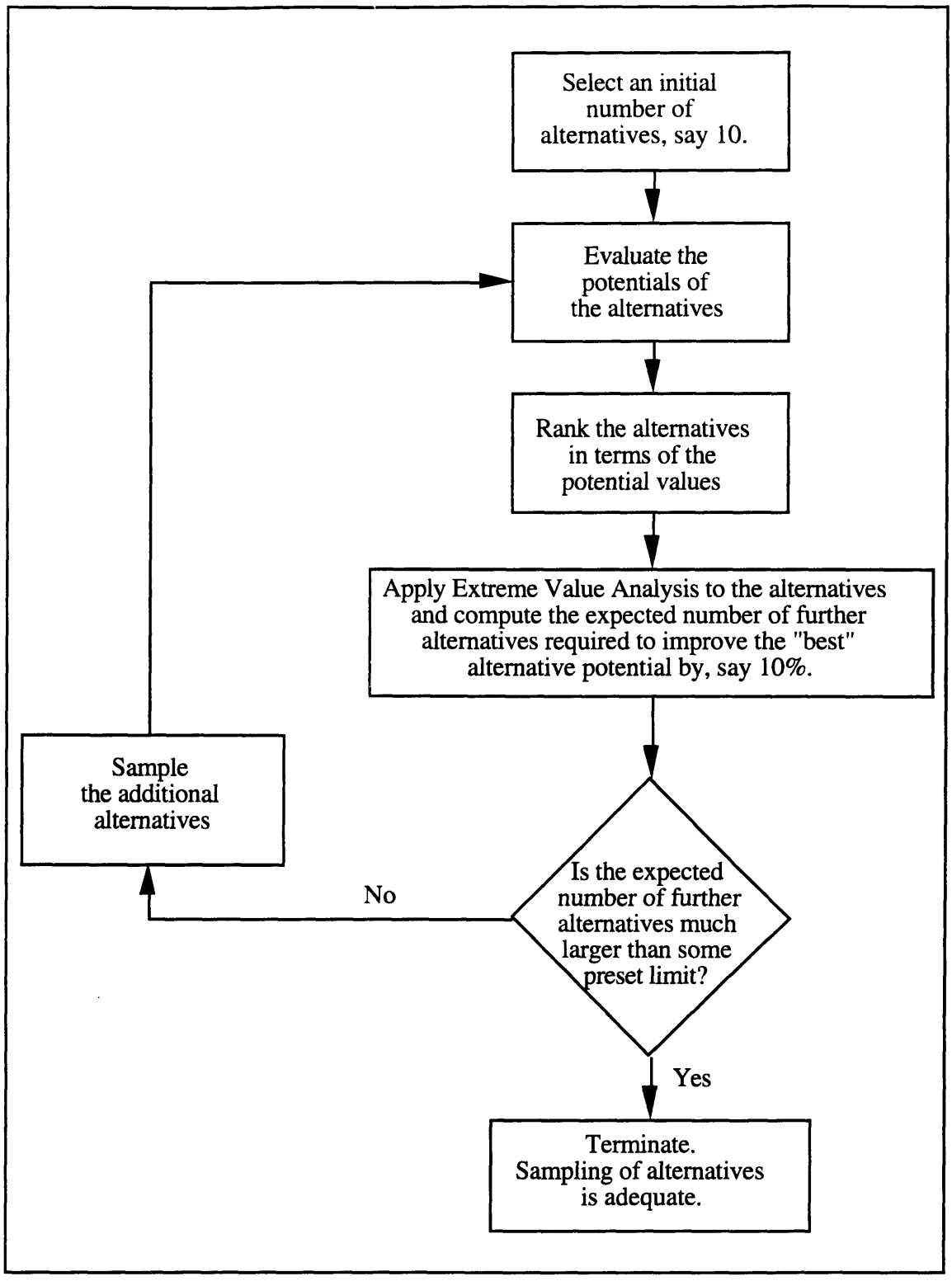


Figure 3.7: Summary of the Alternatives selection process

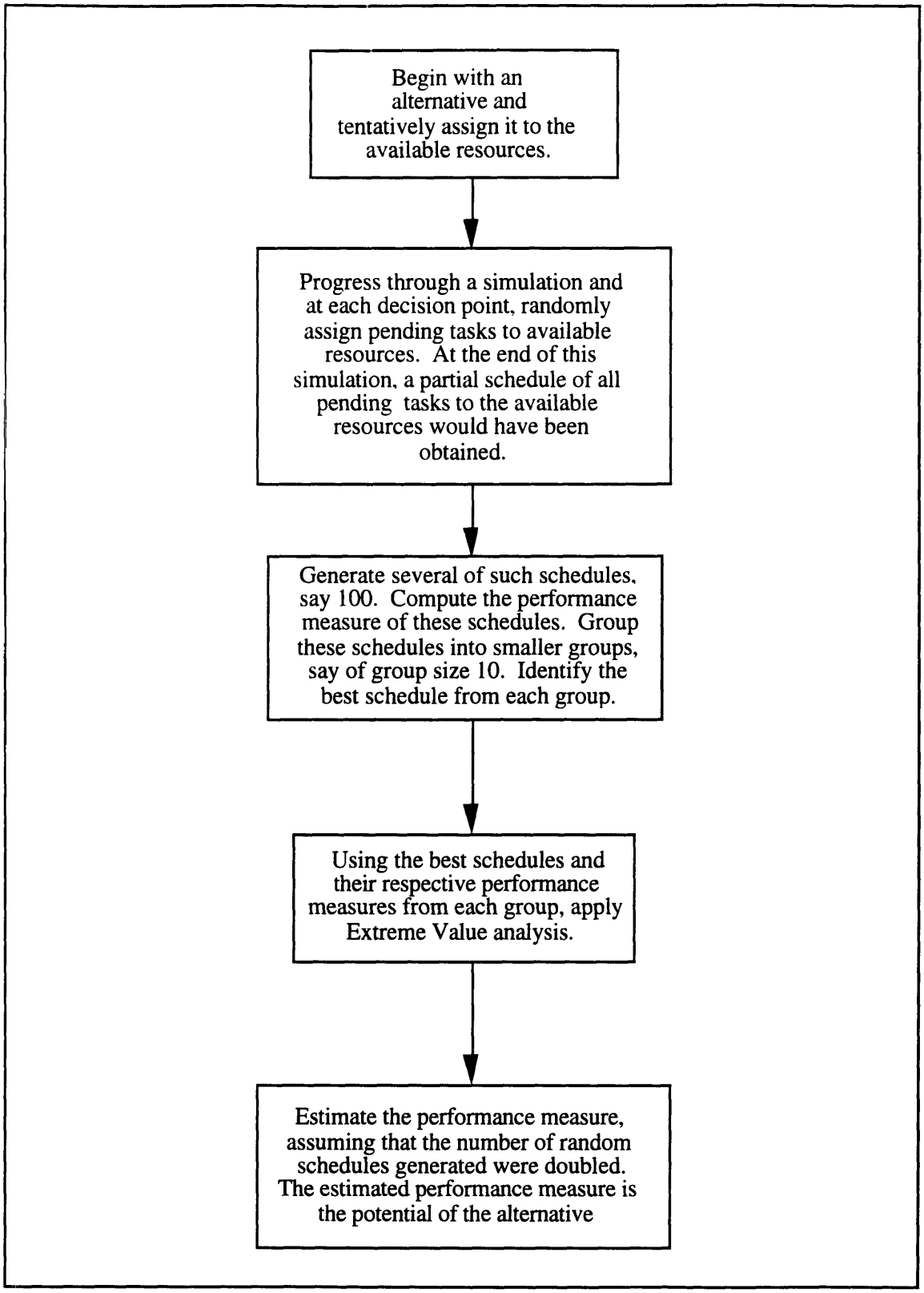


Figure 3.8: Estimation of the Potential of an Alternative

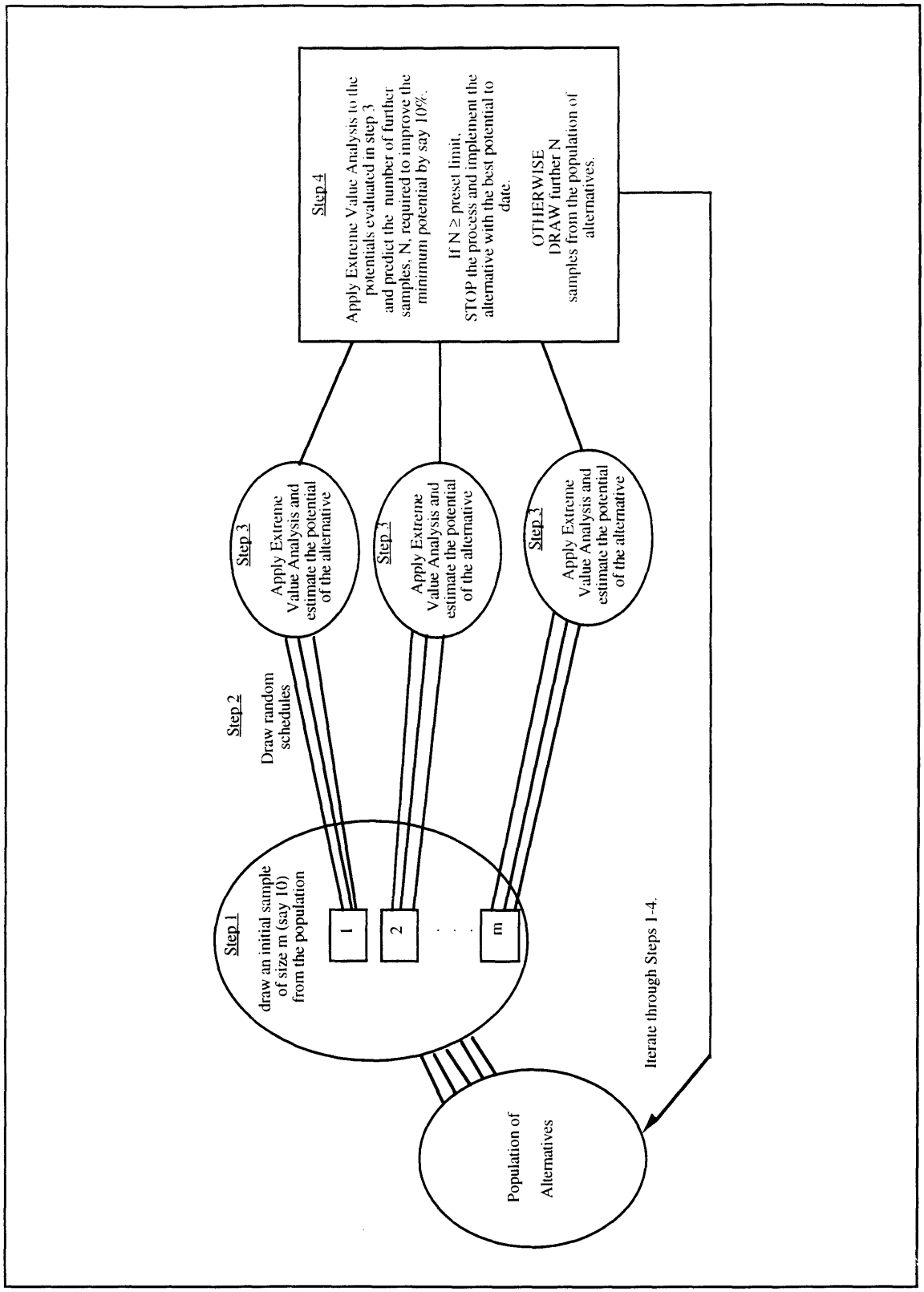


Figure 3.9: The "Dual-loop" Extreme Value Scheduling Approach

## 3.4 Results and Discussion

The Muth and Thompson 6x6 and 10x10 static job shop scheduling problems require the minimization of the makespan performance measure. The makespans obtained using the SEVAT procedure, some common dispatching rules (such as the Shortest Processing Time (SPT), Longest Processing Time (LPT), First In First Out (FIFO), Last In Last Out (LIFO) and the Random rule) and the Fuzzy Logic Approach discussed in Chapter 2, are summarized below in Figure 3.10.

| Method | Makespan (6x6 problem) | Makespan (10x10 problem) |
|---|---|---|
| Optimal | 55 | 930 |
| SEVAT | 57 | 988 |
| SPT | 88 | 1074 |
| LPT | 67 | 1197 |
| FIFO | 74 | 1259 |
| LIFO | 72 | 1223 |
| Random | 60 | 1106 |
| Fuzzy | 71 | 1134 |

Figure 3.10: Makespans obtained for the Muth and Thompson 6x6 and 10x10 static job shop scheduling problems

The optimal makespan for the 6x6 and 10x10 problems are 55 and 930 respectively [Vancheeswaran and Townsend, 1993]. The makespans obtained using the SEVAT procedure compares favorably with other approaches cited in [Vancheeswaran and Townsend, 1993].

The Muth and Thompson 6x6 and 10x10 job shop scheduling problems require the minimization of the makespan performance measure, and as such do not require multiple criteria decision making. Multiple criteria could be incorporated in the decision making process by normalizing the performance measures and obtaining the weighted sum of these normalized values (also known as the utility). For example, let us assume that the performance measures of interest are, namely, cost and quality. Since minimization of cost and maximization of quality is desired, the normalization of these performance measures will be different. A simple linear normalization could be used as described below:

- Cost: $$1 - \frac{\text{Cost - minimum Cost}}{\text{maximum Cost - minimum Cost}} \quad (3.5)$$

- Quality: $$\frac{\text{Quality - minimum Quality}}{\text{maximum Quality - minimum Quality}} \quad (3.6)$$

The maximum Cost and Quality and the minimum Cost and Quality are obtained from the random samples at each decision point. The utility of an alternative is calculated as follows:

$$\begin{aligned} \text{utility} = {} & (\text{Normalized Cost} * \text{Weight of Cost}) \\ & + (\text{Normalized Quality} * \text{Weight of Quality}). \end{aligned} \quad (3.7)$$

The Weights of Cost and Quality are values that range from 0 to 1.0 and the sum of these weights must add to 1.0.

# Chapter 4
# Job Shop Scheduling Using Genetic Algorithms

In this chapter, an artificial search method that mimics the natural evolution of organisms, namely Genetic Algorithms, will be discussed and developed to solve job shop scheduling problems. Unlike the schedule building approach of the SEVAT procedure described in Chapter 3, this method is a schedule permutation approach. Feasible candidate schedules are usually generated randomly and then permutated to produce a "good" schedule. A brief survey of previous Genetic Algorithms approaches to scheduling will be presented followed by a short discussion on premature convergence. As in Chapter 3, this method will be applied to the Muth and Thompson benchmark problems and the results compared with some common dispatching rules and the Fuzzy Logic approach described in Chapter 2. The application of the Genetic Algorithms approach to dynamic job shop scheduling will be explored in Chapter 5.

## 4.1 Introduction

The basic idea behind Genetic Algorithms is to produce a pool of solutions, from which the worst solutions are eliminated and the rest are modified. This process of elimination and modification continues until a good solution is obtained.

The search of a complex solution space involves the tradeoff between exploiting good solutions and exploring the search space. Genetic Algorithms has been said to strike a balance between exploration and exploitation. The basic assumption of Genetic Algorithms is that the best solutions will be found in regions of the search space containing relatively high proportions of good solutions and that these regions can be identified by robust sampling of the search space [Booker, 1987]. However Genetic Algorithms has also been criticized for rapidly locating the region in the search space where the global optimum resides, but does not locate the optimum with similar speed [De Jong, 1992].

Genetic Algorithms mimic population genetics and assume the following:

- Individuals (also referred to as chromosomes or genotypes) are fixed length strings having finite number of possible values or alleles at each position.
- A population contains a finite number of individuals.
- Each individual has a fitness.

The general mechanics of Genetic Algorithms is described in Figure 4.1. The process begins with the random generation of individuals that form the population. At each iteration, the fitness value of each individual in the population is evaluated. The fitness value is usually the value of the objective function to be "optimized". A genetic operator termed "Reproduction", then stochastically reproduces members of the current population according to their relative fitness values. Variation to the chromosomes are realised by two other genetic operators, viz, "Crossover" and "Mutation". The resulting chromosomes replace the chromosomes of the previous population to yield the next generation and the above procedure is repeated iteratively, and newer generations are reproduced. The driving force behind Genetic Algorithms, is the reproduction of individuals in proportion to fitness together with the crossover operator. The mutation operator is only a background operator that guarantees that no allele will be permanently lost from the population [Booker, 1987]. Termination of the iterative process of the Genetic Algorithms occurs when a maximum number of iterations have been reached or when there is no significant improvement in the fitness of the population in successive generations.

The implementation of Genetic Algorithms depends on several parameters/operators. The most preferable values for these parameters/operators has been attempted through the use of another Genetic Algorithms, by defining 19 components [Freisleben and Hartfelder, 1993]. On further inspection, these 19 components can be broadly classified into the following parameters/operators [Prinetto et al, 1993] :

- Coding Strategy
- Population
- Crossover
- Mutation
- Parents Selection
- Local Improvement

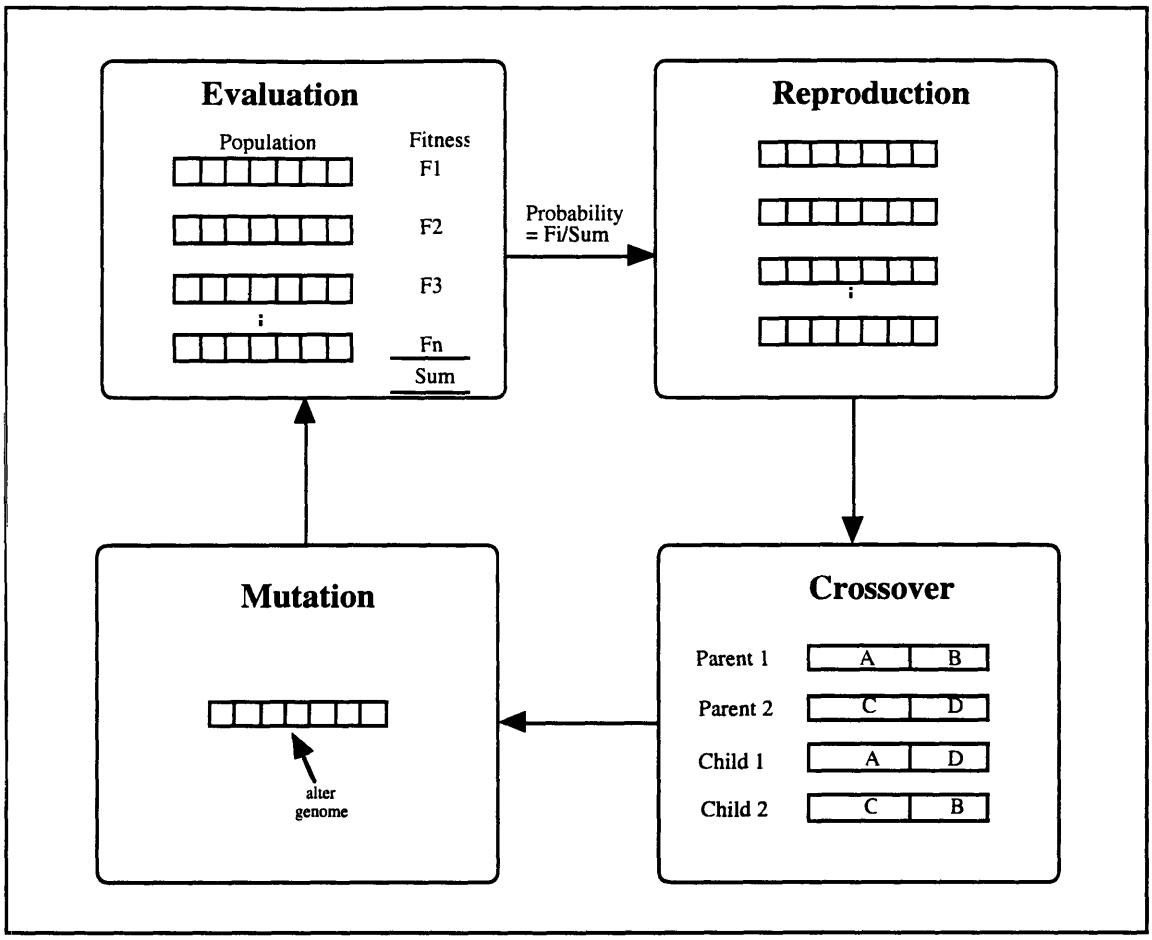


Figure 4.1: A Generic Genetic Algorithm

### 4.1.1 Coding Strategy

Usually the most difficult task in applying Genetic Algorithms to any problem is to represent the solutions as fixed length strings. In the case of job shop scheduling, the representation of schedules is usually achieved through the use of binary strings. Binary strings for Genetic Algorithms was thought to be a maximally simple representation - that it maximizes the schemata sampled per individual of a population. This has been shown to be not true. Expressive representations provide a more powerful apparatus for adaptation [Antonisse, 1989].

The choice of binary representation for schedules allow for the use of the traditional crossover and mutation operators. However, the representation of the schedule itself becomes complex. For example, in [Nakano and Yamada, 1991], the schedule is

represented as a bit vector for every job pair. Each bit in the vector compares an operation of one job to the operation of the other job for each of the machines in the job shop. A value '1' is assigned if the operation of the first job is performed prior to the operation of the second job, and a '0' otherwise. This kind of representation requires MN(N-1)/2 bits for N jobs and M machines. Therefore for a reasonable sized problem of 100 jobs and 20 machines, this will require 99000 bits and this form of representation grows exponentially with the number of jobs. This kind of representation also implicitly assumes that the job routings are completely fixed and does not accommodate alternate job routings.

In addition to making the representation complex, the binary representation of schedules also creates another problem. Since the precedence constraints have to be observed in a job shop schedule, the use of traditional crossover and mutation operators result in schedules that violate the precedence constraints. Therefore, to satisfy the precedence constraints, an additional operator is required that makes 'illegal' schedules (schedules that violate the precedence constraints after the crossover and mutation operations) legal.

On the other hand, the choice of a more expressive schedule representation (symbolic representation), makes the representation direct and simple. However, new crossover and mutation operators that are usually problem specific, have to be designed. Once these operators are designed, there is no need for an additional operator to ensure that schedules are legal. In this thesis, the symbolic representation of schedules will be adopted.

### 4.1.2 Population

Any implementation of the Genetic Algorithms must be careful that the initial population contains an adequate pool of alleles for each position. It might be equally important that there be an adequate initial pool of schemata. If the initial distribution of points does not uniformly sample the space, crossover may never get the opportunity to direct the search into unrepresented regions. The commonly used random number generators are very poor when it comes to distributing samples randomly in more than one dimension. Random number generators having this capability are very expensive computationally. A good compromise is to produce initial populations as usual, then subject them to repeated crossing over with uniform random pairing. If this is done to the

point of stochastic equilibrium, we can be assured that the initial pool of schemata will be as robust as possible [Booker, 1987].

The population is often chosen in the range of 20 to 100. Very small populations cannot establish or maintain sufficient diversity, while in the large population good strings may get swamped by numerous inferior strings, or close copies of themselves. Similarly, while the mutation rate should be sufficient to prevent stagnation in the population, it must still be low enough that the propagation of schemata is not unduly hindered [Cartwright and Mott, 1991].

### 4.1.3 Crossover

Crossover is the mechanism through which promising schemata ('genetic material') of the good individuals are propagated through the generations of individuals. The traditional crossover operator is described as in Figure 4.2. In this operator, two parents produce two offsprings. A random point is selected in the individual and the two parents are dissected at this point. Either the preceding or the trailing segments of the dissected individual are interchanged.

Traditional crossover as defined in Figure 4.2, may not always work for all problems. Usually, crossover operators are designed for specific problems. For example, in a textile machinery scheduling problem [Filipic, 1992], only one offspring is produced for every two mating parents. The crossing sites are selected in the same way as with multiple-point crossover, and one of the parents is selected to play a more dominant role during mating. The alternate sections of the genes are copied directly from the dominant parent to the offspring. The genes that are to be contributed by the other parent is checked in turn for consistency. If the gene value keeps the partially built schedule legal, it is passed on to the offspring, otherwise it is randomly altered to make the partially built schedule legal. This approach of randomly selecting gene values also acts as the mutation operator.

