University of London
Imperial College of Science and Technology
Department of Management Science

JOB-SHOP SCHEDULING WITH APPROXIMATE METHODS

by

Alexander S. Spachis
Dipl. Eng., MSc, DIC

A thesis submitted for the
Degree of Doctor of Philosophy
in Management Science

November 1978

ABSTRACT

The general job-shop scheduling problem can be solved optimally only with prohibitively expensive enumerative methods, as has been demonstrated by the recent advancements in the theory of computational complexity. Thus sub-optimal or approximate procedures are the only realistic alternative.

The objective of this thesis is to investigate the suitability and performance of approximate methods (ad hoc algorithms, exact methods for relaxed problems and incomplete search procedures) applied to different facets of the basic problem. Apart from the development of efficient single-pass and enumerative flow-shop and job-shop heuristics, a new methodology is suggested with which performance guarantees are established and the expected behaviour of heuristics is assessed probabilistically, under different job-shop environments. A model of the dynamic behaviour of heuristics based on local neighbourhood search is developed for stopping decisions. Statistical methods are used to obtain estimates of optimal solution values for improved bound calculations and stopping rules in incomplete search procedures.

The behaviour of a new global and a number of local dispatching rules (single-pass, non-delay heuristics) is investigated in a dynamic and stochastic job-shop environment, with simulation. The effects of various data structures (differing in loading and in the statistical distribution, variance and accuracy of processing times estimates) on average waiting and on other measures of performance are studied. Approximate formulae are suggested for the calculation of the expected performance of heuristics in job-shops with differing variance of processing times.

Finally, the context within which these approximate methods can be applied and the aspects of implementation that require further research are discussed.

ACKNOWLEDGEMENTS

My sincere thanks are due to my supervisor Mr. J. King, Reader in the Department of Management Science, for his guidance encouragement and continuous interest throughout this study.

I would like to thank also Dr. N. Christofides for providing sources of information on the complexity of computer calculations, Dr. G.P. Cosmetatos for his advice on the approximate queueing formulae, Dr. R. Mattya for his comments on parts of this thesis, Dr. N. Meade for his advice on statistical tests and Mr. J. Beasley, for helpful discussions on problems related to this research.

Thanks are due also to the library, technical and other staff of the department for their help.

The financial support of the Greek Ministry of Coordination (Fellowship award) is acknowledged.

The understanding and help of my wife Helene and all my family have been a great support for my work. To them I dedicate this thesis.

CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1

INTRODUCTION

1.1 Definitions

1.2 Notation

1.3 Classification

1.4 Criteria of performance

1.5 Assumptions

1.6 Thesis outline

1.1 Definitions

'Scheduling is the allocation of resources over time to perform a collection of tasks' (Baker, 1974). This is one of the many definitions available. All of them have two elements in common, though they use different words: resources and tasks. Resources (or facilities) may be machines in an engineering industry, computers, doctors-nurses in a hospital, and generally processors. Tasks or more commonly jobs, require one or more operations on any combination of the facilities.

Scheduling includes also the sequencing function which given a set of 'tasks' and 'facilities' defines a sequence-succession of the operations on each facility. In this sense a sequence is not automatically a schedule. The sequence does not indicate start and completion times of the tasks. It is possible though to derive a complete schedule from the sequence.

The single most important field of application of scheduling is in production planning, within the area of operations management. The problem originates from the production function in industry, and this is reflected in the prevailing terminology.

Nowadays a number of problems outside this traditional area can be formulated and solved as scheduling problems. The most common are: the examination of patients by doctors in hospitals, manpower scheduling (e.g. nurses in hospitals) to provide a minimum service level per shift, the sequencing of programs in computers, the order of visiting cities by a salesman, preparing school timetables for teachers and classes. Besides these, there are a lot of common problems that are by nature scheduling and sequencing ones, and therefore can be dealt with using the tools of scheduling theory.

1.2 Notation

The notation and terminology adopted are the ones suggested by Conway, Maxwell and Miller (1967), Baker (1974) and adopted by the major periodicals in the field.

C_i completion time of job i

C_{max} makespan (time elapsed between start of schedule and finish of last operation).