Several crossover operators are discussed in [Cleveland and Smith, 1989]. These operators were designed for the sector scheduling problem, which refers to the automated subcomponent of an actual computer board assembly and test facility. These operators can also be used in job shop scheduling. Some of these crossover operators are:

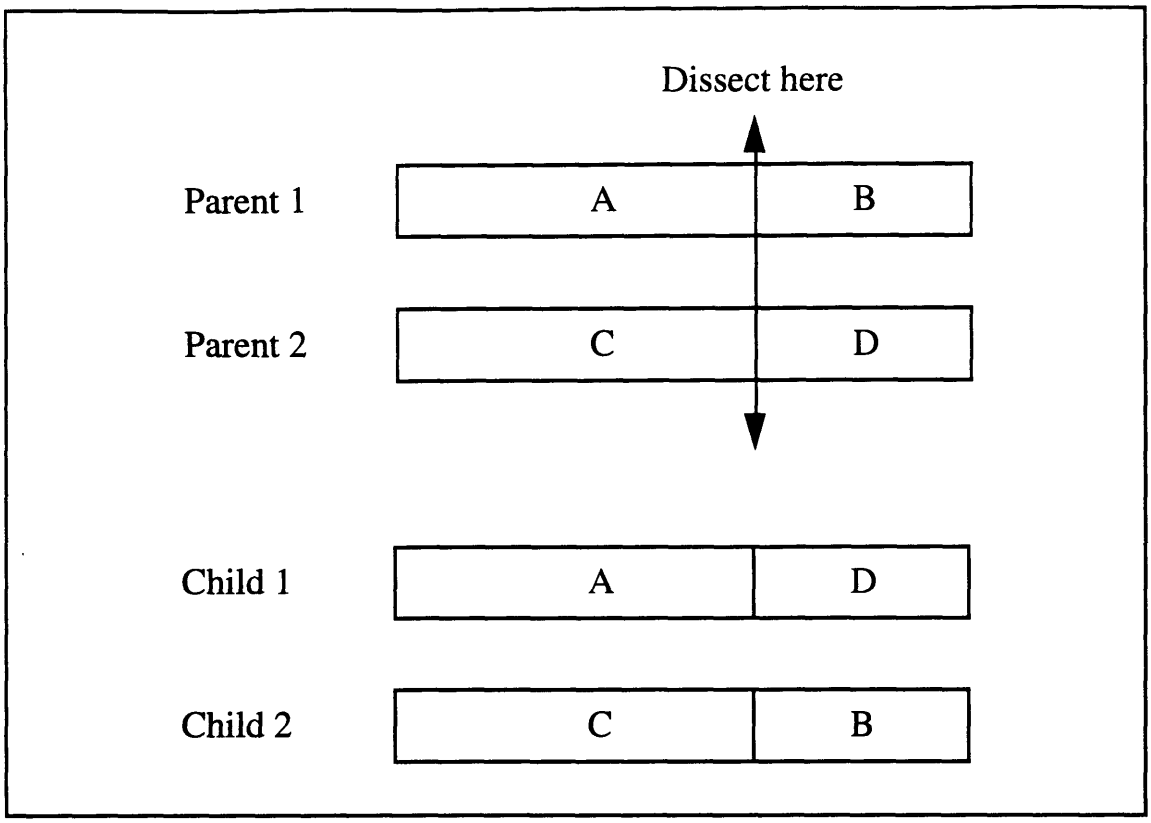


Figure 4.2: The Traditional Crossover Operator

- The PMX crossover operator ensures legality of the solutions it produces by choosing an interval on each of the solutions, swapping the intervals (probably creating an illegal solution), and then mapping the selected intervals within each solution (restoring the legality of each).

- The subtour-swap operator, works by alternately selecting segments from the two parent solutions and incorporating those into the offspring solutions.

- The subtour-chunking operator works by selecting chunks in as much the same way as segments are selected by the subtour-swap operator. In this case, however, the chunks are placed in the child solution in approximately the same position that they occupied in the parent solutions. Conflicts are resolved by trimming the chunks and sliding them to the right and left to make them fit.

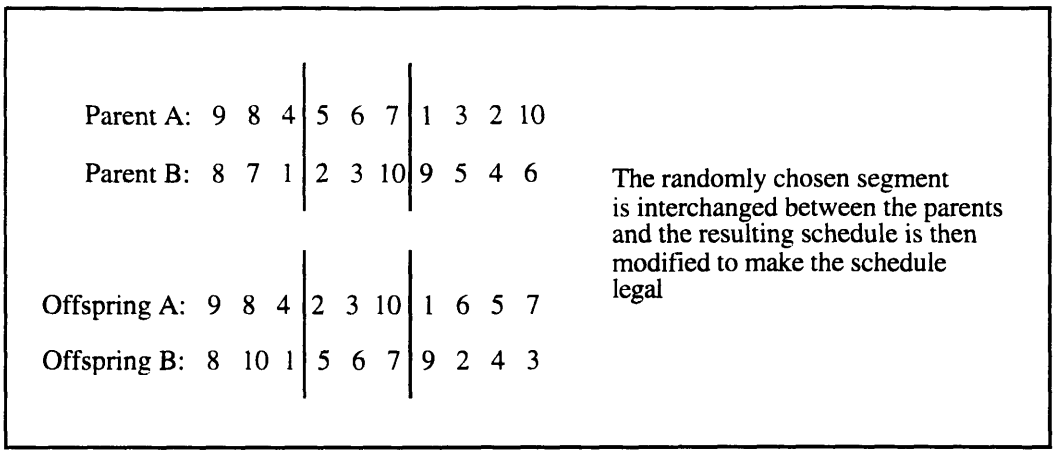


Figure 4.3: The PMX crossover operator

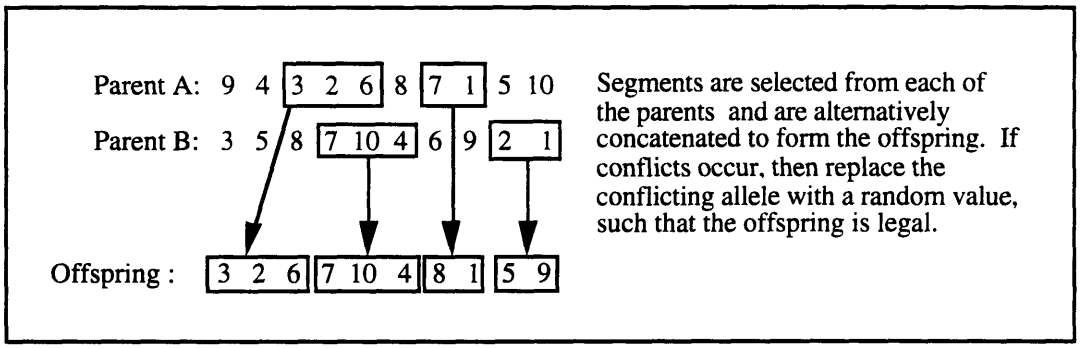


Figure 4.4: The Subtour-Swap operator

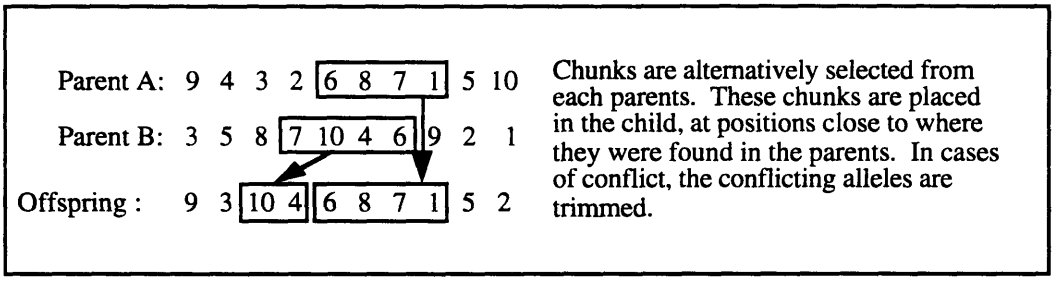


Figure 4.5: The Subtour-Chunk operator

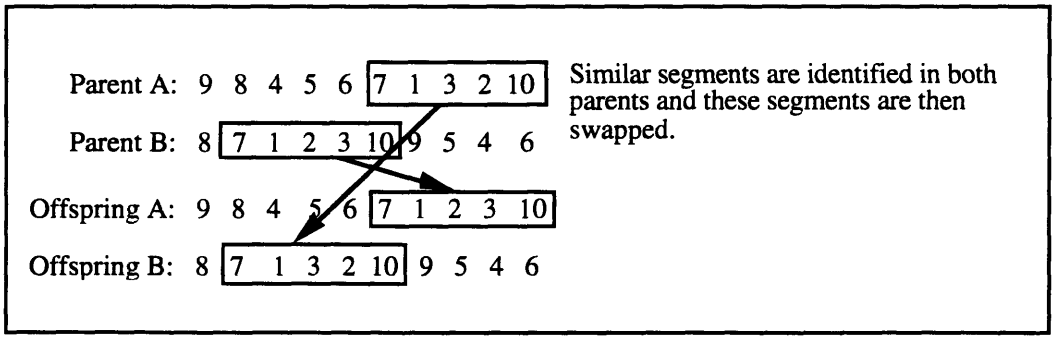


Figure 4.6: The Subtour-Replace Operator

- The subtour-replace operator examines the two solutions to be crossed for similarly located subsequences that have the same elements (and length of at least two). When two such subsequences are found, in the respective parents, they replace each other in the solutions.

- The weighted chunking operator is exactly like the subtour-chunking operator, but it also incorporates a bit of domain knowledge - e.g. due dates. When two chunks collide in the child, one of the chunks is moved around the other and this decision is made according to the due dates.

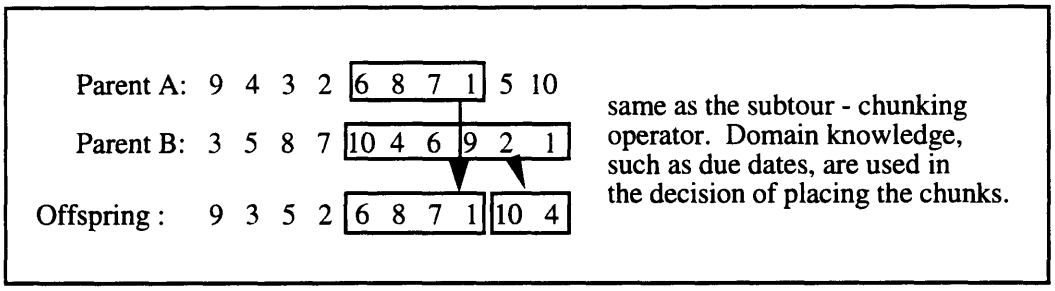


Figure 4.7: The Weighted Chunking Operator

The one-point, two-point and uniform crossover operators are often referred to in the literature. These crossover operators can be distinguished by using the crossover mask [Syswerda, 1989], which consists of a string of bits, the same size as the chromosomes to be crossed. The parity of each bit in the mask determines, for each corresponding bit in a child, which parent it will receive that bit from. The mask-based crossover operations are the same for each of the three different crossover operators: The difference lies in the masks. Each type of crossover mask has a characteristic pattern (Figure 4.8)

| | |
|---|---|
| one point crossover : | 11110000 |
| two point crossover: | 00111100 |
| uniform crossover : | 01001101 |

Figure 4.8: Crossover masks for various crossover operators.

The 1-bits in the masks for the one-point and two-point crossovers are contiguous. With Uniform crossover, the 1-bits are uniformly distributed throughout the chromosome, occuring at each position with a chance of 0.5.

Another crossover operator, the order crossover, is described in [Braun, 1990]. The order crossover operator involves two parents to produce an offspring. A crossover point is randomly chosen and the first parent string is divided into two substrings. The elements of the second substring of the first parent are removed from the second parent string. The concatenation of the first substring of the first parent and the remaining substring of the second parent constitutes the offspring.

The order crossover operator was modified to produce a more general order crossover operator. The first parent is again divided into two substrings, except now, the crossover point is restricted to occur much closer to the starting point of the parent string. The first substring like before is appended to the offspring and elements of the first substring are removed from both parents. Next the procedure is repeated with the second parent (divided into 2 substrings and the first substring is appended to the offspring and elements of the substring are also removed from both parents). Substrings from the two parents are alternatively extracted and appended to the offspring. This process is repeated until the offspring is created.

Multi-point crossover has been reported in the literature to be very disruptive. However, computational experiments show that for very small populations, the use of uniform crossover is advantageous, whereas with larger population sizes, the 2-point crossover approach is more advantageous than uniform crossover [De Jong and Spears, 1989].

### 4.1.4 Mutation

Mutation is the mechanism in which untried schemata or schemata that have been lost due to the crossover operation are introduced into the population. This genetic operation ensures that newer search areas are introduced into the search process. Traditional mutation using the binary representation involves simply swapping the binary genome to its complementary value, i.e. changing a '1' to a '0' and vice versa.

Like crossover, mutation operators usually become problem-specific if the symbolic representation of individuals is adopted. The mutation operator is not applied to every individual in a generation. The mutation operation is made to occur very rarely in a

population. The mutation rate of individuals are usually assumed constant over generations. Varying mutation rates over generations and/or across the gene representation of the individuals was found to improve the performance in a Genetic Algorithms application of minimizing the stackloss over a range of industrial burners [Fogarty, 1989].

The optimal mutation rate is proportional to the chromosome. Search by mutation and crossover are complementary. The probability that mutation will give better individuals decreases with the number of alleles that are correct, whereas, the probability that crossover produces a better individual increases with the number of correct alleles [Muhlenbein, 1992].

### 4.1.5 Parents Selection

This section refers to the mechanism of reproduction. Most Genetic Algorithms use proportionate selection as a reproduction strategy. In this strategy, the darwinian notion of 'survival of the fittest' is strictly adhered to. Each individual in the population is given a chance of representing itself in the next generation. However, the probability of its representation is proportional to its fitness and one individual may be represented more than once in the next generation. This selection strategy is sometimes also referred to as the 'roulette wheel' strategy.

Some other selection strategies that are commonly used in Genetic Algorithms are are [Thierens and Goldberg, 1994]:

- Tournament Selection
  A set of individuals are randomly chosen and the best is picked for reproduction. The number of individuals in the set is usually two but larger tournament sizes can be used to increase the selection pressure.

- Truncation Selection
  This is also know as block selection. All individuals are ranked according to their fitness and the best are selected as parents. A threshold, T, is defined such that T% best individuals are selected.

- Elitist Recombination

  In this selection strategy, the selection and recombination phases are intertwined. Competition for survival takes place at the level of each family - the mating parents and their offsprings. For every mating pair, two offsprings are created and the best two of these four individuals go to the next generation, so individuals can only be replaced by other individuals with a higher fitness value.

When all the parents are deleted from the pool, and replaced by the offsprings, this is known as generational replacement [Polant, 1992].

In the discussion so far, we had assumed that the genetic operators are applied to the entire population. Two other models, viz, the island model and the neighbourhood model, are also commonly used in Genetic Algorithms [Muhlenbein, 1989]. In the island model, subpopulations are isolated, and sometimes immigrants from the other islands arrive. In the neighbourhood model, isolation by distance is enforced. The neighbourhoods of different individuals overlap and selection is performed only in the neighbourhood.

One of the problems with fitness proportionate reproduction is the "scaling problem". For example let us assume that the Genetic Algorithms is applied to a maximization problem where the fitness varies between 100 and 1100 with an average of 550. The selective pressure toward the top ranked candidate would be 2.0 (1100/550). Let us suppose that this is sufficient selective pressure and the search progresses and in the later stages, the range of fitness may vary from 1000 to 1200 with an average of 1100. Now the selective pressure is 1.09 (1200/1100), which may not be adequate and the search may stagnate. The use of rank-based reproduction eliminates the "scaling problem." Other parameters that affect the selective pressure include [Whitley, 1989] :

- population size
- crossover rate
- mutation rate
- generation gap
- scaling window
- selection strategy.

The value returned by an evaluation function should not be considered as an "exact" measure of fitness. The exact value returned can vary greatly depending on how the function was implemented. Allocating reproduction according to rank prevents scaling problems [Whitley, 1989].

### 4.1.6 Local Improvement

Genetic Algorithms have been labeled to be unsuited for fine tuning solutions that are close to optimal. Local improvement operators, when added to the recombination (crossover) operation, helps overcome the above mentioned shortcoming [Ulder et al, 1990]. The local search algorithm chosen for this purpose should be one that meets the time capacity constraints. In the case of severe time restrictions, one can still use a truncated version of the local search algorithm, such that it goes through only a small number of iterations. Genetic Algorithms that incorporate heuristics about the problem into the recombination operators (crossover and mutation) are also referred to a Heuristic Genetic Algorithms [Jog et al, 1989].

## 4.2 Scheduling Using Genetic Algorithms

In this section, a brief survey of the attempts to apply Genetic Algorithms to scheduling problems will be discussed. This literature survey is by no means exhaustive; it is presented as a representative discussion of the various Genetic Algorithms strategies adopted and reported in the literature. At the end of this section, a table summary of the various Genetic Algorithms scheduling approaches is presented in terms of 4 dimensions, viz, the problem investigated, individual representation, crossover operator and mutation operator (Figure 4.9).

In [Mattfeld et al, 1994], the standard job shop problem is used to illustrate the implementation of a Genetic Algorithms approach that mimics "social-behavior" patterns to counter the problem of premature convergence. The standard job shop problem refers to a static job shop scheduling problem where there are n jobs, each with m operations and there are m machines in the job shop. The objective of the scheduling problem is to sequence the various operations of the jobs to the machines so that the makespan is minimized. The benchmarked, standard job shop problems in [Muth and Thompson,

1963], viz, the 10x10 and the 20x5 problems were investigated (where nxm refers to n jobs and m machines). The symbolic representation of individuals is used in this problem. A gene or allele in this representation is a job and this allele is repeated m times (where m is the number of operations and the number of machines in the workshop). The ith occurence of an allele refers to the ith operation of the job. The crossover operator is the order-crossover operator and mutation involves the position-based random change of alleles.

In [Tamaki and Nishikawa, 1992], Genetic Algorithms based on the neighbourhood model is applied to a Job Shop scheduling problem. This problem differs from the standard Job Shop problem in that the number of operations in a job, need not necessarily equal the number of machines in the job shop. The performance measure of the scheduling problem is makespan. The encoding of the individuals in the problem is done with the aid of disjunctive graphs. An individual is represented as a binary string, where the length equals the number of disjunctive arcs. The 2-point crossover operator, in which an individual is split at two randomly selected points and its middle portion is swapped for that of its mate, and the traditional mutation operator were used in this Genetic Algorithms implementation.

In [Filipic, 1992], Genetic Algorithms were used to schedule 15 textile machines, such that the total energy consumption of the machines were minimized. This problem also includes constraints, such as, surcharge on excess power consumption and set-up times required to initialize each job before loading onto the textile machines. The individuals represent the schedules using symbolic representation. Each individual (schedule) is a string of length N, where N is the total number of jobs to be executed and the value of the ith string position denotes the set-up time of the ith job. A problem specific recombination operator was designed that includes both the crossover and mutation aspects of Genetic Algorithms.

In [Reeves and Karatza, 1993], Genetic Algorithms was applied to a dynamic flowshop problem. In this problem, Genetic Algorithms was used to solve the successive sequencing problem for those jobs that are available just before successive decision points. The representation of individuals is symbolic and uses a sequence representation. In addition, the parent selection was based on rank. A sequence based crossover operator and an adaptive mutation rate was used in this problem.

In [Husbands and Mill, 1991], Genetic Algorithms was used to 'optimize' simultaneously, individual job process plans and the overall schedule. The Genetic Algorithms model for this problem consists of several populations. The genotype of each population is different and represents the solution to one of the subproblems. The fitness of an individual takes into account the interactions with members of other populations and the separate species co-evolve in a shared world. Possible conflicts, such as disputes over shared resources are decided by a further co-evolving species, referred to as the Arbitrators, and these evolve under pressure to make decisions that benefit the whole ecosystem. Symbolic representation is used to model the individuals and the PMX crossover operator is used. The mutation operator is a problem specific one.

In [Syswerda and Palmucci, 1991], Genetic Algorithms was used in a resource scheduling problem, where equipment in a Navy research laboratory are to be scheduled for use by various users. An individual in this problem is a string that consists of permutations of tasks, and the symbolic representation is used. In this application of Genetic Algorithms, the swap mutation (choose two positions in the individual and swap the positions) operator and the position-based crossover operator are used.

In the Genetic Algorithms approach of [Yamada and Nakano, 1992], the individuals are defined using the symbolic representation, using the operation completion times of the jobs. The general Job Shop problem, using the Muth and Thompson testbed problems, was the focus of this approach. The crossover operation is based on the Giffler and Thompson's algorithm, which has the mutation operation built into it. The elitist model of population propagation is used. In the crossover operation, two parents produce two children. The best of the 4 individuals (2 parents and 2 children) is chosen to represent in the next generation. The other representative in the next generation will be the better of the two children or the other child if one was already chosen previously.

In the Job Shop problem of [Nakano and Yamada, 1991], the performance measure investigated is the makespan and the Muth and Thompson benchmark scheduling problems are used. The individual (also the schedule) is represented as a bit vector for every job pair. Each bit in the vector compares an operation of one job to the operation on each of the machines. A value of '1' is assigned if the operation of the first job is performed prior to the operation of the second job, and '0' otherwise. This kind of representation requires MN(N-1)/2 bits for a N job and M machines problem. The binary representation was chosen so that the conventional crossover and mutation operators could be used.

However, an additional operation is required to make sure that the individuals that are recombined (crossovered and mutated) are legal.

In [Bagchi et al, 1991 and Uckun et al, 1993], the performance measure investigated is machine utilization. The representation of individuals was symbolic, and uses the queue of job orders. The ordering of the individuals represents the scheduling priority of the job orders. More sophistication to the representation is realized through including the process plans and the machines required to process the jobs. The PMX, order and position based crossover operators are used. Two mutation operators were also used; the first is an order-based mutation operator and in the second mutation operator, a job is randomly chosen and an alternative process plan is assigned to it.

In [Whitley et al, 1989], Genetic Algorithms were used to solve the Travelling Salesman Problem and later extended to the Job Shop scheduling problem. In this approach, a one at a time recombination strategy was adopted, i.e., the newly created offspring replaces the lowest ranking individual, rather than a parent. In addition, the reproductive trials are allocated according to rank of the individual in the population, rather than according to proportionate fitness. The crossover operator used in this approach is called the "edge recombination" operator that preserves edges. There was no specific mutation operator (mutation was assumed in the recombination process). As for the job shop scheduling problem, the approach assumes that once the first machine is sequenced, then the remaining machines can be scheduled using simple rules. The schedule of the first machine was treated as a Travelling Salesman Problem. A operator referred to as the 'scheduler' was developed that creates the final schedule of the job shop from the schedule of the first machine. The fitness of the individuals (schedules of the first machine only) was computed from the quality of the final schedules generated by the 'scheduler.'

In [Cleveland and Smith, 1989], the sector scheduling problem is investigated. The sector scheduling problem refers to an automated subcomponent of an actual computer board assembly and test facility. The overall facility is composed of sectors, each of which resembles a flow line. The only control one has over the processing that takes place within the sector is in determining the release times for jobs entering the first station. Once a job has entered a sector, it automatically moves from station to station and jobs are processed at each station in a first come first served manner. Individual representation was symbolic and the generational replacement strategy was adopted. Recombination was performed

| Source | Problem (performance measure) | Representation of Individuals | Crossover Operator | Mutation Operator |
|---|---|---|---|---|
| [Mattfeld et al, 1994] | Standard Job Shop Problem (makespan) | Symbolic | Order-crossover | Postion based random change of alleles |
| [Tamaki and Nishikawa, 1992] | Job Shop Problem (makespan) | binary | 2-point crossover | Traditional Mutation |
| [Flipic, 1992] | Textile machines (minimize energy consumption) | Symbolic | A problem specific recombination operator was designed (includes both crossover and mutation) | |
| [Reeves and Karatza, 1993] | Dynamic Flow Shop | Symbolic | Sequence based crossover | Adaptive Mutation rate |
| [Husbands and Mill, 1991] | Optimizing Process Plans and the overall schedule | Symbolic | PMX crossover | Problem specific |
| [Syswerda and Palmucci, 1991] | Resource Scheduling of Lab Equipment | Symbolic | Position based crossover | Swap mutation |
| [Yamada and Nakano, 1992] | Standard Job Shop Problem (makespan) | Symbolic | Crossover is based on the Giffler and Thompson's Algorithm - mutation is built in | |
| [Nakano and Yamada, 1991] | Standard Job Shop Problem (makespan) | Binary | Conventional crossover | Conventional Mutation |
| [Bagchi et al, 1991 and Uckun et al, 1993] | Job Shop Problem | Symbolic | PMX, order and position based crossover operators | Order-based mutation and swapping alternative process plans |
| [Whitley et al, 1989] | Travelling Salesman Problem and the Job Shop problem | Symbolic | Edge Recombination | Mutation was assumed in the crossover operator |
| [Cleveland and Smith, 1989] | Sector Scheduling Problem | Symbolic | Subtour replace, subtour chunking, subtour swap, PMX and weighted chunking | No mutation |
| [Kanet and Sridharan, 1991] | Scheduling Problem | Symbolic | Problem Specific | Problem Specific |

Figure 4.9: Summary of Genetic Algorithms application in Scheduling

using 5 operators, viz, the subtour replace, subtour chunking, subtour swap, PMX and weighted chunking operators. No mutation operation was discussed.

In the scheduling problem investigated by [Kanet and Sridharan, 1991], all the jobs have a single operation. Each job has an arrival time, a nominal processing time and each job can be processed on any one of several parallel non-identical machines. Exact processing times are dependent on which machine the job is scheduled for. The performance measure is cost. The Genetic Algorithms, in this problem was applied slightly differently. A subset of the best schedules from the initial population is chosen to comprise the mating pool. The schedules in the mating pool are then combined through a mating process (problem specific crossover) to produce offsprings that together with the mating pool comprise the next generation. The representation of the individuals is symbolic and consists of an ordered list of unscheduled tasks. Mutation is performed by selecting a member of the mating pool randomly and replacing it completely with this new randomly generated schedule.

Figure 4.9, summarizes the application of Genetic Algorithms to scheduling problems in terms of the problem investigated, the individual representation, crossover operator and the mutation operator.

## 4.3 Premature Convergence

Premature convergence is the loss of population diversity before optimal or at least satisfactory values have been found [Eshelman and Schaffer, 1991]. As proportions of better schema increase in the population, the proportions of less desirable schema will decrease. Eventually, the very best schemata and individuals occur in dominating proportions and, in this sense, the search converges to a solution. This convergence occurs prematurely, that is before the optimal solution is found [Booker, 1987].

Traditionally, the burden of diversity preservation as well as vigorous recombination has been placed completely upon crossover, thereby forcing a tradeoff between preservation and exploration. This tradeoff is not a serious constraint, provided a very large population is used. Given a large population, it is unlikely that an individual will be crossed over with a sibling or near ancestor, so it is very likely that the schemata from any parent will be tested in a new context. When Genetic Algorithms are used for practical

applications, the cost is measured in terms of total evaluations rather than generations, and large populations become inefficient. When small populations are used, on the other hand, operators that tend to preserve schemata, also cause rapid convergence [Eshelman and Schaffer, 1991].

Although premature convergence can be easily avoided by an increase of the mutation rate, a higher mutation rate, tends to break growth of the high performance individuals as well as the poor ones [Tamaki and Nishikawa, 1992]. On the other hand, if we restart populations when they converge, we may be able to keep the processing of schemata high [Eshelman and Schaffer, 1991].

Strategies for maintaining population diversity can be be naturally grouped according to where they occur in the Genetic Algorithms' reproduction-recombination-replacement cycle [Eshelman and Schaffer, 1991]. The strategies for combating premature convergence will be grouped as:

- mating strategies
- crossover strategies
- population management strategies
- other strategies

### 4.3.1 Mating Strategy

All other things being equal, children produced by crossover by diverse parents will tend to be more diverse. An 'incest prevention' mechanism has been proposed as an approach for preventing similar individuals from mating. Individuals are randomly paired for mating and are only mated if their Hamming distance is above a certain threshold. The threshold is initially set to the expected average Hamming distance of the initial population, and then is allowed to drop as the population converges. Mating diverse individuals have the side effect that more of the genetic schemata are disrupted by crossover, because fewer of the schemata are shared [Eschelman and Schaffer, 1991].

Another variation of the idea of exploration versus exploitation is to view Genetic Algorithms from the perspective of population diversity and selective pressure. Population diversity and selective pressure are inversely related. Maintaining population diversity

offsets the effect of increasing selective pressure and increasing selective pressure results in a faster loss of population diversity. Many of the various parameters used to tune Genetic Algorithms are indirect ways of affecting selective pressure and population diversity. As selective pressure is increased, the search focuses on the top individuals in the population, but because of this "exploitation," genetic diversity is lost. Reducing the selective pressure, increases "exploration" because more genotypes and more schemata are involved in the search. Selective pressure can be directly controlled by effecting reproduction according to rank [Whitley, 1989].

Using rank allows the algorithm to maintain a relatively steady percent involvement. There is some increased ability to discover good solutions, simply because the search is never hampered by lack of alleles and continues longer. While the ranking method prevents diversity from being lost too quickly, it does not allow allele proportions to change when warranted [Booker, 1987].

By keeping the population sorted and performing selection on the basis of rank, a constant selection differential is maintained between the best and the worst individuals in the population. This slows down initial convergence to promising subspaces, and increases the potential of finding the optimum solution in the final stages [De Jong, 1992].

Concepts from Taboo Search are also used in Genetic algorithms to prevent loss of diversity in the population. The newly created individual itself can be considered to be taboo, effectively forbidding it to reproduce until its taboo tenure has expired. This would be an imitation of the natural process of maturation of an organism before reproduction can occur [Reeves, 1993].

### 4.3.2 Crossover Strategy

The most straightforward method for making crossover more vigorous is to increase the rate of crossover use. A more radical method for maintaining population diversity is to use a more disruptive crossover operator, such as, uniform crossover [Eshelman and Schaffer, 1991]. Uniform crossover is a technique that is best applied to binary strings.

Crossover is usually implemented by choosing one crossover point at random and then exchanging segments between the two parent strings. Each application of the crossover operator searches both familiar and unexplored regions of the search space. A shortcoming of this form of crossover is that some of the schemata present in the parent strings are "disrupted" and not transmitted to the offsprings. An immediate extension is to use two random crossover points instead of one, which has been demonstrated to improve performance and also enhance exploration [Booker, 1987].

Crossover becomes less effective over time as the strings in the population become more similar. Consequently, it was speculated that the performance of the Genetic Algorithms can be improved if crossover is constrained to always produce variations wherever possible. More generally, for any two individuals, only the reduced substrings containing non-matching alleles are considered. Crossover points are randomly selected for these reduced strings, and then mapped back into the original strings [Booker, 1987].

Another aspect of crossover that can be beneficially manipulated is the frequency at which it is applied. Most often, the crossover rate is kept fixed throughout the evolution. Dynamic variation of the crossover rate may be beneficial. An entropy measure over the entire population, is proposed in the literature. When this entropy measure increases, the crossover rate is decreased and vice versa. The purpose for doing this is to go 'easy' with crossover until the variation is absorbed; then introduce more variation by increasing the crossover rate. This form of crossover rate variation has been shown to be beneficial with classifier systems; however, their use with other applications has been questioned. The reason being that entropy is a measure of diversity and by the time diversity has decreased, it may be too late to halt or reverse premature convergence. A variable crossover rate is analogous to the temperature parameter in simulated annealing [Booker, 1987].

The neighbourhood model for crossover is also proposed as a means of preventing the loss of diversity in the population. The mates are chosen such that they are in the neighbourhood. Therefore, the reproduction occurs locally in the neighbourhood. Even if the fitness of an individual is relatively very high in the population, it can spread over the succeeding populations only through an overlap of the neighbourhoods. This prohibits a rapid increase of a relatively high performance individual and diversity in each population is favourably maintained [Tamaki and Nishikawa, 1992].

Another approach to preventing population diversity is to use the 'island model', where several populations are each isolated on an island. These populations are optimized by the Genetic Algorithms until they degenerate. The degeneration is then removed by refreshing the population on each island through individuals of the other islands and the evolution then continues [Braun, 1990].

### 4.3.3 Population Management Strategy

Duplicate checking is a strategy proposed to combat premature convergence. A new individual is added to the population if it is not identical to any member that already exists in the population. This strategy also has the side effect of disrupting more schemata by crossover. This approach has a larger overhead in terms of computation than incest prevention [Eschelman and Schaffer, 1991].

An approach, termed as the uniqueness value, has been suggested in the literature that avoids premature convergence. The uniqueness value is defined as the minimum Hamming distance allowed between any offspring and all existing strings in the population. Whenever a new individual is closer than this to an existing structure, the alleles values that match are randomly changed in the offspring until the required distance is achieved. To make sure the that Genetic Algorithms will eventually converge, the uniqueness value is decreased as the search proceeds. There are two problems with the uniqueness approach. Firstly, it becomes prohibitively expensive to implement when the population is large or when the strings are long. The second problem is that by forcing the genetic algorithm to sample the space robustly, parent strings are prevented from reliably transmitting good schemata to their offspring [Booker, 1987].

Rapid convergence often occurs after an individual or small group of individuals contribute a large number of offspring to the next generation. Since populations are finite, a large number of offspring for one individual means fewer offspring for the rest of the population. When too many individuals get no offspring at all, the result is a rapid loss of diversity and premature convergence. By measuring the percentage of current population producing offspring - a measure called percent involvement, - one can anticipate this rapid convergence and prevent it [Booker, 1987].

## 4.4 Mechanics of Genetics Algorithms for Job Shop Scheduling

In this section, a Genetic Algorithms approach to job shop scheduling will be described, that accommodates multi-criteria decision making and is applicable to the dynamic job shop and considers alternate job/machine routings. The general framework of the Genetic Algorithms approach adheres to the standard 4-stage Genetic Algorithms cycle, consisting of Evaluation, Reproduction, Crossover and Mutation (Figure 4.1). The details of these different stages will be described below. However, it is important to discuss the representation of individuals, prior to discussing the various stages.

### 4.4.1 Representation

Each individual in this Genetic Algorithms approach represents a schedule (Figure 4.10). An individual is a string of resource/job-operation pair allocations, ordered chronologically in time. This list of resource/job-operation pairs, is devoid of the starting time and completion times of the various operations. These information is not necessary to define a schedule. It is also assumed that the operation allocated to a machine cannot begin until a previously allocated operation is completed. An implicit assumption about this representation is that no operation or job is purposely delayed. In other words, when an operation is available to be processed on a machine, it will be processed without any delay. This assumption results in a unique mapping between the individual as represented by Figure 4.10, and a schedule.

The Genetic Algorithms approach begins by creating a population of individuals based on the workload outstanding at the job shop. The population consists of schedules that are created using the random dispatching rule, i.e., operations are randomly assigned to available machines.

### 4.4.2 Evaluation

The individuals that constitute the population are evaluated based on the performance measures of interest, say makespan, flowtime, waittime, cost or tardiness. To facilitate multicriteria decision making, these performance measures are normalized in the same manner as described in Chapter 3 (Eqns. 3.5-3.7). The maximum and minimum

performance measures of the population are identified and the performance measures of an individual are then scaled between '1' and '0' using the maximum and minimum performance measures of the population. A composite fitness of an individual is computed as the weighted sum of these normalized performance measures.

### 4.4.3 Reproduction

Based on the discussion of premature convergence in the previous section, it should be noted that reproduction based on proportional fitness, is an inferior strategy in combating premature convergence. This is because of the scaling problems associated with fitness-proportionate reproduction. A better strategy as noted earlier is to perform reproduction based on rank. However, rank based reproductive strategies do not allow for allele proportions to change when required [Booker, 1987].

A compromise between rank-based and fitness-proportionate reproduction is to perform the normalized fitness-proportionate reproduction. The normalization of fitness values will eliminate the scaling problems and will maintain consistent reproductive pressure. The reproductive probability of an individual is computed in exactly the same manner as for fitness-proportionate reproduction, where the fitness is replaced with the normalized fitness. The reproductive probability is equal to the ratio of the normalized fitness of an individual to the sum of normalized fitness of the population.

Once the probabilities of all the individuals in the population are computed, then the 'roulette wheel' strategy could be used to perform the reproduction. This strategy assumes that a wheel is divided into sectors corresponding to the number of individuals in the population and the size of each individual sector is proportional to its probability. This wheel is then turned randomly and the individual that corresponds to the sector that is reached is then reproduced in the next generation.

### 4.4.4 Crossover

A order-based crossover operator is proposed. In this crossover operation, two parents produce two children (Figure 4.11). A random point is chosen and the two parents are dissected at that point. The substrings of the parents before the dissection point are

inherited by the children. One cannot concatenate the children substrings from the substrings of the parents aft of the dissection point, as would have been with the conventional crossover operation. In order to preserve the precedence constraints, a slight modification is necessary.

The elements of the substring inherited by a child is removed from the other parent. The order of the remaining substring of the parent is preserved and is then concatenated to the child. For example (Figure 4.11), Child 1 inherits the substring of Parent 1 before the dissection point. The elements of this substring is then removed from Parent 2. The remnants of Parent 2 is then concatenated to Child 1. The same procedure is applied to Child 2, thereby producing 2 children. The operation described here, preserves the precedence constraints of the child. The generational replacement strategy (where children replace parents in the next generation) is adopted.

### 4.4.5 Mutation

Two mutation operators are proposed (Figure 4.12). The first mutation operator is the simple swap mutation operator where two adjacent alleles of an individual are interchanged. This mutation operation is performed only if it does not violate the precedence constraints of the job.

The second mutation operation is proposed to accommodate alternate machines to process a job operation. In this operation, an allele is chosen randomly and the resource on which the operation is to be processed is replaced with one of the alternative resources.

The mechanics of the Genetic Algorithms approach, described above is summarized in Figure 4.13. The procedure outlined above is for solving static scheduling problems. In the case of dynamic scheduling problems, the procedure outlined above will be modified slightly. In the case of dynamic scheduling problems, the dynamic problem will be viewed as a series of static problems. Whenever a dynamic event occurs, the schedule generated using Genetic Algorithms will be regenerated by including the new dynamic event in the previous schedule. A dynamic event consists of three distinct events, and they include job arrivals, breakdown of machines and repair of machines. This is summarized in Figure 4.14.

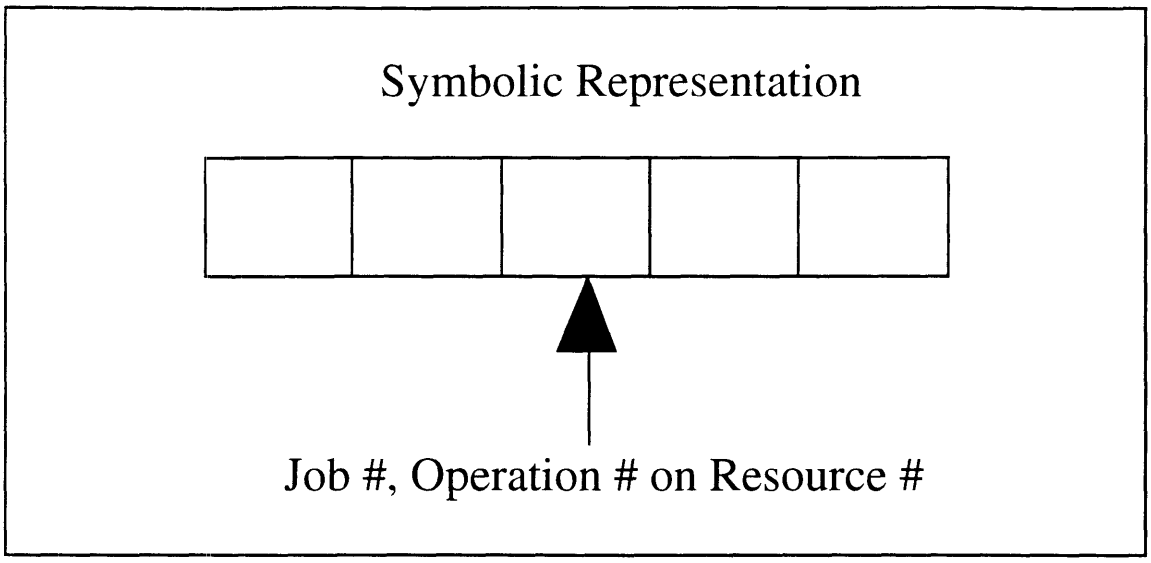


Figure 4.10: Representation of Individuals

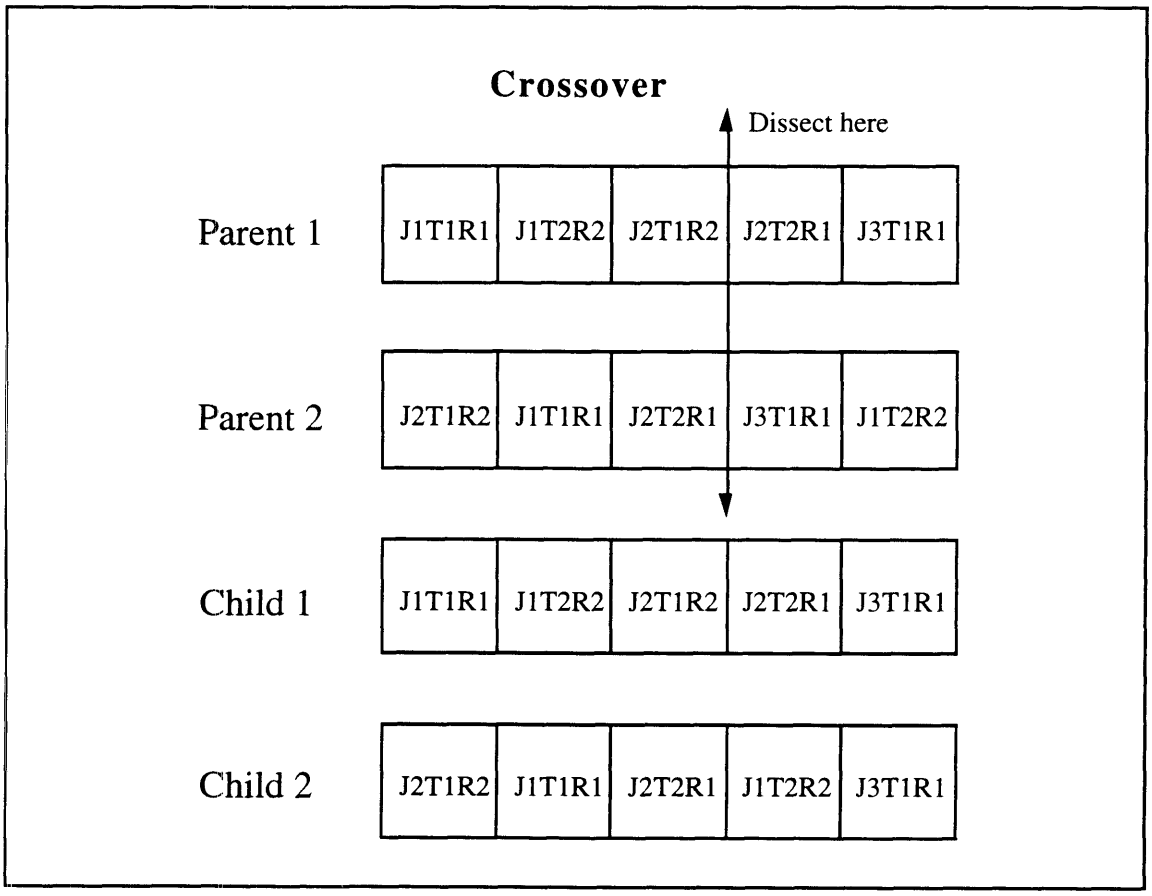


Figure 4.11: The Crossover Operator

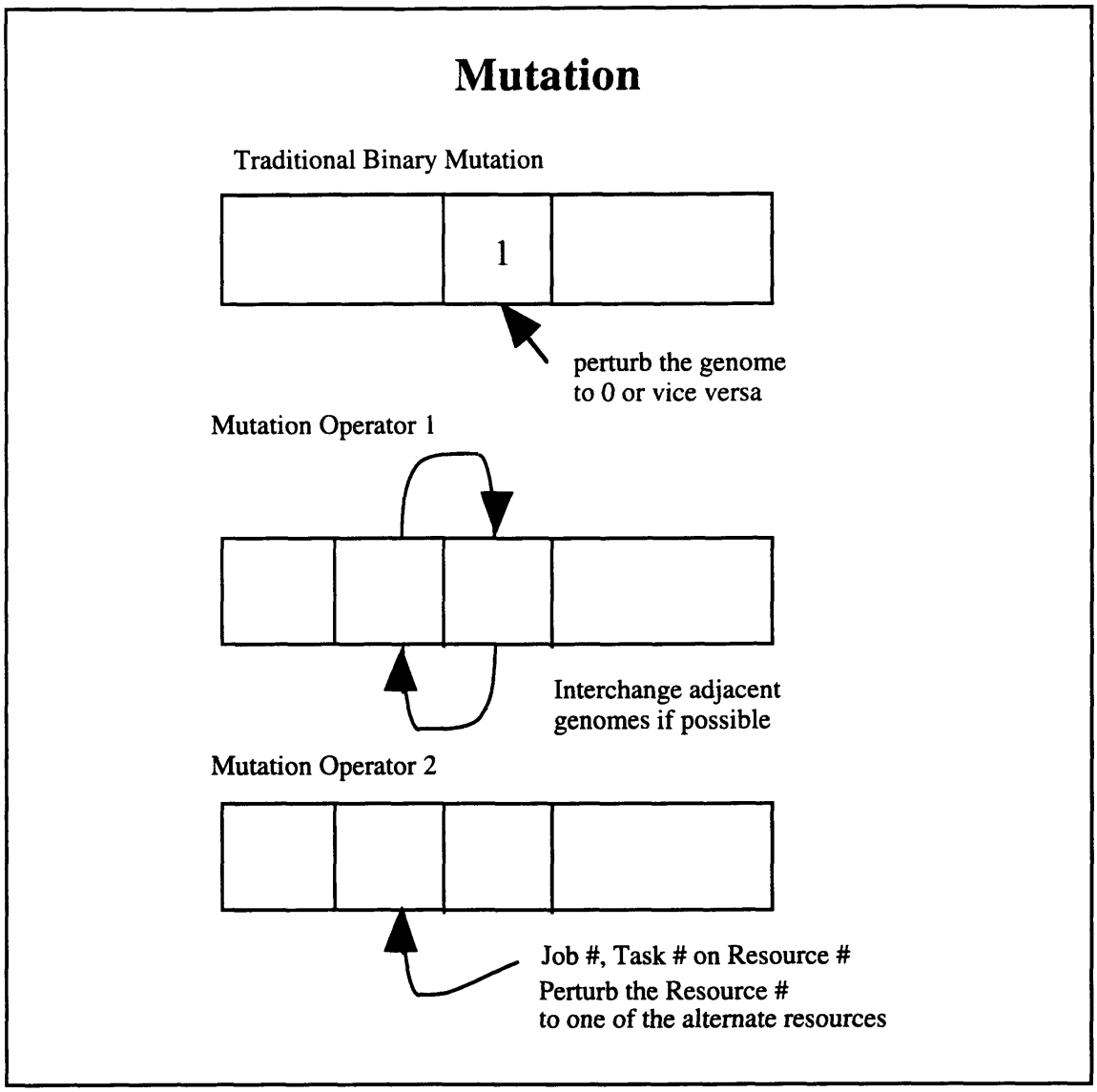


Figure 4.12: The Mutation Operators

The general Genetic Algorithms framework described in this thesis, to solve the dynamic job shop problem, is similar to the approach proposed in [Reeves and Karatza, 1993]. However, the problem investigated in [Reeves and Karatza, 1993] is the dynamic flow shop problem as opposed to the dynamic job shop problem that is investigated in this thesis. As a result, the genetic operators described in [Reeves and Karatza, 1993] differ considerably from those proposed in this thesis.

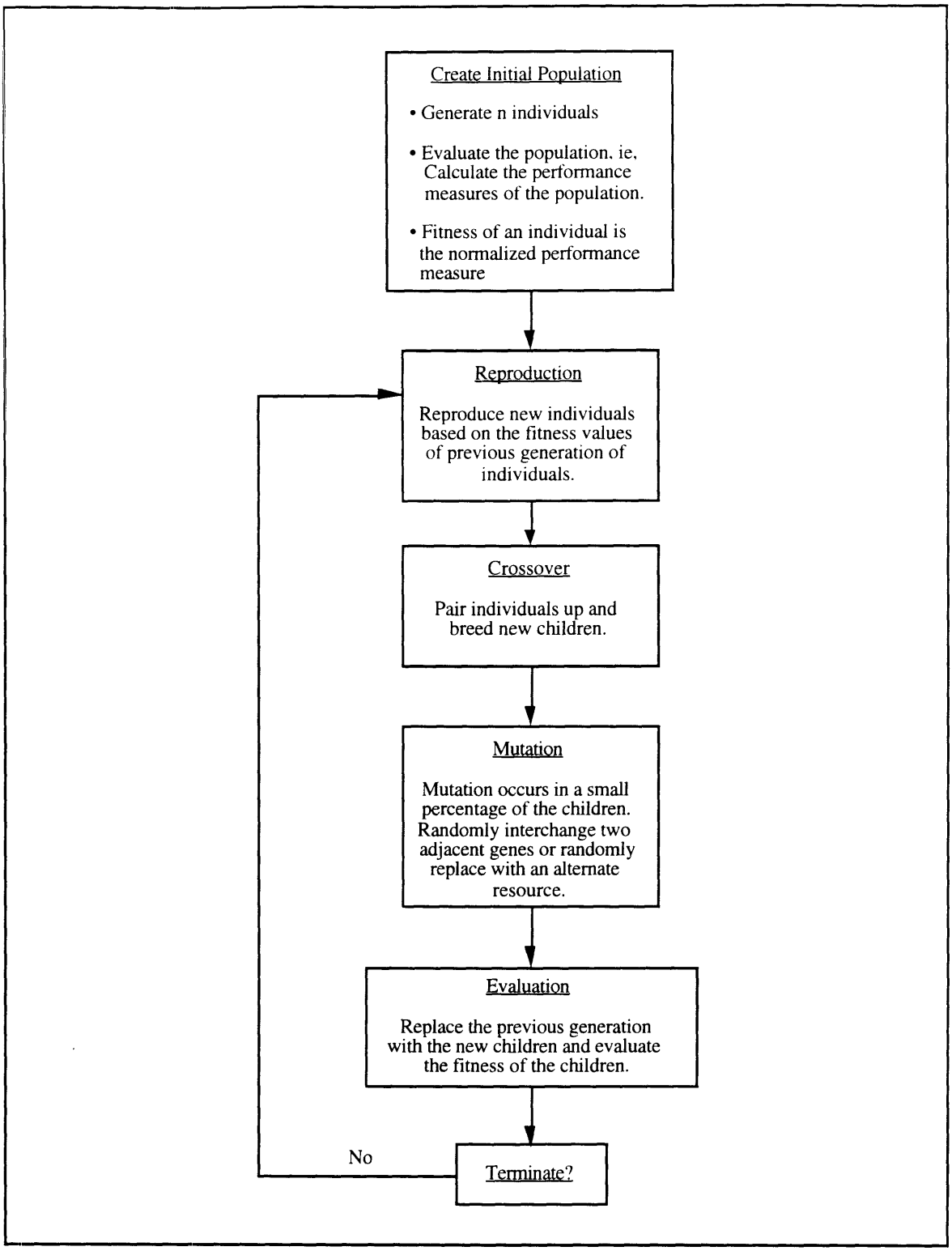


Figure 4.13: Summary of the Genetic Algorithms Approach for Job Shop Scheduling Problems.