F_i flow time (C_i less time of entry)

W_i waiting time

P_i aggregate processing time of job $P_i = \sum_{j=1}^{m} P_{ij}$

P_{ij} processing time of job i in facility j

D_i Due date

R_i time of entry

L_i lateness (or missed due date)

E_i earliness

T_i tardiness

I_j idle time of machine j over makespan C_{max}

Then, by definition:

$$W_i = F_i - P_i$$

$$C_i = R_i + F_i$$

$$C_i = R_i + P_i + W_i$$

$$L_i = C_i - D_i \qquad \text{(may be positive, zero or negative)}$$

$$T_i = \max (0, L_i)$$

$$E_i = \max (0, -L_i)$$

$$I_j = C_{max} - \sum_{i=1}^{n} P_{ij}$$

The following notation is also used:

n	number of jobs
m	number of machines
G	general job-shop sequence
F	general flow-shop sequence
P	permutation flow-shop
D	deterministic arrivals or servicing in stochastic and/or dynamic problems
M	negative exponential process (Markovian)
E_k	Erlang process
G_i	general independent
UR	uniform rectangular distribution
A	set of active schedules
μ	mean value
σ	standard deviation
V	coefficient of variation ($V = \sigma/\mu$)
α	linear regression constant
β	linear regression coefficient
r	correlation coefficient
b_e	lower bound
b_e^*	minimum of lower bounds
b_u	upper bound
a	location parameter
b	scale parameter
c	shape parameter

1.3 Classification

There are several possibilities for distinguishing scheduling problems.

(i) Randomness

The problem is deterministic if all the data involved are deterministic (processing times, sequences, technological constraints, availability of facilities).
The problem is stochastic if any of the data is stochastic.

(ii) Change of characteristics over time. The problem is static if none of the initial data changes over time, e.g. if all the jobs that are to be considered are available simultaneously at the beginning of the scheduling period.
The problem is dynamic if the data is subject to change with time, e.g. when the jobs arrive intermittently during the scheduling period.

This broad classification can be represented in the following table:

	Deterministic	Stochastic
Static	I	III
Dynamic	II	IV

The simplest form is the static and deterministic (I).
At the other end the dynamic and stochastic (IV) ones are the most complex.

Further classification is possible based on the resources available. There may be only one unit of each resource or many in parallel (scheduling of parallel processors) and they may be in one single stage or multistage (general job-shop scheduling).

The sequence (technological and precedence constraints) is another important feature for classification. The scheduling problems are divided into:

G general job-shop

F flow-shop (every job has the same path)

P permutation (the same job-sequence in all machines)

The classification adopted can be represented with four parameters as follows (Conway, Maxwell and Miller, 1967):

$$p_1 / p_2 / p_3 / p_4$$

For static and deterministic problems:

p_1 is the number of jobs n

p_2 is the number of machines m

p_3 describes the sequence pattern of operations (G,F or P)

p_4 describes the criterion of performance on which the schedule will be evaluated

For dynamic and stochastic problems a classification/notation based on queuing theory is adopted, $p_5 / p_6 / p_7$, where

p_5 is the arrival rate descriptor.
It can be D : deterministic
M : negative exponential/Poisson (Markovian)
E_k : Erlang with parameter K
G_i : general independent

p_6 describes the processing times distribution

p_7 is the number of facilities in parallel

Additionally it is useful for certain problems to specify another three parameters, $p_8 / p_9 / p_{10}$, where

p_8 describes the priority discipline in the queues

p_9 is the size of the population

p_{10} is the limit to the size of the queue.

1.4 Criteria of performance

The real scheduling problems in the production context are not restricted to finding a schedule or sequence allowing the tasks to be performed, but also fulfilling some goals/objectives. In practice these may be:

reduce queueing times

reduce stocks of finished goods or raw materials

reduce work in progress

increase production output

reduce back orders

reduce delivery periods and product lead times

reduce idle time of facilities

reduce idle time of manpower

reduce staff level and overtime

meet delivery targets

etc

Apart from these direct objectives, some indirect ones are of interest:

improve competitive position

reduce labour turnover

increase controlability (better information flow)

improve return on investment

etc

To measure the fulfilment of these objectives, some criteria or measures of performance are needed. The general term of 'regular measures of performance' is used for those criteria that can be described as a function of job completion times $Z(C_1, C_2, \ldots C_n)$ where the objective is to minimise Z and where Z can increase only if one of the completion times in the schedule increases. There are many

possible criteria related to the jobs or to the facilities, some of them common, some others of academic interest only. It has been possible to count over 20 different criteria or measures of performance, all of them related to time directly or indirectly. Some of them are different in name only, because it can be proved that they are equivalent to others. They can be obtained from each other by simple transformations involving constants, and schedules that are optimal for one of them are optimal for the whole group (Baker 1974, Coffman 1976, Rinnoykan 1976, Lenstra 1977).