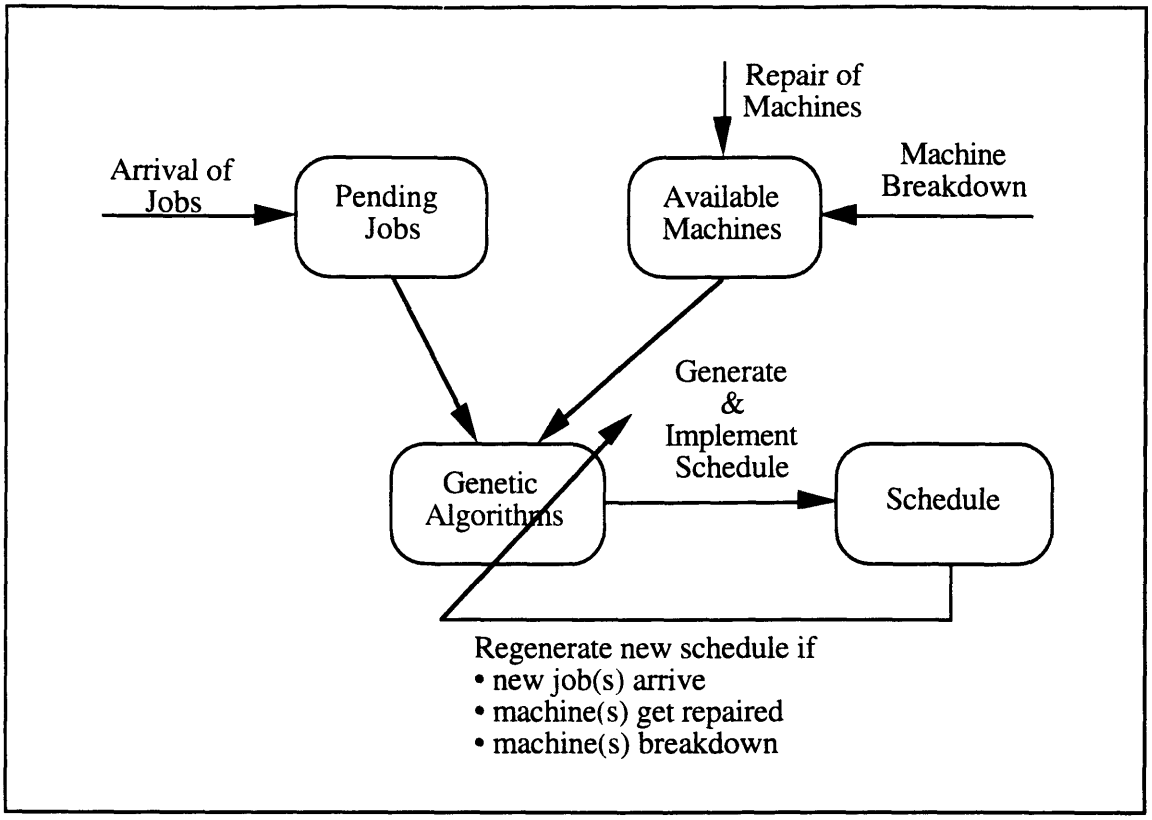


Figure 4.14: The Genetic Algorithms Approach for Dynamic Job Shop Scheduling

The crossover operator used in this thesis is similar to the order-crossover operator proposed in [Braun, 1990]. The author is not aware of the use of the second mutation operator described in Figure 4.12 in job shop scheduling problems. This mutation operator accommodates the processing of a job operation on several alternative machines. Research in job shop scheduling, on the other hand, has primarily focused on the standard job shop problem (Figure 4.9), where the problem is static and assumes that each job operation can be performed on only one machine.

## 4.5 Results and Discussion

The Genetic Algorithms approach as outlined above was applied to the Muth and Thompson benchmark problems. The application of the Genetic Algorithms approach to dynamic job shop scheduling, will be discussed in Chapter 5. The specifications of the benchmark problems are specified in Appendix B.

As noted in Chapter 3, the performance measure of interest in the static benchmark problems (the 6x6 and 10x10 problems) is makespan. Since this is a single criteria scheduling problem, the standard fitness-proportionate reproduction strategy was utilized (i.e. without normalizing the fitness values). The makespans obtained using some common dispatch rules, the Fuzzy Logic approach discussed in Chapter 2 and the Extreme Value Theory approach are summarized in Figure 4.15.

| Method | Makespan (6x6 problem) | Makespan (10x10 problem) |
|---|---|---|
| Optimal | 55 | 930 |
| Genetic Algorithms | 55 | 1006 |
| SEVAT | 57 | 988 |
| SPT | 88 | 1074 |
| LPT | 67 | 1197 |
| FIFO | 74 | 1259 |
| LIFO | 72 | 1223 |
| Random | 60 | 1106 |
| Fuzzy | 71 | 1134 |

Figure 4.15: Makespans obtained for the Muth and Thompson 6x6 and 10x10 static job shop scheduling problems

The Optimal makespan for the 6x6 and 10x10 problems are 55 and 930 respectively [Vancheeswaran and Townsend, 1993]. The optimal makespan for the 6x6 problem is relatively easy to obtain. The 10x10 problem is reported in the literature to be much harder. The makespan of 1006 [Chen, 1993] for the 10x10 problem, compares favourably with all the other dispatching rules and the Fuzzy Logic Approach. This result also compares very favourably with the results published in [Vancheeswaran and Townsend, 1993].

A makespan of 965 has been achieved using Genetic Algorithms, [Nakano and Yamada, 1991]. The individuals in this approach were represented as a bit vector for every job pair and as discussed earlier, this representation requires MN(N-1)/2 bits for a N job and M machines problem. The number of bits become excessive for moderate job shops where the number of machines could be in the order of 20 and the number of jobs could easily reach 100 (99000 bits for the example just cited). Such an approach may solve the

10x10 problem with very good results, but its application to other problems may be severely hampered. The author would like to reiterate that the purpose of the Genetic Algorithms approach and the Extreme Value Theory approach are not to solve the benchmark problems to optimality. Rather, these approaches are designed to be applied to more realistic job shop problems, in particular the dynamic shop job problem. The benchmark problems are used as a means of "calibrating" the usefulness of these approaches and to compare them with other approaches.

# Chapter 5
# Dynamic Job Shop Scheduling

A review of the job shop scheduling approaches in which Genetic Algorithms had been applied (Figure 4.9), shows that most of the problems attempted were the standard job shop problems (examples of which are the Muth and Thompson 6x6 and 10x10 job shop scheduling problems). The situation is similar, based on the literature survey presented in Chapter 2. The Muth and Thompson problems [Muth and Thompson, 1963], were developed in the 1960s and have become a standard for many job shop scheduling exercises. These problems are inadequate in reflecting the requirements of actual job shops. Firstly, these problems are designed for the makespan performance measure, and makespan is not necessarily an important performance measure in job shop scheduling as compared to, say, cost or tardiness. Moreover, actual job shop scheduling problems are multi-criteria in nature. Secondly, these problems are static problems, while actual job shop scheduling problems are dynamic in nature, where the arrival of jobs are not deterministic and resources (machines) breakdown and get repaired. Thirdly, these problems do not consider alternate job/machine routings.

In this chapter, the dynamic job shop scheduling problem will be examined. This problem addresses the shortcomings of the static benchmark scheduling problems discussed above. Besides including dynamic events, the dynamic job shop scheduling problem will be designed such that it also considers multiple performance measures simultaneously (namely cost and tardiness) and alternate job/machine routings.

In the last twenty years or so, the primary method of analyzing the dynamic job shop scheduling problem has been through computer simulation of real or representative job shops [Ramasesh, 1990 and Day and Hottenstein, 1970]. The number of factors that characterize a dynamic job shop is very large and that there is considerable variation in the modelling and experimentation across the vast number of studies that has been reported in the literature [Ramasesh, 1990].

This chapter will discuss some of these factors that characterize the analysis of dynamic job shops. After this discussion, specifications for a dynamic job shop will be designed based on the reported studies in the literature. The two scheduling methodologies proposed in this thesis, namely, scheduling based on Extreme Value Theory and Genetic Algorithms, will be tested on this dynamic job shop problem and the performance of these approaches will be compared against some common dispatching rules and the Fuzzy Logic dispatching method as outlined in Chapter 2.

Some of the factors that characterize the analysis of a dynamic job shop are broadly categorized as, Job Shop, Jobs, Performance Measures and Simulation Analysis (Figure 5.1). The categorization is by no means unique. Some of these factors could also be placed in a different category than that shown.

## 5.1 Factors that influence the Job Shop

Some of the factors that influence the Job shop include, the size of the job shop, the utilization of the job shop and the breakdown/repair of the machines in the job shop. This section considers factors that influence the general makeup of the job shop. Factors that influence the jobs, the performance measures or the simulation analysis, will be discussed separately.

### 5.1.1 The Size of the Job Shop

The typical size of a real job shop ranges from about 30 to 1000 machines [Day and Hottenstein, 1970]. Most simulation studies of dynamic job shops consider job shops consisting of relatively small number of machines that range from about 4 to 12 (Figure 5.2). The reason for this discrepancy is that, the size of the job shop does not significantly influence the performance of the job shop. Furthermore, it is reported in the literature that a job shop consisting of six machines is adequate in representing the complexity involved in a large dynamic job shop [Day and Hottenstein, 1970 and Ramasesh, 1990].

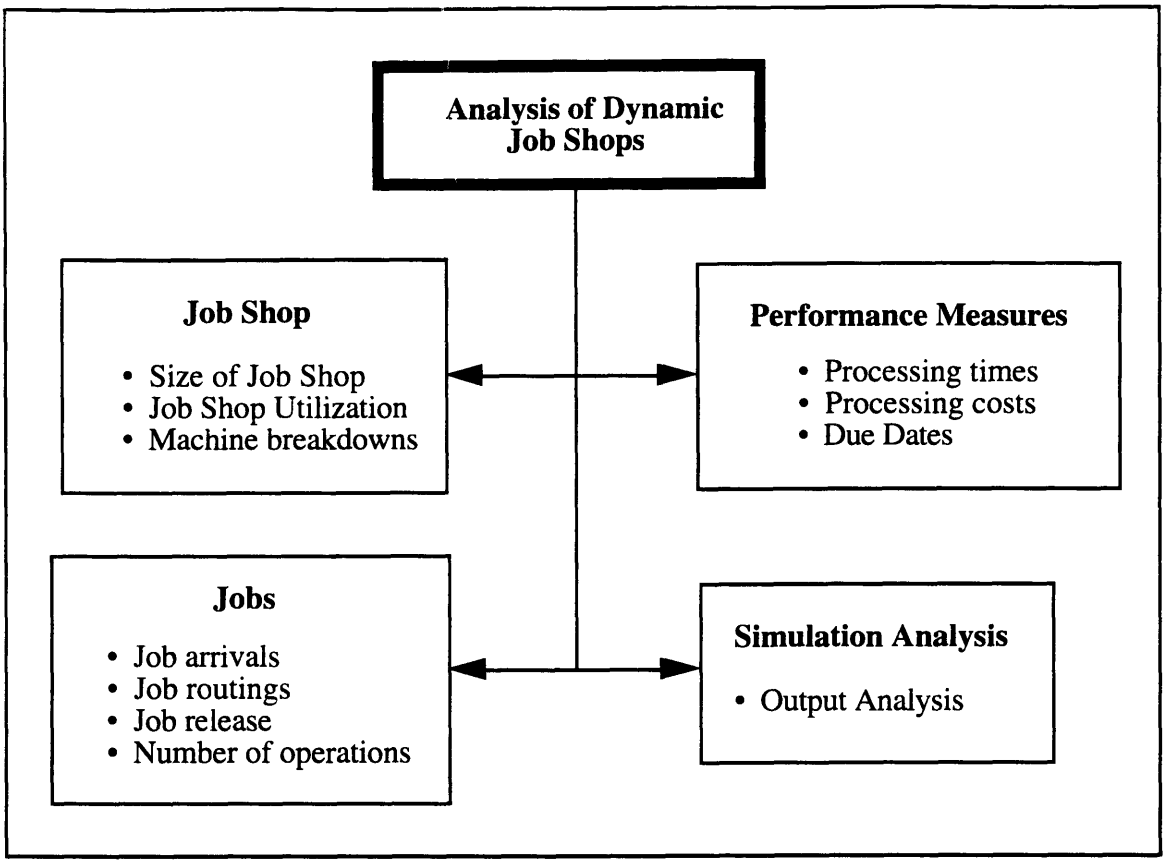


Figure 5.1: Factors Influencing the Analysis of a Dynamic Job Shop

| Reference | Number of Machines |
|---|---|
| [Ragatz and Mabert, 1988] | 5 |
| [Karsiti, et al, 1992] | 10 |
| [Udo, 1993] | 8 |
| [Raghu and Rajendran, 1993] | 12 |
| [Sim et al, 1994] | 9 |
| [Kannan and Ghosh, 1993] | 10 |
| [Enns, 1993] | 4 |
| [Anderson and Nyirenda, 1990] | 8 |
| [Scudder et al, 1993] | 9 |
| [Vig and Dooley, 1991] | 5 |
| [Gee and Smith, 1993] | 8 |

Figure 5.2: Size of Job Shops reported in recent Dynamic Job Shop studies.

### 5.1.2 Job Shop Capacity / Job Shop Utilization

Most studies in the literature report that the job shop utilization levels are set by adjusting the job arrival rates [Ramasesh, 1990]. However, it must be noted that utilization levels so set, are by no means exact, because of the random nature of the job arrivals [Ramasesh, 1990]. In addition, there is a very severe interdependence associated with utilization rates set this way and the TWK (total work content) due dates assignment approach (Eqn. 5.5). For example, due dates may be tight, even with an allowance factor, K=7, (which in most studies are categorized as moderate or loose due dates) when the shop load level is high at around 95%. Conversely, with K=3 (categorized as tight), due dates may not necessarily represent tight conditions, if the shop load level is low [Ramasesh, 1990]. The relationship between job arrival rates and the shop utilization levels is given by [Karsiti et al, 1992]:

$$\lambda = \frac{M * \eta}{\bar{n} * \bar{p}} \tag{5.1}$$

where $\lambda$ is the job arrival rate
$M$ is the number of machines
$\eta$ is the machine load capacity
$\bar{n}$ is the average number of operations
$\bar{p}$ is the average operation time

A summary of several studies of dynamic job shops indicate that the shop utilization levels vary between 80% and 95% (Figure 5.3).

### 5.1.3 Machine Breakdowns

The largest source of randomness in many manufacturing systems is that associated with machine breakdowns. To explicitly account for the breakdowns in a simulation model, it is important to have an accurate assessment of mean running time and mean repair time for the actual system [Law and McComas, 1989]. In most dynamic job shop scheduling problems, machine breakdowns are not considered. In one of the few studies that considers machine breakdowns [Benton, 1993], the time between failures of each

machine was assumed to follow a erlang-distribution, with mean of 30 days and parameter=1, and the downtimes are exponentially distributed with a mean of 0.2 days.

| Reference | Shop Utilization |
|---|---|
| [Ragatz and Mabert, 1988] | 87% |
| [Karsiti, et al, 1992] | 90% |
| [Udo, 1993] | 90% |
| [Raghu and Rajendran, 1993] | 85% and 90% |
| [Kannan and Ghosh, 1993] | 90% |
| [Enns, 1993] | 90% |
| [Anderson and Nyirenda, 1990] | 90% |
| [Scudder et al, 1993] | 80% and 90% |
| [Vig and Dooley, 1991] | ranges from 85% to 95% |
| [Gee and Smith, 1993] | 90% |

Figure 5.3: Job Shop Utilization Levels reported in recent Dynamic Job Shop studies.

## 5.2 Factors that influence the Jobs

In this section, factors that influence the jobs in a dynamic job shop will be discussed. Factors, such as, the job arrival process, job routings, job release policy and the number of operations per job are examined.

### 5.2.1 Job arrival process

In many simulation studies of the dynamic job shop, the Poisson arrival rate has been assumed. The mean for the Poisson distribution may be determined arbitrarily, but the usual practice is to set it at a lower value than the mean service rate of the shop in order to prevent continual shop overload. Given that the times of arrival occur by a Poisson process, it can be shown that the interarrival times have a related exponential distribution. Hence the exponential distribution is also representative of the Poisson process, but it describes the time between arrivals, and specifies that these time arrivals are completely random. Empirical observations of job shops have revealed that the arrival of jobs approximates very closely to a Poisson distribution [Day and Hottenstein, 1970].

It is also reported that the Poisson distribution is a good approximation to the arrival process if the different sources that generate job arrivals are statistically independent [Ramasesh, 1990]. Other distributions described in the literature are the erlang, binomial, geometric and empirical distributions. A survey of recent studies of dynamic job shop scheduling problems is tabulated in Figure 5.4 and shows that the Poisson and exponential distributions are extensively used to model the job arrival process.

### 5.2.2 Job Routings

Most studies dealing with dynamic job shop scheduling have assumed fixed job routing. The sequence of machines that a typical job must go through for the required operations was assumed to be given at the outset. In general, however, if one starts with a set of job orders, only the sequence of operations can be deduced in advance from this information and not the machine sequence [Karsiti et al, 1992].

In actual manufacturing environments, process routings can be far more complex and even dynamic. Rarely is every job processed on every machine, as in the classical job shop problem [Rodammer and White, 1988]. In real manufacturing systems, two or more machines are capable of processing the same operations and therefore, there exists several alternate routings for a job in a real job shop [Kim, 1990].

The random sequence in which a job will visit the machines in the job shop is determined by sampling without replacement from a distribution, and the uniform distribution has been generally used for this purpose [Ramasesh, 1990]. A heuristic rule, known as the minimum job in queue, (MINJQ) is applied to assign jobs to machines. From the set of machines capable of performing an operation, the machine with the minimum number of jobs waiting in its queue will be chosen to process the job. By assigning the job to the machine with the minimum number of jobs waiting in the queue, the total load can be distributed fairly among all the machines [Karsiti et al, 1992].

| Reference | Distibution | Parameter |
|---|---|---|
| [Ragatz and Mabert, 1988] | Poisson | mean = 1.39 arrivals per hour |
| [Karsiti et al, 1992] | Poisson | arrival rate is adjusted so that machine capacity was 90% |
| [Udo, 1993] | exponential | mean = 120 |
| [Raghu and Rajendran, 1993] | exponential | arrival rate adjusted so that machine utilization levels of 85% and 95% were achieved. |
| [Sim et al, 1994] | Poisson | mean varying from 0.6 - 0.775 |
| [Benton, 1993] | exponential | adjusted to achieve total machine utilization |
| [Kannan and Ghosh, 1993] | exponential | mean = 10h |
| [Enns, 1993] | Poisson | adjusted to achieve utilization rate of 90% |
| [Anderson and Nyirenda, 1990] | Poisson | not known |
| [Scudder et al, 1993] | Poisson | adjusted so that nominal utilizations are 80% and 90%. |
| [Vig and Dooley, 1991] | exponential | adjusted so that nominal utilizations are 90% |
| [Gee and Smith, 1993] | geometric | mean = 2.5 time periods to achieve 90% utilization |

Figure 5.4: Job arrival models reported in recent Dynamic Job Shop studies.

### 5.2.3 Job Release

Most research reported in the literature assume that jobs are released to the shop as they are received, thereby effectively bypassing the releasing decision. There are reasons why jobs should not be released as they are received [Ragatz and Mabert, 1988], viz.,

- parts delivered to the finish stockroom or the assembly floor, long before they are actually needed, tie up unnecessary capital.

- parts produced too soon, may disappear, may be damaged by excessive handling or may occupy valuable space for too long.

- jobs released too early to the shop floor will compete for resources with more urgent jobs and may interfere with the progress of those jobs.

It has been suggested that if job release is controlled carefully, sophisticated dispatching rules can be replaced with the FCFS (first come, first served) dispatching rule, without any deterioration in shop performance [Ragatz and Mabert, 1988].

Some of the job releasing mechanisms reported in the literature are [Ragatz and Mabert, 1988]:

- Backward infinite loading, (BIL): releases a job to the shop, a fixed number of hours per job operation ahead of its due-date.

- Modified infinite loading, (MIL): gives each job a flow allowance based on the number of operations in the job and also the number of jobs waiting in queue along the job's routing.

- Maximum number of jobs (MNJ): jobs are released to the shop floor, one at a time, until the number of jobs in the shop has reached a specified maximum.

- Backward finite loading (BFL): this releasing mechanism works with a planning horizon that is broken into time buckets. The current workload profile of each machine in the job shop is also maintained. This releasing mechanism works backward from a job's assigned duedate (i.e., starting with the last operation in the job and working towards the first). This mechanism attempts to fit each operation into available capacity for the appropriate machine. If adequate capacity is not available, the mechanism backs the operation up to an earlier time bucket. Once the operation is loaded, the preceding operations are loaded in a like manner.

In another approach [Scudder et al, 1993], a decision variable called status is computed as follows:

$$STATUS_i = d_{ij} - \Delta p_{ij} - t \qquad (5.2)$$

where $d_{ij}$ = duedate of operation j of job i
$t$ = present time
$\Delta$ = release multiplier

The job is placed in the inactive queue, if the status is positive. Otherwise, the job is placed in the active queue. Jobs in the active queue can be immediately assigned to the machines.

### 5.2.4 Number of operations per job

In simulation studies of dynamic job shops (Figure 5.5), the number of operations per job are usually uniformly distributed. The number of operations per job in these studies, vary between 1 and 12.

## 5.3 Factors that influence the performance measures

In this section, aspects of dynamic job shop scheduling that influences the performance measures will be discussed. In particular, performance measures that deal with tardiness and cost are examined. Factors that influence these performance measures are the processing times, processing cost and the due-date assignment policy.

### 5.3.1 Processing times

Operation times, i.e., the times for setup and for processing are set when the jobs arrive at the job shop. Most studies have used the exponential distribution for the setup and processing times. Setup times are usually assumed to be sequence independent and are thus combined with the processing times [Ramasesh, 1990].

Other distributions that have been used include the normal, poisson, uniform and erlang distributions. The nature of processing time distribution affects the performance of the job shop significantly. For example, the exponential distribution has been found to

favour the SPT dispatching rule [Ramasesh, 1990]. Figure 5.6 tabulates the various models used to generate processing times for simulation studies of dynamic job shops.

| Reference | Distribution | Number of Operations per job |
|---|---|---|
| [Ragatz and Mabert, 1988] | Random | maximum limited to 8 |
| [Karsiti, et al, 1992] | Uniform | [1,9] |
| [Udo, 1993] | Uniform | [4,8] |
| [Raghu and Rajendran, 1993] | Uniform | [4,12] |
| [Sim et al, 1994] | not known | [3,6] |
| [Benton, 1993] | Uniform | [2,7] |
| [Kannan and Ghosh, 1993] | Uniform | [4,12] |
| [Enns, 1993] | Random | [2,6] |
| [Anderson and Nyirenda, 1990] | Uniform | [1,8] |
| [Scudder et al, 1993] | Random | [2,7]; mean = 4 |
| [Vig and Dooley, 1991] | Uniform | [1,10] |
| [Gee and Smith, 1993] | Uniform | [1,6] |

Figure 5.5: Number of operations per job, reported in recent Dynamic Job Shop studies.

### 5.3.2 Processing cost

Most of the simulation studies of dynamic job shops deal with the tardiness performance measure. In one of the few studies dealing with the cost performance measure [Scudder et al, 1993], the hourly processing cost of each machine was uniformly distributed between $6 and $12 an hour. This study also considered a setup cost of $15 an hour. Material cost was also assumed to vary uniformly for each job between [$50,$100].

### 5.3.3 Due Dates

Research in the area of due date assignment tends to fall into two classes. One group, assumes that ready times for jobs are given, so the research effort is directed towards determining the flow allowance in order to set the due dates. The other group assumes that due dates are given, so the research task becomes that of determining the lead time in order to set ready times for the job [Udo, 1993].

| Reference | Distribution | Parameter |
|---|---|---|
| [Ragatz and Mabert, 1988] | exponential | mean = 1 hour |
| [Karsiti, et al, 1992] | not known | mean = 2.5 |
| [Udo, 1993] | exponential | mean = 120 |
| [Raghu and Rajendran, 1993] | rectangular | 3 cases: [1,50], [1,100] and 25% from [50,100] and 75% from [1,50] |
| [Sim et al, 1994] | not known | [1,4] |
| [Benton, 1993] | normal<br><br>uniform | (mean 9h, variance 3h) for processing times<br>[0.25, 2.5]h for setup times |
| [Kannan and Ghosh, 1993] | normal | mean = 11.4h and standard deviation = 2.28h |
| [Enns, 1993] | exponential | mean = 1.0 |
| [Anderson and Nyirenda, 1990] | exponential | mean = 100 |
| [Scudder et al, 1993] | normal | mean = 9h; var = 3h<br>min processing time = 0.5h |
| [Vig and Dooley, 1991] | 2-Erlang | mean = 1.0 |
| [Gee and Smith, 1993] | 2 step procedure using geometric and exponential distributions | mean processing time = 5.2 time periods |

Figure 5.6: Processing times, reported in recent Dynamic Job Shop studies.

For either group of approaches, the methods for setting due dates can be either dynamic or static. The dynamic method employs job characteristics and job shop congestion information in determining due dates. The static method, on the other hand, considers only job content information such as arrival time, routing and processing times. For static methods, the job flow allowance is a fixed amount for given job data and does not depend on the workload status of the shop when the job arrives [Udo, 1993].

Due dates can also be set either exogenously or endogenously. Exogenous due dates are set by some external agency without regard to the processing characteristics of the shop itself, other jobs in the shop, or the dispatching rule to be used. Examples of the exogenous duedate setting methods include the CON (constant) and RAN (random) methods. The CON method sets the allowable shop time (difference between duedate and arrival time for a job) as a constant amount, independent of any characteristic of the job to which it is assigned. This method represents the case in which the salesman quotes

duedates uniformly on all orders. RAN duedates, in which the allowable shop time is assigned at random, corresponds to situations in which the duedate is chosen by the customer and accepted by the firm's salesman [Day and Hottenstein, 1970].

Endogenous duedates are internally set based on the characteristics of the jobs (total processing time, number of operations, etc. ) and the shop (shop utilization level, work load etc.) is more involved because the method of assigning the due dates will affect the performance of the job shop scheduling rules [Ramasesh, 1990].

Some of the common due-date assignment methods are as follows [Kaplan and Unal, 1993]:

| | | |
|---|---|---|
| CON (constant) | $D_i = r_i + k$ | (5.3) |
| RAN (random) | $D_i = r_i + e_i$ | (5.4) |
| TWK (total work content) | $D_i = r_i + kP_i$ | (5.5) |
| SLK (slack) | $D_i = r_i + P_i + k$ | (5.6) |
| NOP (number of operations) | $D_i = r_i + kN_i$ | (5.7) |

where $D_i$ is the due date assigned to job i;
$r_i$ is the time job i is released to the shop;
$P_i$ is the processing time of job i;
$N_i$ is the number of operations that job i will undergo;
$e_i$ is a random parameter;
k is a constant parameter.

The standard approach of meeting due-dates is via the use of the mean tardiness performance measure and this ignores the consequences of jobs that complete early. The earliness tardiness penalties approach is designed to counter this shortcoming. A basic and general model to incorporate both earliness and tardiness penalties is

$$f(s) = \alpha \sum_{i}^{n} E_i + \beta \sum_{i}^{n} T_i \qquad (5.8)$$

where $E_i$ and $T_i$ represents the earliness and tardiness of job i and $\alpha$ and $\beta$ are the earliness and tardiness penalty cost respectively [Baker and Scudder, 1990].

In an approach reported in [Ovacik and Uzsoy, 1994], due dates are assumed to be uniformly distributed on an interval determined by the expected workload of the system and two parameters, $\tau$ and R, where $\tau$ is the parameter denoting the percentage of operations expected to be tardy and R is the parameter that determines the range of the interval.

## 5.4 Output Analysis

The statistical analysis of the dynamic job shop requires that the job shop be in steady state. The transient states of the job shop will "contaminate" the estimate of the steady state means (mean Job Cost and mean Job Tardiness). Steady state does not mean that the output measure will take on the same value in a particular simulation run. Rather, it means that they will all have approximately the same distribution [Law and Kelton, 1991]. To eliminate the transient effect on the steady state means, some of the initial data must be removed and the steady state means are then calculated from the remaining data. This process is termed "warming up the model" or "initial-data deletion."

The procedure to identify the warm-up period is discussed in [Law and Kelton, 1991]. This is a graphical procedure in which, one monitors the plot of the simulation means and identify the point at which the plots level off at the steady state means. The procedure requires several independent replications of the simulation runs and is summarized in the following four steps [Law and Kelton, 1991]:

1 Make n replications of the simulation run (where $n \geq 5$). Each simulation run is of length m, where m is very large. Let $Y_{ji}$ be the ith observation from the jth replication.

2 Compute $\overline{Y}_i = \sum_{j=1}^{n} Y_{ji}$, for i=1,2,3,...m. This averaged process has the same transient mean curve as the original process, but its plot has only (1/n) of the variance.

3 To filter off the high frequency oscillations from the long-run trend, a moving average operator is applied to the averaged process of step 2.

4 The filtered-averaged data is then the plot from which the warm-up period is identified.

To estimate the steady state means from the simulation runs, an approach called "replication/deletion" will be applied. This approach is detailed in [Law and Kelton, 1991]. There are other alternative approaches that could also be applied, viz., the batch means method, the autoregressive method, the spectrum analysis method, the regenerative method and the standardized time series method.

The replication/deletion method is applied for obtaining the point estimate and the confidence interval for the simulation means. This method requires making k replications of the simulation runs. For each simulation run, data in the initial warm up period is deleted.

Let $Y_{ji}$ be the ith observation (of the reduced simulation data set, after deleting the initial warm up period), of the jth replication. We shall define the simulation mean, $X_j$, as

$$X_j = \frac{\sum_{i=1}^{p} Y_{ji}}{p} \tag{5.9}$$

where p is the size of the reduced simulation data set. The $X_j$'s are independent and identically distributed random variables and are approximately unbiased estimators of the simulation mean, and a $100(1-\alpha)$ percent confidence interval for this mean is given by

$$\overline{X}(k) \pm t_{k-1,1-\frac{\alpha}{2}} \sqrt{\frac{S^2(k)}{k}} \tag{5.10}$$

where k is the number of simulations runs, t denotes the student's t-distribution and

$$\overline{X}(k) = \frac{\sum_{i=1}^{k} X_i}{k} \tag{5.11}$$

$$S^2(k) = \frac{\sum_{i=1}^{k} [X_i - \bar{X}(k)]^2}{k-1} \qquad (5.12)$$

## 5.5 Dynamic Scheduling Problem

Based on the discussion of factors that characterize the dynamic job shop in the previous sections, a dynamic job shop problem was designed to evaluate the performance of the scheduling methods proposed in this thesis, namely the SEVAT procedure and the scheduling methodology based on Genetic Algorithms. The specifications of the dynamic job shop problem will be described along the same dimensions of Figure 5.1, viz, the Job Shop, the Jobs, the performance measures and the simulation analysis.

### 5.5.1 The Job Shop

Since it has been extensively reported in the literature that a job shop with six machines is adequate to represent the complexity involved in a large dynamic job shop [Day and Hottenstein, 1970 and Ramasesh, 1990], the dynamic job shop was designed to consist of six machines. The performance of the scheduling methods will be evaluated through simulation. A larger job shop will increase the simulation effort and require more computational resources.

In the discussion of the previous sections in this chapter, it was seen that most simulation studies of dynamic job shops assume that a job-operation can only be processed on one machine and that all the machines in the job shop are absolutely reliable and do not break down. Based on these assumptions, the job shop utilization level becomes a function of the job arrival rates. The job shop can therefore be maintained at a particular utilization level by manipulating the job arrival rates a priori. However, once the job shop is assumed to be flexible, (i.e. a job-operation could be processed on two or more machines in the job shop) and that the machines are unreliable (i.e. machines break down), the job shop utilization level is no longer the function of job arrival rates alone. The utilization level now depends on the reliability of the machines and the flexibility of the job shop. In view of this difficulty, the dynamic job shop was designed to have a fixed arrival rate for all jobs

and the job shop utilization level was allowed to 'float' according to the conditions of the job shop.

The unreliability of the machines are expressed in terms of two parameters, namely, the Mean Time Between Failure (MTBF) and Mean Time to Repair (MTTR). The failure and repair times for the machines are assumed to follow an exponential distribution. To infuse realism into the simulations, each of the machines were assigned different MTBFs and MTTRs. The MTBFs and MTTRs are chosen randomly from uniform distributions in the range of [5000, 15000] and [1000, 6000] time units respectively. These values were chosen such that, on average, a machine is operational 65% of the time.

### 5.5.2 The Jobs

The number of job types to be processed by the job shop was assumed to be 20. The number of operations per job type was assumed to vary uniformly in the range of [2,10] operations. The job arrival rates were fixed at around 2200 time units. The job routings in terms of operations were fixed. However, the machines to process the operations were not unique. Two situations were examined in the dynamic job shop. In the first case, the job shop was assumed to be absolutely flexible (all the machines were identical and can process any operation). In the second case, a job operation could be processed by three machines instead of all six in the job shop. The job release policy in the job shop was assumed to be immediate and no jobs were to be delayed intentionally.

### 5.5.3 The Performance Measures

The dynamic job shop was designed to operate under multiple performance measures. To this end, Mean Job Cost and Mean Job Tardiness were selected for the dynamic job shop under consideration.

Each machine was assumed to cost a fixed amount, for each time unit of operation. This cost rate was assumed to vary uniformly in the range of [1,6] monetary units /time unit for the various machines in the job shop. The cost of processing a job-operation is the product of the processing time required for the operation and the cost rate of the machine on which the operation is processed. The job cost is simply, the sum of the individual

operational costs incurred in processing all the operations of the job. The processing time of any job-operation was assumed to vary in the range of [1,100] time units.

Tardiness is by definition the positive difference between the completion time and the due date of a job. Two levels of duedates (tight and loose) were used to reflect the 'tightness' of duedates in the job shop. The duedates were exogenously set using the RAN rule, which reflects the situation where the customer sets the due dates. The loose duedates were assigned such that the duedate of a job on arrival, is uniformly varied between [-100, 1500] time units. The negative duedate, reflects the situation where a job is of high priority and needs to be completed as soon as possible. The average processing time for a job is about 300 time units. The 1500 time units for the duedates were chosen to reflect the spread of duedates that are about five times the average job processing times. Similarly, the tight duedates were chosen to lie uniformly between [-100, 500] time units, reflecting the case where the duedates were spread under two times the average job processing time.

### 5.5.4 Simulation Analysis

The approach outlined in the previous section was applied to identify the initial warm up period and a suitable length of simulation. For this purpose, the SPT rule was applied to a dynamic job shop with parameters that are described in Expt. 1 of Figure 5.9. The dynamic job shop was simulated for 3500 jobs and the filtered-moving average plots for Job Tardiness and Job Cost are shown in Figure 5.7. 9 replications were simulated (with a different random number seed in the simulations) and a moving window of 100 jobs was used to filter off the high frequency variations. From Figure 5.7, it may be seen, that the Mean Job Tardiness is much more variable than the Mean Job Cost. By closer examination of the two plots in Figure 5.7, one can roughly state that steady state begins at around 300 jobs. Therefore, for the purposes of this thesis, a warm up period was conservatively chosen at 500 jobs and the simulation length was made two times the warm up period, i.e. 1000 jobs. In other words, simulation data was collected, beginning at the completion of the 500th job and continued until 1500 jobs were completed.

A summary of the various parameters that were used to design the dynamic job shop is presented in Figure 5.8. These parameters are classified under the four dimensions discussed earlier, viz, the Job Shop, the Jobs, the Performance Measures and the Simulation Analysis.

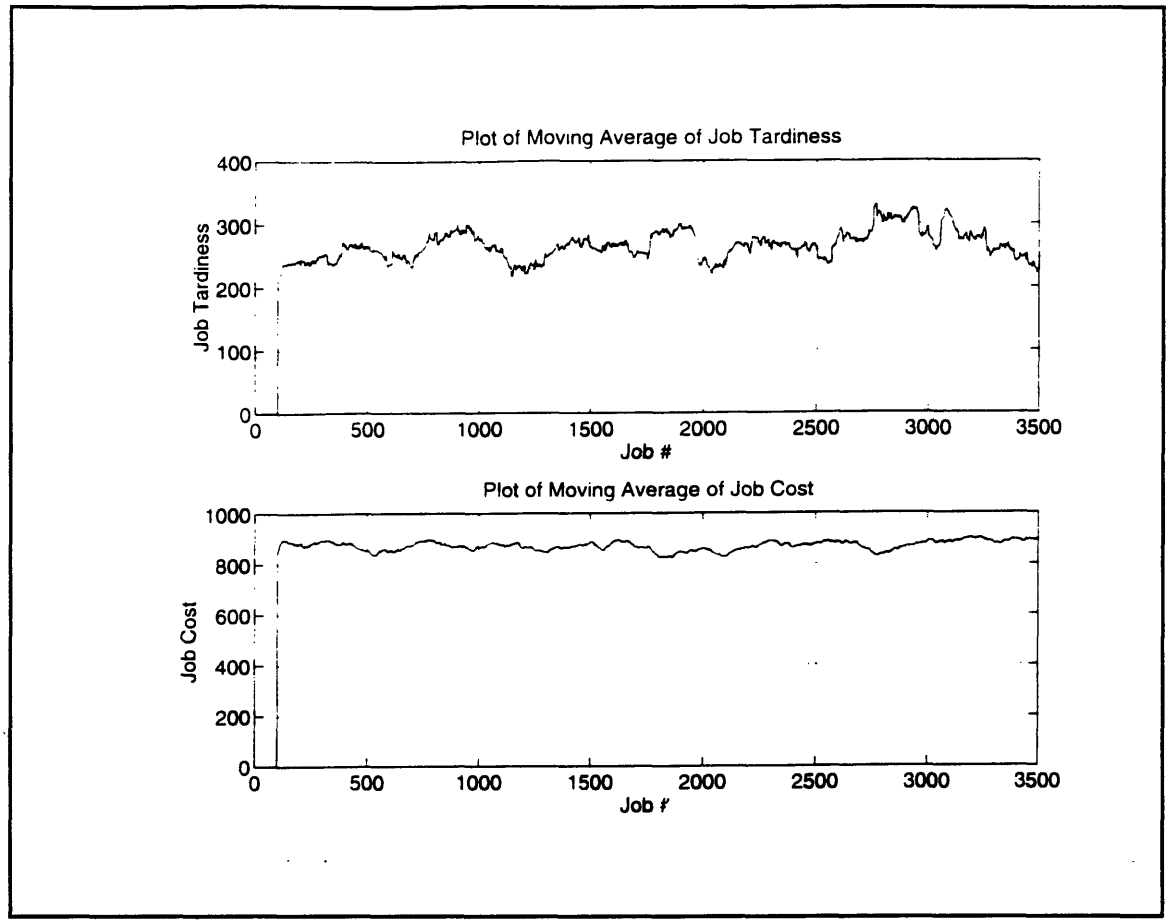


Figure 5.7: Identifying the Warm Up Period.

## 5.6 The Factorial Design

In this thesis, the two scheduling methods proposed, viz, the Extreme Value Scheduling approach and the Genetic Algorithms Scheduling approach will be compared against some common dispatching rules and the Fuzzy Logic Scheduling method described in Chapter 2. To ensure that the comparison of scheduling methods are performed over varied job shop conditions, three variable factors were identified, viz,

- the number of suitable resources to process a job operation,
- the 'tightness' of the duedates, and
- the inclusion/disclusion of machine breakdowns.

| Dimension | Characteristic | Specification |
|---|---|---|
| Job Shop | Size | 6 machines |
| | Utilization level | variable |
| | Machine breakdowns | MTBF = U[5000,15000]<br>MTTR = U[1000,6000] |
| Jobs | Arrival Rate | 2200 time units |
| | # of suitable resources to process a operation | 3 or 6 |
| | Job Release Policy | Immediate |
| | # of operations per job | U[2,10] |
| Performance Measures | Performance Measures | Cost and Tardiness |
| | Cost rate of machine | U[1,6] |
| | Processing time of an operation | U[1,100] |
| | Duedates | loose: U[-100,1500]<br>tight: U[-100,500] |
| Simulation Analysis | Method | Replications/Deletions |
| | Warm-Up period | 500 jobs |
| | Simulation period | 1000 jobs |

Figure 5.8: Summary of the Design of Experiments

Two levels were assumed for each of the factors and a full factorial experiment ($2^3 = 8$ experiments) was designed (Figure 5.9).

## 5.7 Results and Discussion

The dynamic job shop as described in Figure 5.8 was simulated for all the various experiments listed in Figure 5.9. For each experiment, the following scheduling methods were applied:

- Scheduling method based on Extreme Value Theory (SEVAT).

- Scheduling method based on Genetic Algorithms (GA).
- Scheduling method based on Fuzzy Logic.
- Shortest Processing Time (SPT) dispatching rule.
- Longest Processing Time (LPT) dispatching rule.
- First-In-First-Out (FIFO) dispatching rule.
- Last-In-Last-Out (LIFO) dispatching rule.
- Random dispatching rule.

| Expt # | # of suitable resources to process an operation | Duedates | Machine Breakdowns |
|---|---|---|---|
| 1 | 6 | Tight | Yes |
| 2 | 6 | Loose | Yes |
| 3 | 3 | Tight | Yes |
| 4 | 3 | Loose | Yes |
| 5 | 3 | Tight | No |
| 6 | 3 | Loose | No |
| 7 | 6 | Tight | No |
| 8 | 6 | Loose | No |

Figure 5.9: Factorial Design of Experiments

Each simulation of an experiment, for each of the scheduling methods, was replicated 10 times, using 10 different seeds for the random number generator in the simulations. The Mean Job Tardiness (MJT), the Mean Job Cost (MJC) and the 90% confidence intervals for these means were computed for each experiment/scheduling method. Subsequently, the maximum and minimum values for these means were also identified for each of the 8 experiments listed in Figure 5.9. The maximum and minimum values are used to normalize the means (so that the normalized means and their confidence intervals lie between 0 and 1). The normalization equations used are:

$$\text{Normalized MJT} = 1 - \frac{\text{MJT - Minimum MJT}}{\text{Maximum MJT - Minimum MJT}} \qquad (5.13)$$

$$\text{Normalized MJC} = 1 - \frac{\text{MJC - Minimum MJC}}{\text{Maximum MJC - Minimum MJC}} \qquad (5.14)$$

Since the objective of the scheduling exercise, is to perform scheduling in a dynamic job shop with respect to both Cost and Tardiness, the performance of each scheduling method will be measured using a composite performance measure, termed utility and is defined as:

$$\text{utility} = \frac{\text{Normalized MJT} + \text{Normalized MJC}}{2} \qquad (5.15)$$

The results of the dynamic job shop scheduling problem, using the various scheduling methods will be presented in the form of 'high-low' plots. (Figures 5.10-5.19). The horizontal bars on these plots denote the confidence intervals, while the dark spots denote the means. The results of the LPT rule were consistently inferior to the rest of the scheduling methods, and were dropped from the analysis, so that the normalization equations (5.13 and 5.14) become more discriminatory.

A major concern with the Genetic Algorithms scheduling method is that the simulation of the dynamic job shop may have resulted in a solution space filled with very "rich" solutions, therefore rendering the crossover and mutation operators to be ineffective. If this were true, then the results that are obtained using the Genetic Algorithms method, may not necessarily be attributed to the Genetic Algorithms exclusively, but rather to the rich solution space. A 3 stage approach to demonstrate the effects of the Genetic Algorithms Approach was adopted:

1 To completely switch off the crossover and mutation operations in each generation. At each decision point in the dynamic scheduling process, a population of individual schedules are initialized by generating random schedules. The best of these random schedules will be used at each decision point and the results obtained using this approach should be compared with the "total" Genetic Algorithms approach where the population of random schedules are permutated through crossover and mutation.

2 To switch off the mutation operation but allow the crossover operation to function. This will show the effect of crossover on the scheduling quality.

3 Similarly, the crossover operation is next switched off, but allowing the mutation operation to function. This will demonstrate the effect of mutation on the overall schedule quality.

The three-stage approach to demonstrate the effects of the Genetic Algorithms (GA) was performed on Experiment 2 of the eight dynamic scheduling experiments (Figure 5.9). The results are as shown in Figure 5.10. The results of the first three stages are labelled as "None", "Crossover" and "Mutation" respectively. The results obtained using the full GA approach as outlined in Chapter 4 of this thesis is also included ('GA') for comparison purposes. The 'GA' and the 'None' schemes provide very similar results. Furthermore, on comparing the 'utility' plot of Figure 5.10, the performance of the 'GA', 'None', 'Crossover' and 'Mutation' schemes are all very similar. This clearly indicates that the crossover and mutation operators do not contribute to the performance of the GA scheduling approach.

The crossover operator as outlined in the thesis is clearly not effective. It is theorized that with 'generational replacement,' sufficient improvement in the population fitness is not realized and this contributes to the dismal performance of the crossover operation in the overall GA scheme. If a more 'greedy' strategy, such as 'elitist recombination' [Thierens and Goldberg, 1994] were used, there may be a significant improvement in the performance of the crossover operation. In 'generational replacement', the two parents during crossover are completely replaced by the two resulting children. In 'elitist recombination,' the best two individuals from the family (consisting of two parents and the two children) replace the parents of the old generation. With 'elitist recombination' the population is always replaced with more superior individuals in each generation.

The results obtained using this new replacement strategy during crossover is labelled as 'New Crossover' and the resulting overall GA is labelled 'New GA'. From Figure 5.10, it is very clearly observed that the new replacement strategy has resulted in a significant improvement in performance compared to just using the best schedule from the random initialization of the population at each decision point. These results are for Experiment 2 of the 8 experiment dynamic scheduling problem (Figure 5.9), and were found to hold for all 8 experiments.

The Genetic Algorithms scheduling approach as outlined in Chapter 4 of this thesis, was modified to use elitist recombination instead of generational replacement. This modified Genetic Algorithms scheduling method together with the other scheduling methods outlined at the beginning of Section 5.7, were then applied to the 8 dynamic job shop experiments (Figure 5.9) and the results are shown in Figures 5.11- 5.18.

In the Extreme Value Approach, an alternative consists of job operations that can be immediately assigned to currently available machines (Figure 3.1). To evaluate the potential of these alternatives, random schedules consisting of job-operations that are currently pending at the job shop are formulated. From these random schedules, the potential of an alternative is estimated using Extreme Value Theory. A question that may arise pertaining to the formulation of alternatives is, "What is the effect of including 'future' assignments in the alternatives?" (i.e. the assignment of job operations on to machines that may not be currently available).

To answer this question, a variable called Decision Horizon (dh) will be introduced and is defined as the number of concatenations of future assignments to the alternatives. For example, referring to Figure 3.1,

- an alternative with dh=0 is

  R1T1 R2T2

- alternatives with dh=1 are

  R1T1 R2T2 *R1T3*
  R1T1 R2T2 *R2T3*

- alternatives with dh=2 are

  R1T1 R2T2 *R1T3 R3T4*
  R1T1 R2T2 *R2T3 R4T4*
  R1T1 R2T2 *R2T3 R3T4*

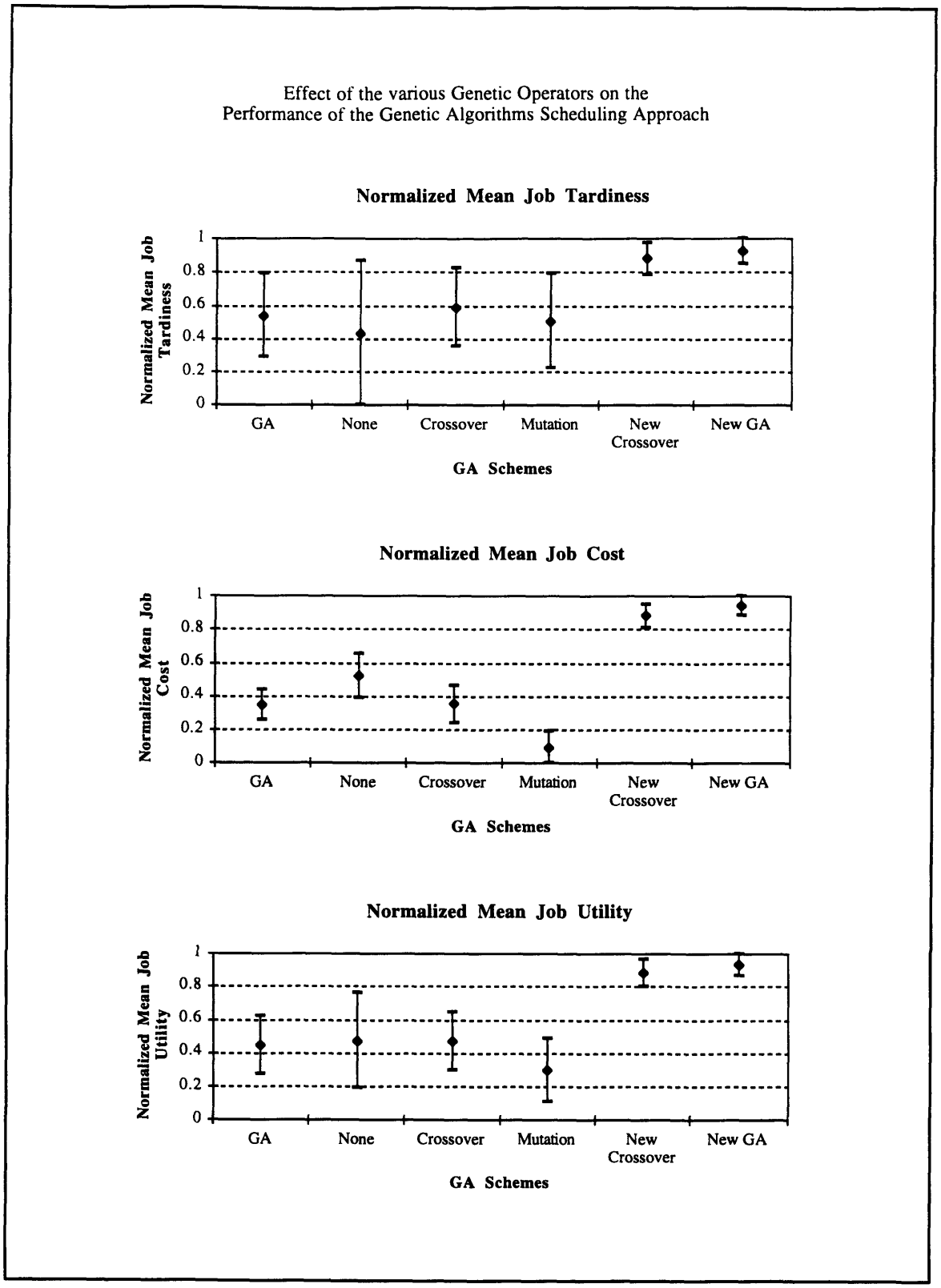


Figure 5.10: Effect of the Various Genetic Operators

Expt 1: **Tight** Duedates. **6** Suitable Resources to Process a job operation and with Resource Breakdowns.

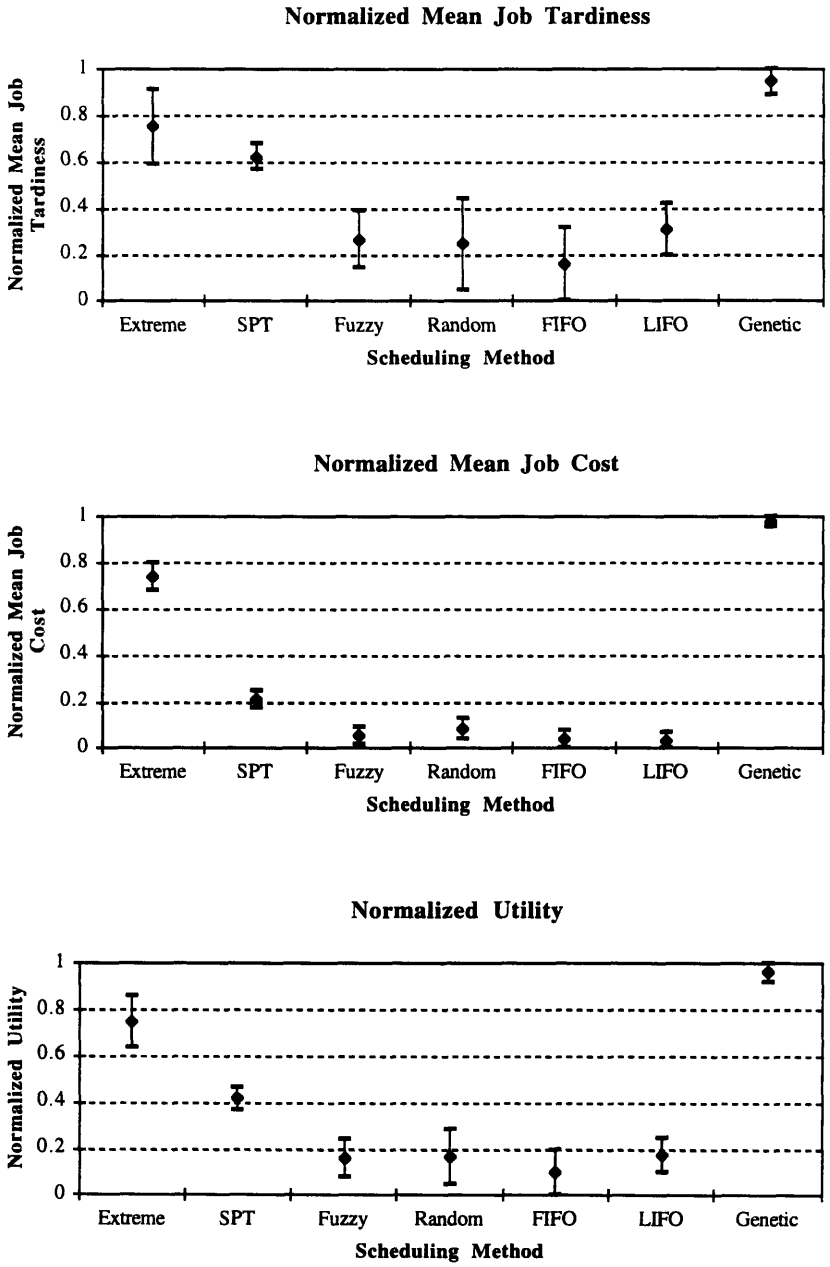


Figure 5.11: Results of Experiment 1.

Expt 2: **Loose** Duedates, **6** Suitable Resources to Process a job operation and with Resource Breakdowns.

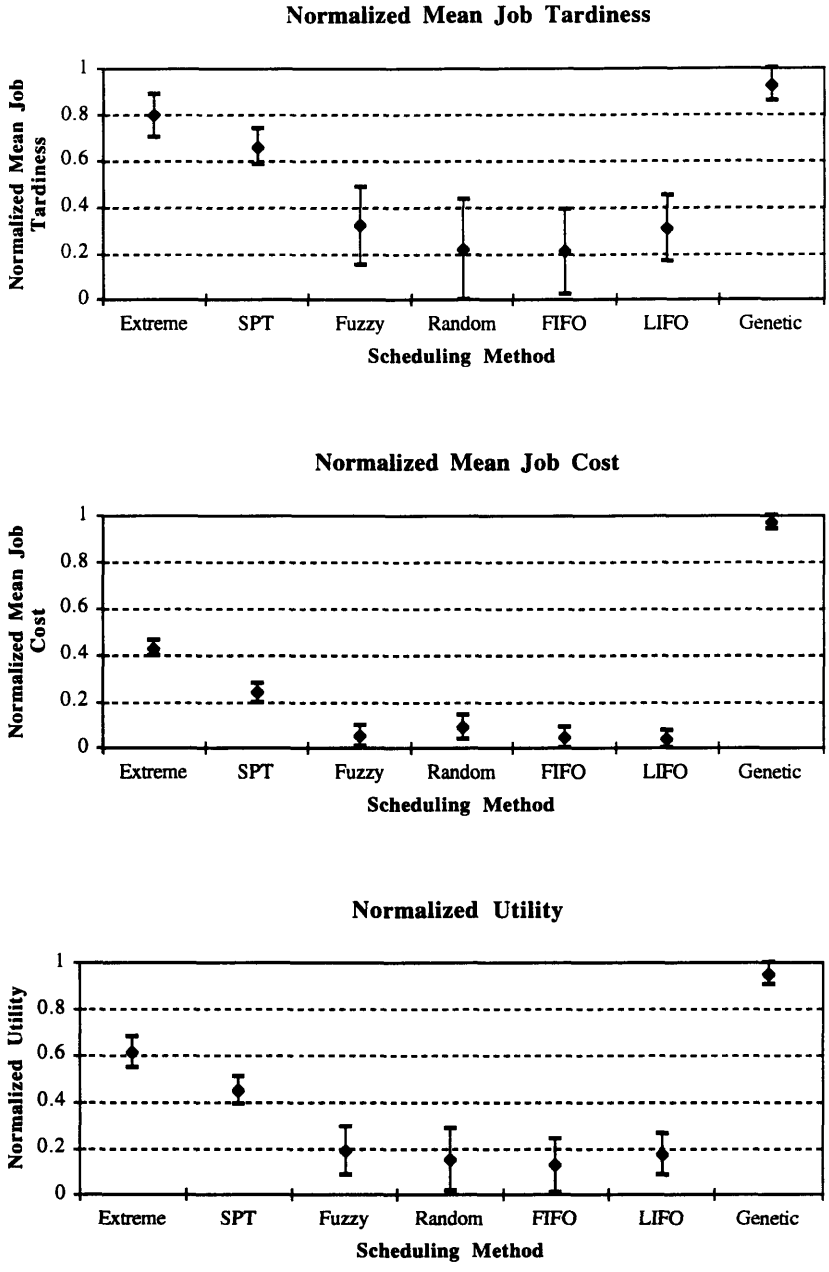


Figure 5.12: Results of Experiment 2.

Expt 3: **Tight** Duedates, **3** Suitable Resources to Process a job operation and with Resource Breakdowns.

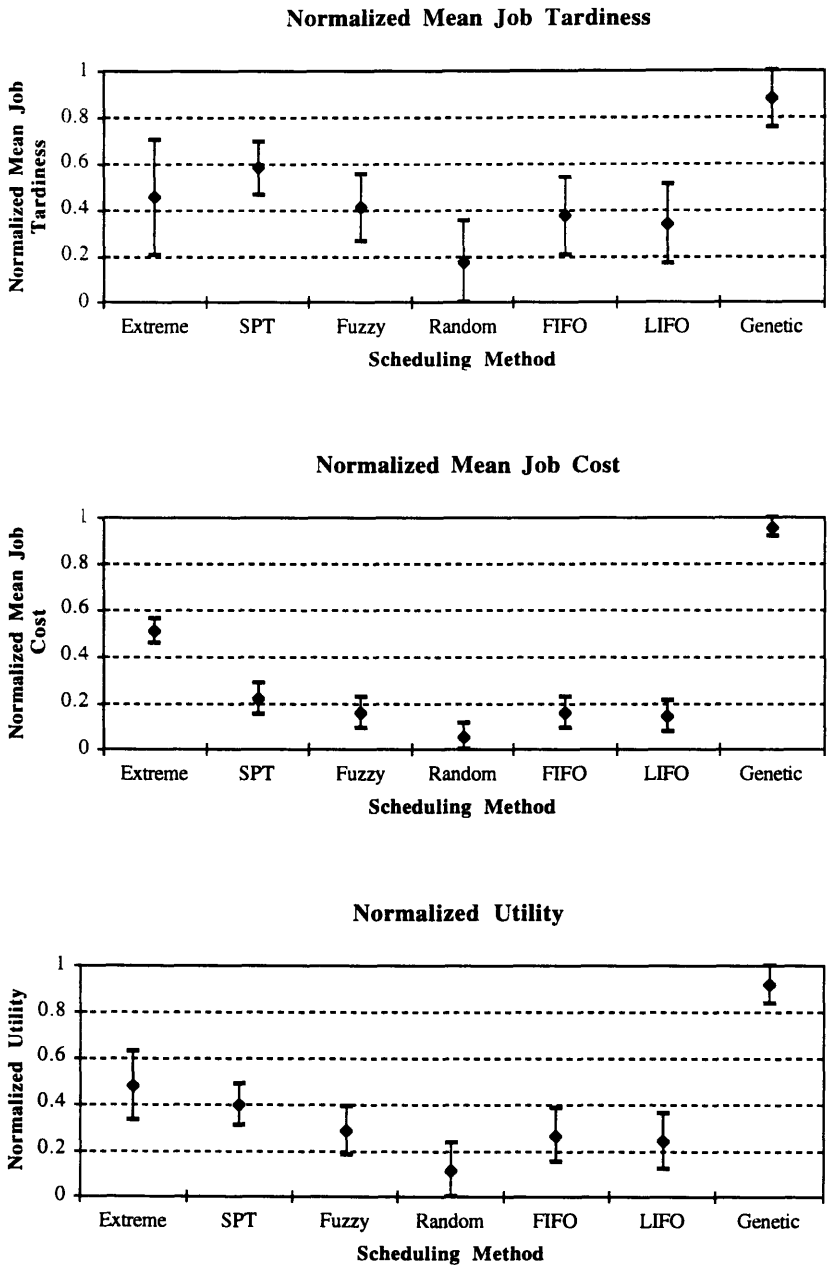


Figure 5.13: Results of Experiment 3.

Expt 4: **Loose** Duedates, **3** Suitable Resources to Process a job operation and with Resource Breakdowns.

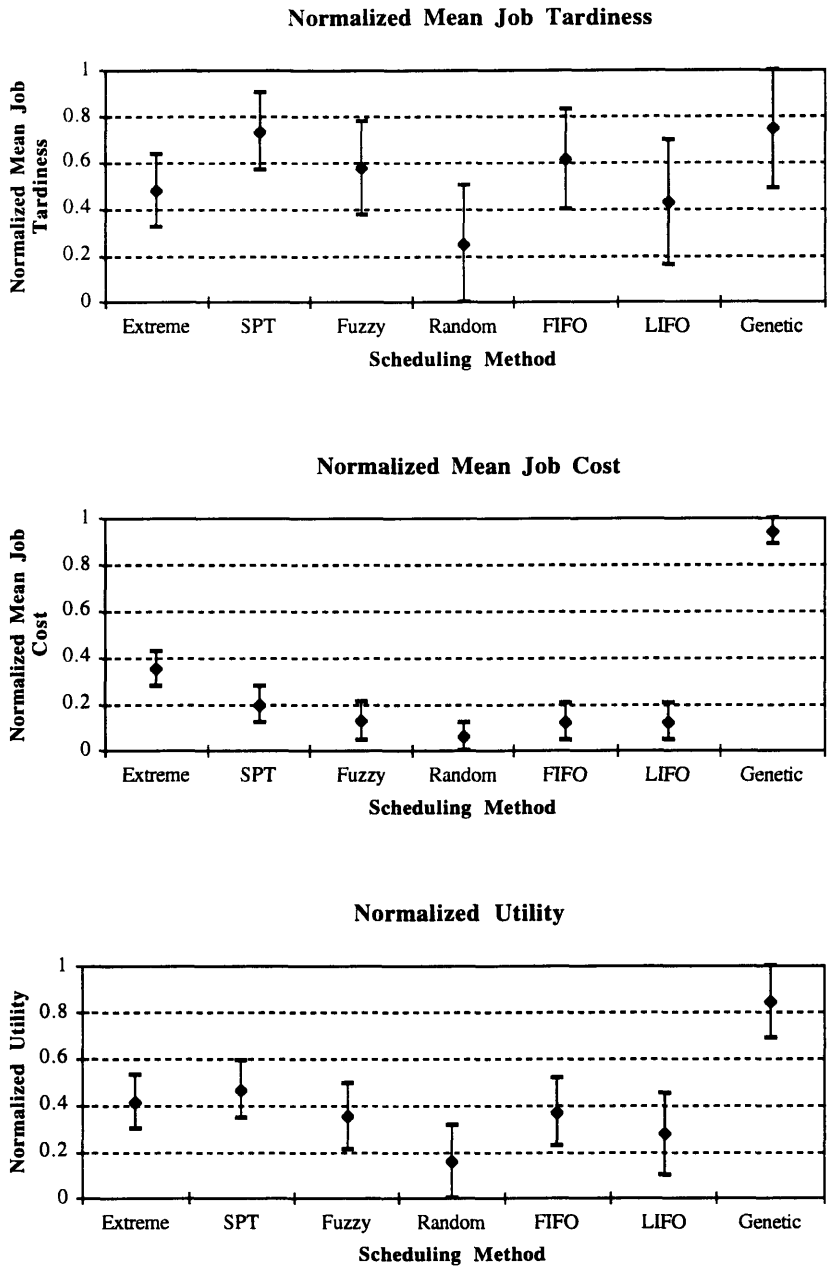


Figure 5.14: Results of Experiment 4.

Expt 5: **Tight** Duedates, **3** Suitable Resources to Process a job operation and with **NO** Resource Breakdowns.

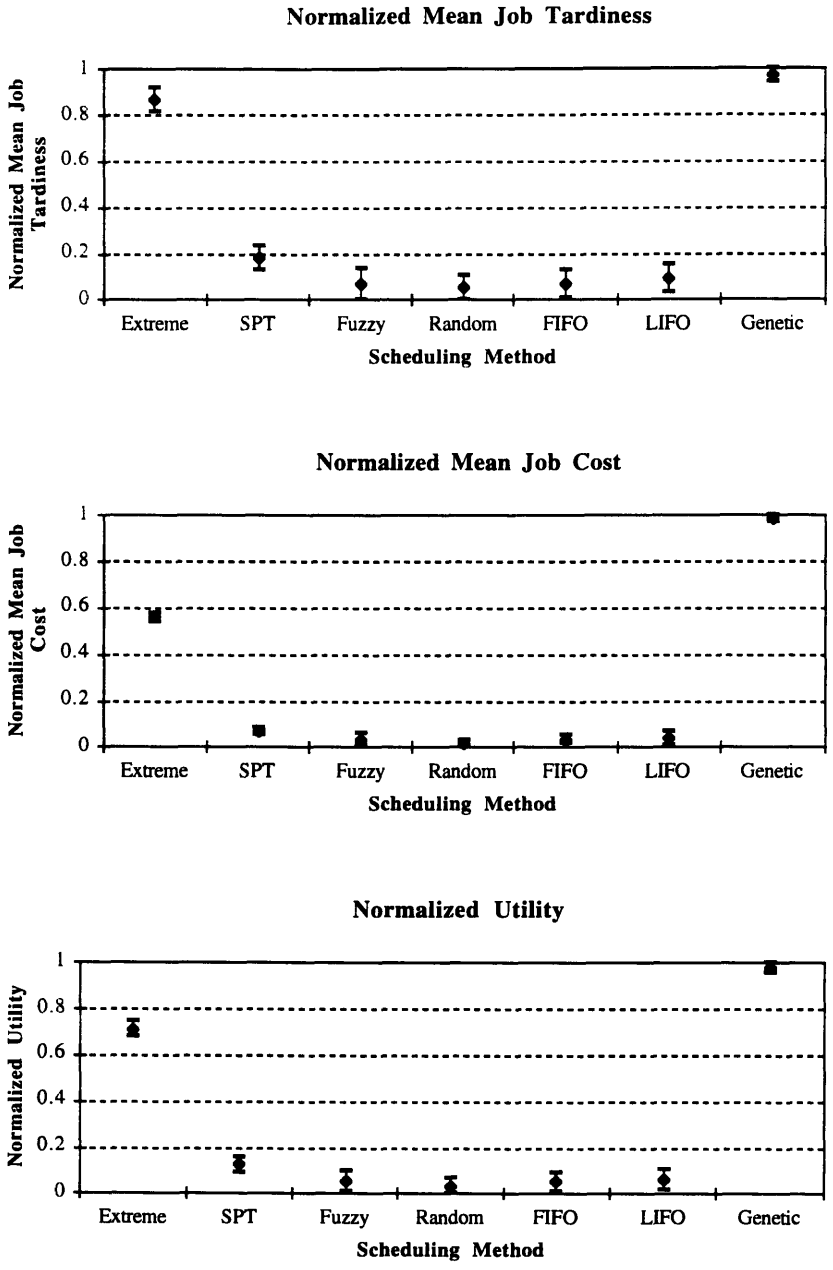


Figure 5.15: Results of Experiment 5.

Expt 6: **Loose** Duedates, **3** Suitable Resources to Process a job operation and with **NO** Resource Breakdowns.

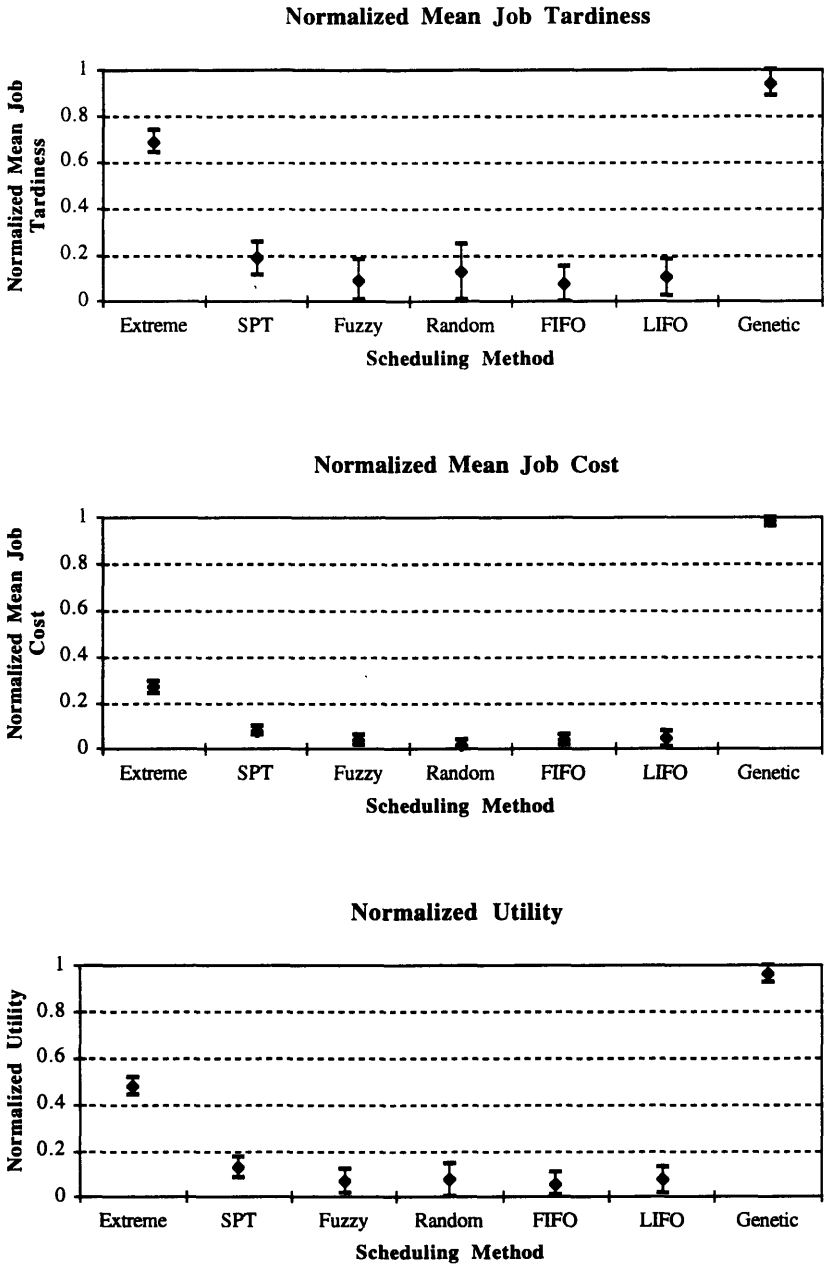


Figure 5.16: Results of Experiment 6.

Expt 7: **Tight** Duedates, **6** Suitable Resources to Process a job operation and with **NO** Resource Breakdowns.

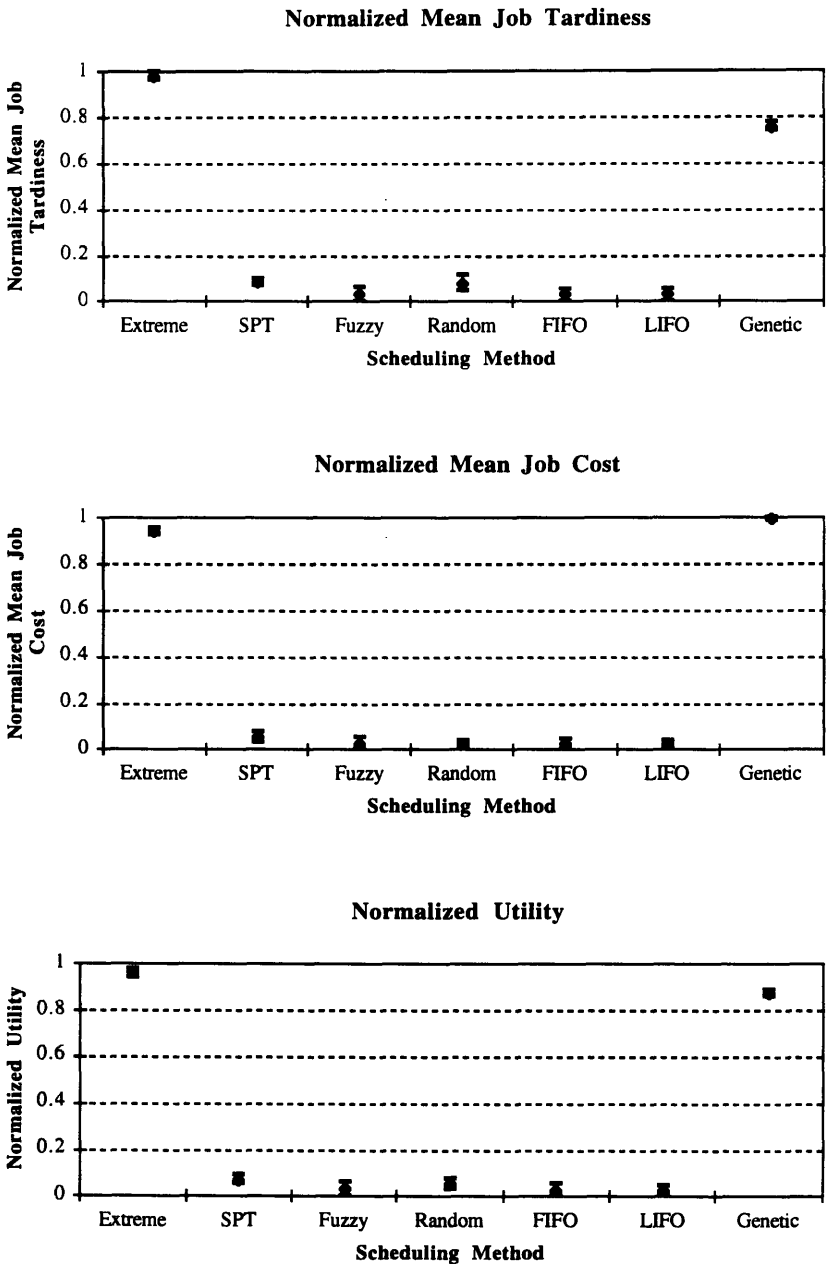


Figure 5.17: Results of Experiment 7.

Expt 8: **Loose** Duedates, **6** Suitable Resources to Process a job operation and with **NO** Resource Breakdowns.

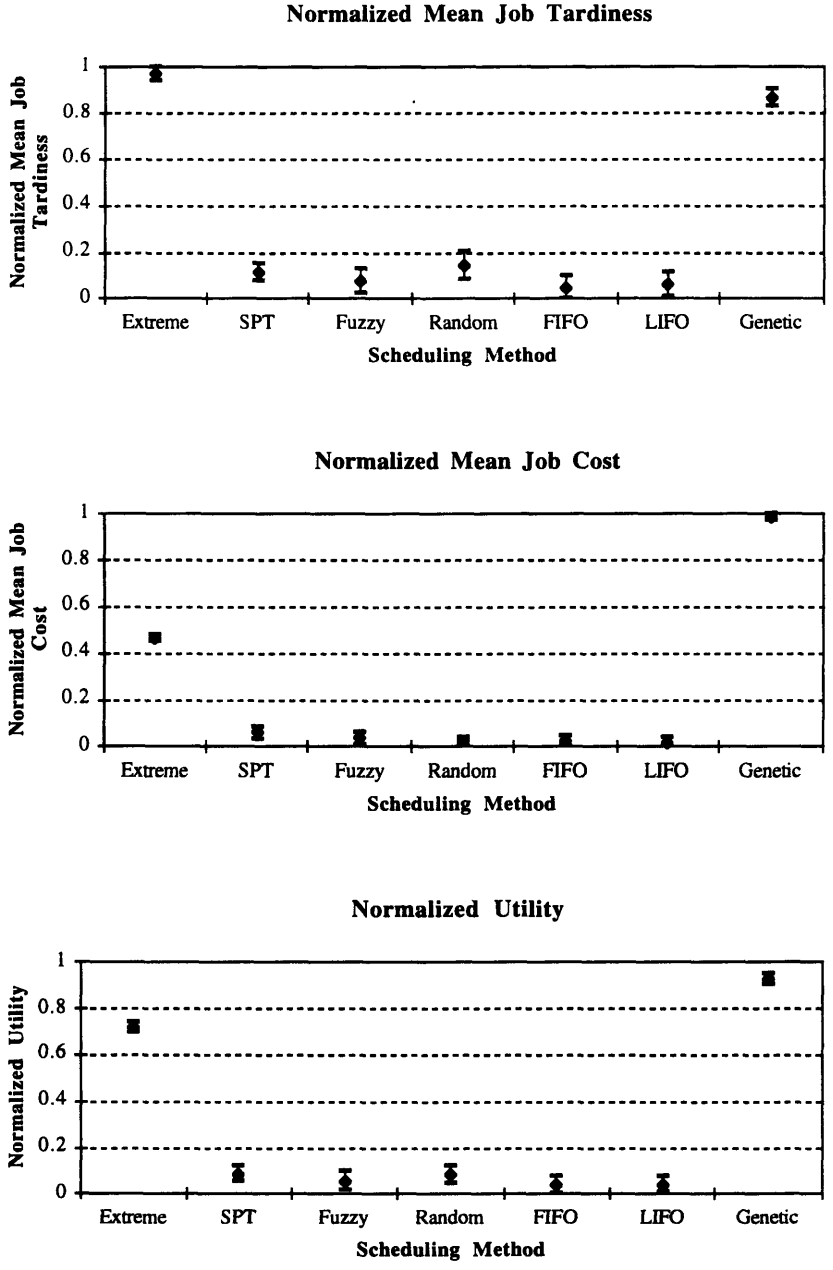


Figure 5.18: Results of Experiment 8.

The Extreme Value scheduling approach was applied to Experiment 2 of the dynamic scheduling experiments (Figure 5.9), incorporating the dh. Simulations were performed for dh=0,1,2,3,4,5 and the results are presented in Figure 5.19. At first glance it appears that there is a deterioration in the results as the dh increases. Intuitively, however, one should expect an increase in the performance as the dh increases.

To reconcile this anomaly, it would be useful to examine the mechanics of the Extreme Value approach closely. In the Extreme Value approach, it is acknowledged that the pool of alternatives is usually large and it would be impractical to evaluate all of them. Therefore, only a sampling of these alternatives are considered. By incorporating the decision horizon, one increases the already large pool of alternatives and by sampling from these pool, adequate representation of the immediate assignments to currently available machines may not be realized. To illustrate, consider once again the example of Figure 3.1. In this example, the immediate assignments to currently available machines are:

R1T1 R2T2

Random schedules will be drawn using R1T1 R2T2 as the beginning portion of the schedules and the potential of the assignment of R1T1 R2T2 will be estimated. By increasing the decision horizon to 1, the alternatives with R1T1 R2T2 as root assignments are

R1T1 R2T2 *R1T3*
R1T1 R2T2 *R2T3*

The concatenation has resulted in two separate "alternatives" with R1T1 R2T2 as root assignments. When the potential of R1T1 R2T2 *R1T3* and R1T1 R2T2 *R2T3* are evaluated, it is the potential of the concatenated alternatives that are estimated and not that of the immediate assignments R1T1 R2T2. It must be pointed out that in a schedule building approach, it is important to estimate the potentials of the immediate assignments and not that of future assignments. It is this inaccurate estimation of potentials of the immediate assignments that results in the deterioration of the performance of the schedules as the decision horizon increases. On the other hand, if exhaustive evaluation of all alternatives are considered, then the incorporation of the decision horizon may lead to an increase in the schedule quality.

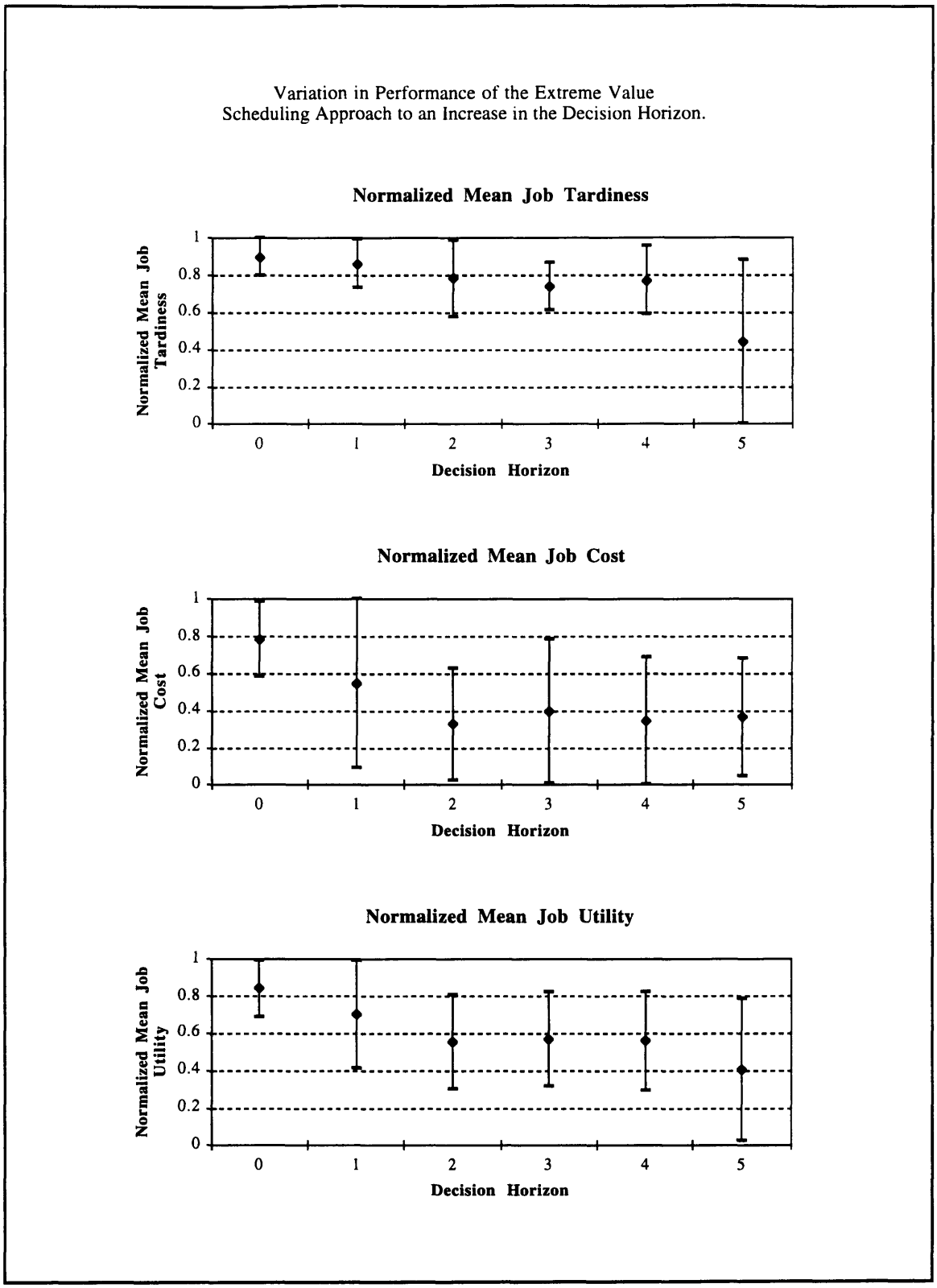


Figure 5.19: Effect of Decision Horizon on the Performance of the Extreme Value Scheduling Approach

Figure 5.20, presents the summary of the results (Figures 5.11 to 5.18). The best two scheduling approaches in terms of the Mean Job Tardiness and Mean Job Cost and the Utility (composite performance measure) are tabulated for each experiment.

A review of the Figure 5.20, reveals that both the Extreme Value and the Genetic Algorithms scheduling methods have generally outperformed all the other scheduling methods studied in this thesis. Specifically, the Genetic Algorithms method has consistently performed best and the Extreme Value approach is consistently second best in terms of Mean Job Cost performance measure. As for the Mean Job Tardiness performance measure, the norm was that the Genetic Algorithms and the Extreme Value approaches finished best and second best respectively. The exception to this norm occurs in Experiments 3,4,7 and 8. In Experiments 3 and 4, the SPT dispatching rule performed better than the Extreme Value Approach. Referring to Figure 5.9, Experiments 3 and 4 correspond to the case where machine breakdowns and repairs are included in addition to the reduced number of suitable resources to process a job operation . However, referring to the utility performance measure (composite of cost and tardiness), the Extreme Value approach still performs second best to the Genetic Algorithms approach. In the case of Experiments 7 and 8, the Extreme Value approach outperforms the Genetic Algorithms approach in terms of the Mean Job Tardiness performance measure. Experiments 7 and 8 refers to the case where no machine breakdowns and repairs are considered and all the machines are assumed capable of processing any job operation.

In addition, reviewing Figures 5.11-5.18, will indicate that the confidence intervals of the performance (the utility) of a scheduling method increases when

- machine breakdowns/ repairs are included
- the number of suitable resources to process a job operation is small.

A comparative study of the computational times required for the various scheduling methods discussed in the previous sections was conducted. This comparison cannot be performed on the basis of computational times required for a scheduling decision made by the various methods. This is so because, the various scheduling methods have different scheduling mechanics and each scheduling decision for the various methods involves different number of scheduling assignments. For example, the dispatching rules (viz,

SPT, FIFO, LIFO and Random) make only one scheduling assignment per scheduling decision. The Extreme Value Method and the Genetic Algorithms Method both involve multiple scheduling assignments per scheduling decision. Therefore the average computational time required to perform a scheduling assignment (rather than a scheduling decision) is a more accurate measurement of computational effort for each of the scheduling methods.

| Expt # | Tardiness | Cost | Utility |
|---|---|---|---|
| 1 | GA<br>Extreme | GA<br>Extreme | GA<br>Extreme |
| 2 | GA<br>Extreme | GA<br>Extreme | GA<br>Extreme |
| 3 | GA<br>SPT | GA<br>Extreme | GA<br>Extreme |
| 4 | GA<br>SPT | GA<br>Extreme | GA<br>Extreme/SPT |
| 5 | GA<br>Extreme | GA<br>Extreme | GA<br>Extreme |
| 6 | GA<br>Extreme | GA<br>Extreme | GA<br>Extreme |
| 7 | Extreme<br>GA | GA<br>Extreme | Extreme<br>GA |
| 8 | Extreme<br>GA | GA<br>Extreme | GA<br>Extreme |

Figure 5.20: Summary of Results

The average computational time required to perform a scheduling assignment for all the scheduling methods discussed previously are based on Experiment 1 of the 8 dynamic scheduling experiments (Figure 5.9). The computational time data for the various scheduling methods is recorded beginning, after the 3000th job-operation was assigned and continued till the 4000th job-operation was assigned (i.e. 1000 scheduling assignments).

This ensures that the computational time data is recorded only after the dynamic job shop has reached steady state in the simulations. Each scheduling method was simulated for 10 different random seeds and the average computational time (in msecs) required for a scheduling assignment, the standard deviation and the 90% confidence intervals are summarized in Figure 5.21. The results indicate that the average computational time for the Extreme Value Method is about an order of magnitude larger than the simple dispatching rules and the Fuzzy Logic approach (which is a combination of dispatching rules). The Genetic Algorithms approach, as seen earlier produced excellent results but requires about two orders of magnitude of computational time compared to the dispatching rules.

| **Method** | **Average** | **Std. Dev.** | **90% Confidence Interval** |
|---|---|---|---|
| Extreme | 144.20 | 5.88 | [140.79, 147.61] |
| Genetic | 1105.64 | 285.08 | [940.39, 1270.88] |
| SPT | 18.03 | 0.49 | [17.74,18.31] |
| FIFO | 17.85 | 0.46 | [17.59,18.12] |
| LIFO | 17.83 | 0.50 | [17.54,18.12] |
| Random | 17.81 | 0.37 | [17.59,18.02] |
| Fuzzy | 19.65 | 0.88 | [19.14, 20.16] |

Figure 5.21: Computational Time required for a Scheduling Assignment (in msecs).

# Conclusions

A survey of current scheduling methods has revealed that most of these methods will not be suitable to industry because they require assumptions that do not reflect reality. In particular assumptions that pertain to multiple criteria scheduling, availability of alternate resources to process job-operations, and the basic dynamic behaviour of job-shop scheduling environments (i.e., random job arrivals, breakdown and repair of machines), are very relevant for the application of scheduling methods to real industrial job shops.

Two scheduling methods, based on Extreme Value Theory (SEVAT) and Genetic Algorithms (GA), were developed in this thesis to bridge the gap between academic research and industrial use. The SEVAT approach is a schedule building approach, while the GA approach is a schedule permutation approach.

Both these approaches were applied to static benchmark job shop scheduling problems (the Muth and Thompson 6x6 and 10x10 problems). The results compare favourably with the optimal solutions and with results published in the literature. In addition, these results were found to be better than those obtained using some common dispatching rules and a scheduling method based on Fuzzy Logic, which is a representative approach of current scheduling research.

A dynamic scheduling problem was designed which incorporates multiple criteria scheduling, with alternate resources to process job-operations, and includes dynamic effects, such as stochastic job arrivals and machine breakdowns/repairs. Three factors were identified and varied between two levels each, to reflect the simulation of varied job shop conditions encountered in reality. The SEVAT and GA approaches were applied to the factorial design of experiments . The results were once again compared with some common dispatching rules and the Fuzzy Logic scheduling method. The results overwhelmingly indicate that the SEVAT and GA approaches produce better performance than the other methods.

# References


1. Aarts, E.H.L., P.J.M. Van Laarhoven, J.K. Lenstra, and N.L. Ulder, 1994, "A Computational Study of Local Search Algorithms for Job Shop Scheduling," ORSA Journal of Computing, Vol. 6, No. 2, pp. 118-125.

2. Anderson, E.J., and Nyirenda, J.C., 1990, "Two New Rules to Minimize Tardiness in a Job-Shop," International Journal of Production Research, Vol. 28, No. 12, pp. 2277-2292.

3. Antonisse, J., 1989, "A New Interpretation of Schema Notation that Overturns the Binary Encoding Constraint," Proceedings of the Third International Conference on Genetic Algorithms, pp. 86-91.

4. Arnell, N.W., Beran, M., and Hosking, J.R.M., 1986, "Unbiased Plotting Positions for the General Extreme Value Distribution," Journal of Hydrology, Vol. 86, pp. 59-69.

5. Bagchi, S., S. Uckun, .Y. Miyabe, and K. Kawamura, 1991, "Exploring Problem-Specific Recombination Operators for Job Shop Scheduling," Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 10-17.

6. Baker, K.R., and G.D. Scudder, 1990, "Sequencing with Earliness and Tardiness Penalties: A Review," Operations Research, Vol. 38, No. 1, pp. 22-36.

7. Bardsley, W.E., 1989, "Graphical Estimation of Extreme Value Prediction Functions," Journal of Hydrology, Vol. 110, pp. 315-321.

8. Benton, W.C., 1993, "Time and Cost based Priorities for Job Shop Scheduling," International Journal of Production Research, Vol. 31, No. 7, pp. 1509-1519.

9. Blackstone, J.H., 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 Production Research, Vol. 20, No. 1, pp. 27-45.

10. Booker, L., 1987, "Improving Search in Genetic Algorithms," Genetic Algorithms and Simulated Annealing, L. Davis (ed.), Morgan Kaufman Publishers, pp. 61-73.

11. Braun, H., 1990, "On Travelling Salesman Problems by Genetic Algorithms," Parallel Problem Solving from Nature, Proceedings from 1st Workshop, PPSN I, Dortmund, FRG, pp. 129-133.

12. Cartwright, H.M., and G.F. Mott, 1991, "Looking Around: Using Clues from the Data Space to Guide Genetic Algorithms Searches," Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 108-114.

13. Castillo, E., 1988, Extreme Value Theory in Engineering, Academic Press, San Diego, CA.

14. Cenna, A.A., and M.T. Tabucanon, 1991, "Bicriterion Scheduling Problem in a Job Shop with Parallel Processors, International Journal of Production Economics, Vol. 25, No. 1, pp. 95-101.

15. Chang, Y.L., and R.S. Sullivan, 1990, "Schedule Generation in a Dynamic Job Shop," International Journal of Production Research, Vol. 28, No. 1, pp. 65-74.

16. Chang, Y.L., H. Matsuo, and R.S. Sullivan, 1989, "Bottle-neck based Beam Search for Job Scheduling in a Flexible Manufacturing System," International Journal of Production Research, Vol. 27, No. 11, pp. 1949-1961.

17. Chen, T., 1994, "The Use of Genetic Algorithms in Production Scheduling," S.B. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology.

18. Chock, D.P., P.S. Sluchak, 1986, "Estimating Extreme Values of Air Quality Data using different Fitted Distributions," Atmospheric Environment, Vol. 20, No. 5, pp. 989-993.

19. Chryssolouris, G., 1986, "On Decision Making in Manufacturing Systems," Proceedings of the Japan-USA Symposium on Flexible Automation, ASME, Osaka, Japan (July 14-18, 1986), pp. 459-470.

20. Chryssolouris, G., 1987, "MADEMA: An Approach to Intelligent Manufacturing Systems," CIM Review, Vol. 3, No. 3 (Spring 1987), pp. 11-17.

21. Chryssolouris, G., 1992, "Manufacturing Systems: Theory and Practice," Springer Verlag, New York.

22. Chu, C., M.C. Portmann and J.M. Proth, 1992, "Splitting-Up Approach to Simplify Job-Shop Scheduling," International Journal of Production Research, Vol. 30, No. 4, pp. 859-870.

23. Cleveland, G.A., and S.F. Smith, 1989, "Using Genetic Algorithms to Schedule Flow Shop Releases," Proceedings of the 3rd International Conference on Genetic Algorithms, pp. 160-169.

24. Csorgo, S., and Mason D.M., 1987, "Simple Estimators of the Endpoint of a Distribution," Extreme Value Theory, Proceedings of a conference held in Oberwolfach, pp. 132-147.

25. Custodio, L.M.M., J.J.S. Sentieiro, and C.F.G Bispo, 1994, "Production Planning and Scheduling Using a Fuzzy Decision System," IEEE Transactions on Robotics and Automation, Vol. 10, No. 2, pp. 160-168.

26. Day, J.E., and M.P. Hottenstein, 1970, "Review of Sequencing Research," Naval Research Logistics Quarterly, Vol. 17, pp. 11-39.

27. De Jong, K.A., 1992, "Are Genetic Algorithms Function Optimizers?," Proceedings of the Second Conference on Parallel Problem Solving from Nature, Brussels, pp. 3-14.

28. De Jong, K.A., and W.M. Spears, 1989, "Using Genetic Algorithms to Solve NP-Complete Problems," Proceedings of the Third International Conference on Genetic Algorithms, pp. 124-132.

29. Enns, S.T., 1993, "Job Shop Flowtime Prediction and Tardiness Control using Queueing Analysis," International Journal of Production Research, Vol. 31, No. 9, pp. 2045-2057.

30. Eschelman, L.J., and J.D. Schaffer, 1991, "Preventing Premature Convergence in Genetic Algorithms by Preventing Incest," Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 115-122.

31. Filipic, B., 1992, "Enhancing Genetic Search to Schedule a Production Unit," 10th European Conference on Artificial Intelligence, B. Neumann (ed.), John Wiley & Sons.

32. Fogarty, T.C., 1989, "Varying the Probability of Mutation in the Genetic Algorithm," Proceedings of the Third International Conference on Genetic Algorithms, pp. 104-109.

33. Foo, S.Y., Y. Takefuji, and H. Szu, 1994, "Job-Shop Scheduling Based on modified Tank-Hopfield Linear Programming Networks," Engineering Applications of Artificial Intelligence, Vol. 7, No. 3, pp. 321-327.

34. Friesleben, B., and M. Hartfelder, 1993, "Optimization of Genetic Algorithms by Genetic Algorithms," Artificial Neural Nets and Genetic Algorithms, R.F. Albrecht, C.R. Reeves and N.C. Steele (eds.), Proceedings of the International Conference in Innsbruck, Austria, pp. 392-399.

35. Gee, E.S., and C.H. Smith, 1993, "Selection Allowance Policies for Improved Job Shop Performance," International Journal of Production Research, Vol. 31, No. 8, pp. 1839-1852.

36. Gershwin, S.B., 1994, "Manufacturing Systems Engineering," Prentice Hall, New Jersey.

37. Grabot, B., and L. Geneste, 1994, "Dispatching Rules in Scheduling: A Fuzzy Approach," International Journal of Production Research, Vol. 32, No. 4, pp. 903-915.

38. Graves, S.C., 1981, "A Review of Production Scheduling," Operations Research, Vol. 29, pp. 646-675.

39. Guida, M. and Longo, M., 1988, "Estimation of Probability Tails based on Generalized Extreme Value Distributions," Reliability Engineering and System Safety, Vol. 20, pp 219-242.

40. Gumbel, E.J., 1958, Statistics of Extremes, Columbia University Press, New York.

41. He, Z., T. Yang and D.E. Deal, 1993, "Multiple-Pass Heuristic Rule for Job Scheduling with Due Dates," International Journal of Production Research, Vol. 31, No. 11, pp. 2677-2692.

42. Hoitomt, D.J., P.B. Luh, and K.R. Pattipati, 1993, "A Practical Approach to Job Shop Scheduling Problems," IEEE Transactions on Robotics and Automation, Vol. 9, No. 1, pp. 1-13.

43. Hosking, J.R.M., Wallis, J.R., and Wood, E.F., 1985, "Estimation of the Generalized Extreme Value Distribution by the Method of Probability Weighted Moments," Technometrics, Vol 27, No. 3, pp. 251-261.

44. Husbands, P., and F. Mill, 1991, "Simulated Co-Evolution as the Mechanism for Emergent Planning and Scheduling," Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 264-270.

45. Itoh, K., D. Huang, and T. Enkawa, 1993, "Twofold Look-ahead Search for Multi-Criterion Job Shop Scheduling," International Journal of Production Research, Vol. 31, No. 9, pp. 2215-2234.

46. Jog, P., J.Y. Suh, D. Van Gucht, 1989, "The Effects of Population Size, Heuristic Crossover and Local Improvement on a Genetic Algorithm for the Traveling Salesman Problem," Proceedings of the Third International Conference on Genetic Algorithms, pp. 110-115.

47. Kanet, J.J., and V. Sridharan, 1991, "ProGenitor: A Genetic Algorithm for Production Scheduling," Working Paper, Clemson University.

48. Kanet, J.J., and Z. Zhou, 1992, "A Decision Theory Approach to Priority Dispatching for Job Shop Scheduling," Production and Operations Management Conference 1992 in Orlando.

49. Kannan, V.R., and S. Ghosh, 1993, "Evaluation of the Interaction between Dispatching Rules and Truncation Procedures in Job-Shop Scheduling," International Journal of Production Research, Vol. 31, No. 7, pp. 1637-1654.

50. Kaplan, A.C., and A.T. Unal, 1993, "A Probabilistic Cost-Based Due Date Assignment Model for Job Shops," International Journal of Production Research, Vol. 31, No. 12, pp. 2817-2834.

51. Karsiti, M.N., J.B. Cruz, and J.H. Mulligan, 1992, "Simulation Studies of Multilevel Job Shop Scheduling Using Heuristic Dispatching Rules," Journal of Manufacturing Systems, Vol. 11, No.5, pp. 346-358.

52. Kim, Y.D., 1990, "A Comparison of Dispatching rules for Job Shops with Multiple Identical Jobs and Alternative Routings," International Journal of Production Research, Vol. 28, No. 5, pp. 953-962.

53. Kosko, B., 1992, "Neural Networks and Fuzzy Systems," Prentice Hall, New Jersey.

54. Law, A.M., and M.G. McComas, 1989, "Pitfalls to Avoid in the Simulation of Manufacturing Systems," Industrial Engineering, Vol. 31, pp. 28-31, 69.

55. Law, A.M., and W.D. Kelton, 1991, "Simulation Modelling and Analysis," Mcgraw Hill, New York.

56. Laycock, P.J., R.A. Cottis, P.A. Scarf, 1990, "Extrapolation of Extreme Pit Depths in Space and Time," Journal of the Electrochemical Society, Vol. 137, No. 1, pp. 64-69.

57. Lee, C.C., 1990, "Fuzzy Logic in Control Systems: Fuzzy Logic Controller-Part I. IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 2, pp. 404-418.

58. Leon, V.J., S.D. Wu and R.H. Storer, 1994, "A Game-Theoretic Control Approach for Job Shops in the Presence of Disruptions," International Journal of Production Research, Vol. 32, No. 6, pp. 1451-1476.

59. Li, R.K., Y.T. Shyu and S. Adiga, 1993, "A Heuristic Rescheduling Algorithm for Computer-based Production Scheduling Systems," International Journal of Production Research, Vol. 31, No. 8, pp. 1815-1826.

60. Luh, P.B., and D.J. Hoitomt, 1993, "Scheduling of Manufacturing Systems using the Lagrangian Relaxation Technique," IEEE Transactions on Automatic Control, Vol. 38, No. 7, pp. 1066-1079.

61. 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 Production Research, Vol. 31, No. 1, pp. 59-79.

62. Mamdani, E.H., J.J. Ostergaard and E. Lembessis, 1984, "Use of Fuzzy Logic for Implementing Rule-Based Control of Industrial Processes," TIMS/Studies in the Management Sciences, Vol. 20, pp. 429-445.

63. Mattfeld, D.C., H. Kopfer, and C. Bierwirth, 1994, "Control of Parallel Population Dynamics by Social-Like Behaviour of GA-Individuals," Proceedings of the Third Conference on Parallel Problem Solving from Nature, Jerusalem, pp. 16-25.

64. McKay, K.N., F.R. Safayeni, and J.A. Buzacott, 1988, "Job-Shop Scheduling Theory: What is Relevant?" Interfaces, Vol. 18, pp. 84-90.

65. Miyashita, K., and K. Sycara, 1994, "Adaptive Case-Based Control of Schedule Revision," in Intelligent Scheduling, M. Zweben and M.Fox, eds, Morgan Kaufman Publishers, pp. 291-308.

66. Muhlenbein, H., 1989, "Parallel Genetic Algorithms, Population Genetics and Combinatorial Optimization," Proceedings of the Third International Conference on Genetic Algorithms, pp. 416-421.

67. Muhlenbein, H., 1992, "How Genetic Algorithms Really Work I: Mutation and Hillclimbing," Proceedings of the Second Conference on Parallel Problem Solving from Nature, Brussels, pp. 15-26.

68. Musser, K.L., J.S. Dhingra, and G.L. Blankenship, 1993, "Optimization Based Job Shop Scheduling," IEEE Transactions on Automatic Control, Vol. 38, No. 5, pp. 808-813.

69. Muth, J.F., and G.L. Thompson, 1963, Industrial Scheduling, Prentice Hall, Englewood Cliffs, new Jersey.

70. Nakano, R., and T. Yamada, 1991, "Conventional Genetic Algorithm for Job Shop Problems," Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 474-479.

71. Nasr, N., and E.A. Elsayed, 1990, "Job Shop Scheduling with Alternative Machines," International Journal of Production Research, Vol. 28, No. 9, pp. 1595-1609.

72. Ovacik, I.M., and R. Uzsoy, 1994, "Exploiting Shop Floor Status Information to Schedule Complex Job Shops," Journal of Manufacturing Systems, Vol. 13, No.2, pp. 73-83.

73. Parunak, H.V.D., 1991, "Characterizing the Manufacturing Scheduling Problem," Journal of Manufacturing Systems, Vol. 10, No. 3, pp. 241-259.

74. Polant, B., 1992, "Genetic Algorithms: A Introductory Study," Working Paper, Michigan State University.

75. Prinetto, P., M. Rebaudengo and M.S. Reorda, 1993, "Hybrid Genetic Algorithms for the Travelling Salesman Problem," Artificial Neural Nets and Genetic Algorithms, R.F. Albrecht, C.R. Reeves and N.C. Steele (eds.), Proceedings of the International Conference in Innsbruck, Austria, pp. 559-566.

76. Qiao, W.Z., W.P. Zhuang, and T,H, Heng, 1992, "A Rule Self-Regulating Fuzzy Controller," Fuzzy Sets and Systems, Vol. 47, pp. 13-21.

77. Ragatz, G.L., and V.A. Mabert, 1988, "An Evaluation of Order Release Mechanisms in a Job-Shop Environment," Decision Sciences, Vol. 19, pp. 167-189.

78. Raghu, T.S., and C. Rajendran, 1993, "An Efficient Dynamic Dispatching Rule for Scheduling in a Job Shop," International Journal of Production Economics, Vol. 32, No. 3, pp. 301-313.

79. Ramasesh, R., 1990, "Dynamic Job Shop Scheduling: A Survey of Simulation Research," OMEGA: International Journal of Management Science, Vol. 18, No. 1, pp. 43-57

80. Reeves, C., 1993, "Diversity and Diversification in Genetic Algorithms: Some Connections with Tabu Search," Artificial Neural Nets and Genetic Algorithms, R.F. Albrecht, C.R. Reeves and N.C. Steele (eds.), Proceedings of the International Conference in Innsbruck, Austria, pp. 344-351.

81. Reeves, C., and H. Karatza, 1993, "Dynamic Sequencing of a Multi-Processor System: A Genetic Algorithm Approach," Artificial Neural Nets and Genetic Algorithms, R.F. Albrecht, C.R. Reeves and N.C. Steele (eds.), Proceedings of the International Conference in Innsbruck, Austria, pp. 491-495.

82. Rodammer, F.A., and K.P.White, 1988, "A Recent Survey of Production Scheduling," IEEE Transactions on Systems, Man, and Cybernetics, Vol. 18, No. 6, pp. 841-851.

83. Scudder, G.D., T.R. Hoffmann and T.R. Rohleder, 1993, "Scheduling with forbidden early shipments: alternative performance criteria and conditions," International Journal of Production Research, Vol. 31, No. 10, pp. 2287-2305.

84. Sim, S.K., K.T. Yeo and W.H. Lee, 1994, "An Expert Neural Network System for Dynamic Job Shop Scheduling," International Journal of Production Research, Vol. 32, No. 8, pp. 1759-1773.

85. Smith, J.A., 1987, "Estimating the Upper Tail of Flood Frequency Distribution," Water Resources Research, Vol. 23, No. 8, pp. 1557-1666.

86. Smith, R.L., 1986, "Extreme Value Statistics and Reliability Applications," Reliability Engineering, Vol. 15, No. 3, pp. 161-170.

87. Sun, D., and L. Lin, 1994, "A Dynamic Job Shop Scheduling Framework: A Backward Approach," International Journal of Production Research, Vol. 32, No. 4, pp. 967-985.

88. Sycara, K., S.F. Roth, N. Sadeh, and M.S. Fox, 1991, "Distributed Constrained Heuristic Search," IEEE Transactions on Systems, Man and Cybernetics, Vol. 21, No. 6, pp. 1446-1461.

89. Syswerda, G, 1989, "Uniform Crossover in Genetic Algorithms," Proceedings of the Third International Conference on Genetic Algorithms, pp. 2-9.

90. Syswerda, G., and J. Palmucci, 1991, "The Application of Genetic Algorithms to Resource Scheduling," Proceedings of the Fourth International Conference on Genetic Algorithms, pp. 502-508.

91. Taillard, E.D., 1994, "Parallel Taboo Search Techniques for the Job Shop Scheduling Problem," ORSA Journal of Computing, Vol. 6, No. 2, pp. 108-117.

92. Tamaki, H., and Y. Nishikawa, 1992, "A Parallel Genetic Algorithm based on a Neighbourhood Model and its Application to Job Shop Scheduling," Proceedings

of the Second Conference on Parallel Problem Solving from Nature, Brussels, pp. 573-582.

93. Thierens, D., and D. Goldberg, 1994, " Convergence Models of Genetic Algorithm Selection Schemes," Proceedings of the Third Conference on Parallel Problem Solving from Nature, Jerusalem, pp. 119-129.

94. Turksen, I.B., T. Yurtsever, and K. Demirli, 1993, "Fuzzy Expert System Shell for Scheduling," Proceedings of the SPIE - The International Society for Optical Engineering, Vol. 2061, pp. 308-319.

95. Uckun, S., S. Bagchi, K. Kawamura and Y. Miyabe, 1993, "Managing Genetic Search in Job Shop Scheduling," IEEE Expert, pp. 15-24.

96. Udo, G., 1993, "An Investigation of Due-Date Assignment using Workload Information of a Dynamic Shop," International Journal of Production Economics, Vol. 29, pp. 89-101.

97. Ulder, N.L.J, E.H.L. Aarts, H.J. Bandelt, P.J.M. van Laarhoven, and E. Pesch, 1990, "Genetic Local Search Algorithms for the Traveling Salesman Problem," Parallel Problem Solving from Nature, Proceedings from 1st Workshop, PPSN I, Dortmund, FRG, pp. 109-116.

98. Van Ryzin, G.J., S.X. Lou, and S.B. Gershwin, 1991, "Scheduling Job Shops with Delays," International Journal of Production Research, Vol. 29, No. 7, pp. 1407-1422.

99. Vancheeswaran, R., and M.A. Townsend, 1993, "A Two-stage Heuristic Procedure for Scheduling Job Shops," Journal of Manufacturing Systems, Volume 12, No. 4, pp. 315-325.

100. Vig, M.M., and K.J. Dooley, 1991, "Dynamic Rules for Due-Date Assignment," International Journal of Production Research, Vol. 29, No. 7, pp. 1361-1377.

101. Watanabe, T., H. Tokumaru and Y. Hashimoto, 1993, "Job Shop Scheduling Using Neural Networks," Control Engineering Practice, Vol. 1, No. 6, pp. 957-961.

102. Whitley, D., 1989, "The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best," Proceedings of the Third International Conference on Genetic Algorithms, pp. 116-121.

103. Whitley, D., T. Starkweather and D. Fuquay, 1989, "Scheduling Problems and Traveling Salesmen: The Genetic Edge Recombination Operator," Proceedings of the Third International Conference on Genetic Algorithms, pp. 133-140.

104. Willems, T.M., and J.E. Rooda, 1994, "Neural Networks for Job-Shop Scheduling," Control Engineering Practice, Vol. 2, No. 1, pp. 31-39.

105. Yamada, T., and R. Nakano, 1992, "A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems," Proceedings of the Second Conference on Parallel Problem Solving from Nature, Brussels, pp. 281-290.

106. Ye, M.H., and G.B. Williams, 1991, "Expediting Heuristic for the Shortest Processing Time Dispatching Rule," International Journal of Production Research, Vol. 29, No. 1, pp. 209-213.

107. Zeestraten, M.J., 1990, "The Look Ahead Dispatching Procedure," International Journal of Production Research, Vol. 28, No. 2, pp. 369-384.

108. Zhou, D.N., V. Cherkassky, T.R. Baldwin, D.E. Olson, 1991, "A Neural Network Approach to Job-Shop Scheduling," IEEE Transactions on Neural Networks, Vol. 2, No. 1, pp. 175-179.

# Appendix A

Extreme Value Theory, attempts to derive an approximate mathematical expression of the probability tail. An approach commonly known as the Classical Extreme Value Theory, expresses the probability tail in terms of 2 parameters known as the location and scale parameters. An alternative approach, commonly known as the Generalized Extreme Value Theory, does the same with 3 parameters, the former two plus an exponent.

The parameters of the extreme value distributions are estimated assuming that the initial distribution of the data is unknown and that the estimation of the parameters is performed over the population of the extrema and not the population of the data. To achieve this, a sample of size nN is drawn from the initial population and is partitioned into N groups of n elements. From each group, the maximum (or minimum; in the rest of the analysis, we will refer to the maximum) datum $x_{(n)}$ is determined. In this way a random sample X: ($X_1$, $X_2$, ..., $X_N$) of size N from the population of maxima over n elements is generated [Guida and Longo, 1988].

The approximate distribution of maxima based on classical extreme value theory is derived as

$$F_1(x) = \exp[-\exp[-(x-a_n)/b_n]] \qquad (A.1)$$

where $a_n$ is the location parameter and $b_n$ is the scale parameter. Based on the maximum likelihood method (ML), the estimates for $a_n$ and $b_n$ are found to be

$$\hat{b}_n = \overline{X} - \frac{\sum_{i=1}^{N} X_i \exp(-X_i/\hat{b}_n)}{\sum_{i=1}^{N} \exp(-X_i/\hat{b}_n)} \qquad (A.2)$$

$$\hat{a}_n = \hat{b}_n \ln\left[ N^{-1} \sum_{i=1}^{N} \exp\left(-X_i/\hat{b}_n\right) \right] \tag{A.3}$$

where $\overline{X} = N^{-1} \sum_{i=1}^{N} X_i$. The equations above can be solved by the following iteration procedure

$$b_n(2j+2) = b_n(2j+1) + [b_n(2j) - b_n(2j+1)]/3 \tag{A.4}$$

where $b_n(2j)$ is a tentative solution at step j to be inserted at the right hand side of equation (1) and $b_n(2j+1)$ is the corresponding left hand side. The parameter $b_n$ is initialized in the iterations as

$$b_n(0) = N^{-1} \left\{ \sum_{i=1}^{N} X_i + \sum_{i=1}^{N} X_i \ln\left[\frac{i}{N+1}\right] \right\} \tag{A.5}$$

The termination of the iteration is controlled by the difference between the solutions at two consecutive steps. The final value of the iteration, namely the parameter $b_n$ is then introduced into equation (A.2) to yield the other parameter $a_n$ [Guida and Longo, 1988].

In the case of the Generalised Extreme Value Theory, the approximate distribution of the maxima taken from an unknown initial population F(x) is [Guida and Longo, 1988]

$$F_2(x) = \exp\left\{ -\exp\left[ \frac{-x_n^v - a_n'}{c_n'} \right] \right\} \tag{A.6}$$

The maximum likelihood estimates of the three parameters, v,$a_n'$ and $c_n'$ are those which maximize the following log likelihood L:

$$L(X:a_n',c_n',v) = N \ln v - N \ln c_n' - \sum_{i=1}^{N} \frac{X_i^v - a_n'}{c_n'} + (v-1) \sum_{i=1}^{N} \ln X_i - \sum_{i=1}^{N} \exp\left[ \frac{-\left(X_i^v - a_n'\right)}{c_n'} \right] \tag{A.7}$$

Expressions for these parameters are:

$$\hat{c}_n' = \overline{X^{\hat{v}}} - \frac{\sum_{i=1}^{N} X_i^{\hat{v}} \exp\left(\frac{-X_i^{\hat{v}}}{\hat{c}_n'}\right)}{\sum_{i=1}^{N} \exp\left(\frac{-X_i^{\hat{v}}}{\hat{c}_n'}\right)} \tag{A.8}$$

$$\hat{a}_n' = -\hat{c}_n' \ln\left[N^{-1} \sum_{i=1}^{N} \exp\left(\frac{-X_i^{\hat{v}}}{\hat{c}_n'}\right)\right] \tag{A.9}$$

$$\hat{v} = N \left\{\sum_{i=1}^{N} \left\{\ln X_i + \left[\frac{X_i^{\hat{v}} \ln X_i}{\hat{c}_n'}\right]\left[\exp\left[\frac{-\left(X_i^{\hat{v}} - a_n'\right)}{c_n'}\right] - 1\right]\right\}\right\}^{-1} \tag{A.10}$$

Simulation results show that the Generalized Extreme Value Theory estimator is a more efficient extrapolation estimator of probability tails than the Classical Extreme Value Theory [Guida and Longo, 1988].

An alternative approach to the estimation of the probability tails, besides the analytical approaches described above is to use graphical techniques. One such technique using a three point graphical estimation procedure will be described below [Bardsley, 1989]. This procedure is used in conjunction with the Type 2 or Type 3 Extreme Value distributions. The distribution functions of the Type 2 and Type 3 Extreme Value distributions are expressed as:

$$F(x) = \exp\left\{-\left(\frac{x-w}{\delta}\right)^{\frac{1}{k}}\right\} \qquad k<0,\ x\geq w \qquad \text{Type 2} \tag{A.11}$$

$$F(x) = \exp\left\{-\left(\frac{\xi-x}{\sigma}\right)^{\frac{1}{k}}\right\} \qquad k>0,\ x\leq \xi \qquad \text{Type 3} \tag{A.12}$$

where w and $\xi$ are location parameters, k is the shape parameter and $\delta$ and $\sigma$ are scale parameters. The Type 1 distribution (Gumbel distribution) is a limit case for both Type 2 and Type 3 distributions for $k \rightarrow 0$. Also let

$$y = -\ln\{-\ln[F(x)]\} \tag{A.13}$$

then, the unknown parameters are estimated using three x-y points, viz, $x_1,y_1,x_2,y_2,x_3,y_3$, such that $x_1 < x_2 < x_3$. The type of the function is determined from the following expression:

$$\frac{y_2 - y_1}{x_2 - x_1} - \frac{y_3 - y_1}{x_3 - x_1} \tag{A.14}$$

Equation (A.14) gives positive and negative values for Type 2 and Type 3 distributions respectively. For the Type 2 case, the three parameters of the prediction function can be written in terms of the three points as

$$\delta = \frac{x_3 - x_1}{\exp(-ky_3) - \exp(-ky_1)} \tag{A.15}$$

$$w = x_1 - \delta \exp(-ky_1) \tag{A.16}$$

with k being obtained as the specific value, $k_2$, such that $H(k_2)=0$, where H(k) is defined as

$$H(k) = \frac{1 - \exp[-k(y_2 - y_1)]}{x_2 - x_1} - \frac{1 - \exp[-k(y_3 - y_1)]}{x_3 - x_1} \tag{A.17}$$

Moreover, $k_2$ is a unique value located within the interval

$$\frac{\ln\left(\frac{x_2 - x_1}{x_3 - x_1}\right)}{y_3 - y_2} < k_2 < 0 \tag{A.18}$$

For the Type 3 case, the corresponding expressions are

$$\sigma = \frac{x_3 - x_1}{\exp(-ky_1) - \exp(-ky_3)} \tag{A.19}$$

$$\xi = x_1 + \sigma \exp(-ky_1) \tag{A.20}$$

with k being obtained as the specific value $k_3$ giving $H(k_3)=0$ and $k_3$ is a unique value within the interval

$$0 < k_3 < -\ln\left(\frac{1 - \frac{x_2 - x_1}{x_3 - x_1}}{y_2 - y_1}\right) \qquad \text{(A.21)}$$

The three points may or may not coincide with the existing data points. To properly define the curvature of the data plot, $x_1$ and $x_3$ are likely to be located near the lower and upper limits of the data range. The point $x_2$ and $y_2$ will be best located somewhere in the middle region of the data.

# Appendix B

| | | Job Number | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| Task Number | 1 | 3 | 2 | 3 | 2 | 3 | 2 |
| | 2 | 1 | 3 | 4 | 1 | 2 | 4 |
| | 3 | 2 | 5 | 6 | 3 | 5 | 6 |
| | 4 | 4 | 6 | 1 | 4 | 6 | 1 |
| | 5 | 6 | 1 | 2 | 5 | 1 | 5 |
| | 6 | 5 | 4 | 5 | 6 | 4 | 3 |

Figure B.1: Precedence Constraints (Machine requirements for each task) for the 6x6 problem

| | | Job Number | | | | | |
|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 |
| Task Number | 1 | 1 | 8 | 5 | 5 | 9 | 3 |
| | 2 | 3 | 5 | 4 | 5 | 3 | 3 |
| | 3 | 6 | 10 | 8 | 5 | 5 | 9 |
| | 4 | 7 | 10 | 9 | 3 | 4 | 10 |
| | 5 | 3 | 10 | 1 | 8 | 3 | 4 |
| | 6 | 6 | 4 | 7 | 9 | 1 | 1 |

Figure B.2: Process Times for each Task for the 6x6 Problem

| | | Job Number | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Task Number | 1 | 1 | 1 | 2 | 2 | 3 | 3 | 2 | 3 | 1 | 2 |
| | 2 | 2 | 3 | 1 | 3 | 1 | 2 | 1 | 1 | 2 | 1 |
| | 3 | 3 | 5 | 6 | 1 | 2 | 6 | 4 | 2 | 4 | 3 |
| | 4 | 4 | 10 | 3 | 5 | 6 | 4 | 3 | 6 | 6 | 7 |
| | 5 | 5 | 4 | 9 | 7 | 4 | 9 | 7 | 5 | 3 | 9 |
| | 6 | 6 | 2 | 6 | 9 | 5 | 10 | 6 | 7 | 10 | 10 |
| | 7 | 7 | 7 | 8 | 8 | 9 | 1 | 10 | 9 | 7 | 6 |
| | 8 | 8 | 6 | 7 | 4 | 8 | 7 | 9 | 10 | 8 | 4 |
| | 9 | 9 | 8 | 10 | 10 | 10 | 5 | 8 | 8 | 5 | 5 |
| | 10 | 10 | 9 | 5 | 6 | 7 | 8 | 5 | 4 | 9 | 8 |

Figure B.3: Precedence Constraints (Machine requirements for each task) for the 10x10 problem

| Task Number | Job Number | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 1 | 29 | 43 | 91 | 81 | 14 | 84 | 46 | 31 | 76 | 85 |
| 2 | 78 | 90 | 85 | 95 | 6 | 2 | 37 | 86 | 69 | 13 |
| 3 | 9 | 75 | 39 | 71 | 22 | 52 | 61 | 46 | 76 | 61 |
| 4 | 36 | 11 | 74 | 99 | 61 | 95 | 13 | 74 | 51 | 7 |
| 5 | 49 | 69 | 90 | 9 | 26 | 48 | 32 | 32 | 85 | 64 |
| 6 | 11 | 28 | 10 | 52 | 69 | 72 | 21 | 88 | 11 | 76 |
| 7 | 62 | 46 | 12 | 85 | 21 | 47 | 32 | 19 | 40 | 47 |
| 8 | 56 | 46 | 89 | 98 | 49 | 65 | 89 | 48 | 89 | 52 |
| 9 | 44 | 72 | 45 | 22 | 72 | 6 | 30 | 36 | 26 | 90 |
| 10 | 21 | 30 | 33 | 43 | 53 | 25 | 55 | 79 | 74 | 45 |

Figure B.4: Process Times for each Task for the 10x10 Problem