(i) Minimising makespan or the maximum of completion times

C_{max} is equivalent to:

- minimising the sum of machine idle times

$$\sum_{j=1}^{m} I_j = \sum_{j=1}^{m} (C_{max} - \sum_{i=1}^{n} P_{ij}) = mC_{max} - \sum_{j=1}^{m} \sum_{i=1}^{n} P_{ij}$$

-minimising the weighted sum of machine idle times

$$\sum_{j=1}^{m} w_j I_j = \sum_{j=1}^{m} w_j (C_{max} - \sum_{i=1}^{n} P_{ij}) = C_{max} \sum_{j=1}^{m} w_j - \sum_{j=1}^{m} w_j (\sum_{i=1}^{n} P_{ij})$$

- maximising the average utilization of machines over C_{max}

$$m^{-1} \sum_{j=1}^{m} \{(\sum_{i=1}^{n} P_{ij})/C_{max}\} = \{\sum_{j=1}^{m} \sum_{i=1}^{n} P_{ij}\}/mC_{max}$$

- maximising the average number of jobs processed per unit time (expressed above)

(ii) Minimising the sum of completion times is equivalent to:

- minimising the sum of waiting times

$$\sum_{i=1}^{n} W_i = \sum_{i=1}^{n} (C_i - R_i - P_i) = \sum_{i=1}^{n} C_i - \sum_{i=1}^{n} R_i - \sum_{i=1}^{n} P_i$$

- minimising the sum of flow times

$$\sum_{i=1}^{n} F_i = \sum_{i=1}^{n} (C_i - R_i) = \sum_{i=1}^{n} C_i - \sum_{i=1}^{n} R_i$$

- minimising the sum of lateness

By dividing the above four measures by the number of jobs n, it is immediately seen that the following criteria are also equivalent to the $\sum_{i=1}^{n} C_i$:

- average completion time $\bar{C}$
- average waiting time $\bar{W}$
- average flow time $\bar{F}$
- average lateness $\bar{L}$

(iii) Minimising the weighted sum of completion times $\sum_{i=1}^{n} w_i C_i$ is equivalent to:

- minimising the weighted sum of waiting times
- minimising the weighted sum of flow times
- minimising the weighted sum of lateness (but not tardiness)

(iv) Minimising the sum of tardiness is equivalent to minimising the average tardiness $\bar{T} = \sum_{i=1}^{n} T_i/n$

There are a few more criteria that have been encountered in the literature but their value is very limited as they are not likely to be used in many practical situations (e.g. minimising the mean number of jobs in the system, calculated over C_{max}, expressing expected inventory or storage requirements; minimising the weighted sum of completion times with secondary criteria, in Burns, 1976; minimising the time-in-system variance encountered in Merton and Muller 1972, Schrage 1975, Eilon and Chowdhury 1977).

(v) The criteria 'minimising the maximum lateness' and 'minimising the maximum tardiness' are not equivalent but are related in the sense that a schedule that minimises L_{max} minimises also T_{max} (the reverse is not true).

$L_{max} = \max (L_i)$, i=1, ..., n
$T_i = \max (0, L_i)$
$T_{max} = \max (T_i) = \max (\max (0, L_i)) = \max (0, \max (L_i))$
Consider two schedules with $L_{max} \leq L'_{max}$
then $\max (0, L_{max}) \leq \max (0, L'_{max})$ or $T_{max} \leq T'_{max}$

The equivalence of criteria as discussed above reduces dramatically the number of distinct problems to the following:

$$C_{max}\ ,\ L_{max}\ ,\ \sum_{i=1}^{n} C_i\ ,\ \sum_{i=1}^{n} T_i\ ,\ \sum_{i=1}^{n} w_i C_i\ ,\ \sum_{i=1}^{n} w_i T_i$$

The single most important criterion of performance is the Makespan or Schedule Length (C_{max}), equivalent to the total of the machines idle times or to the average utilisation of machines. Its importance is reflected in the frequency with which it is encountered in the literature.

The next most important criterion is the average flow time $\bar{F} = (\sum_{i=1}^{n} F_i)/n$, equivalent to $\bar{C}$, $\bar{W}$ and $\bar{L}$.

A more general class of criteria is related to the idea of 'cost' or 'utility' (e.g. Eskew and Parker, 1975). A value of cost is assigned to each schedule taking into account not only times but also relative values of materials, equipment, labour required etc, in which case it is usual to try and minimise total cost. The same basic idea can be incorporated in the concept of a generalised utility function. As an example, a total cost expression could be a function of lateness, slack, work in progress, utilisation of facilities (discussed in Chapter 7). Both these types of criteria create problems in the sense that they need some coefficients to be determined, which is practically difficult, unless some arbitrary values are used.

1.5 Assumptions

There is a common set of assumptions encountered in the literature on scheduling problems (Gere 1966, Mellor 1966, Conway et al 1967, Day and Hottenstein 1970, Baker 1974).

(i) Machines do not break down and do not need servicing.

(ii) Preemption is not allowed (operations that start being processed are completed without interruption).

(iii) Machines can process one operation only at a time (no overlapping possible).

(iv) Job operations may not overlap (each job can be processed by one machine only at a time).

(v) Set-up and transfer times are either negligible or incorporated in the processing times.

(vi) No machine interchange (flexibility) is possible.

(vii) Processing times are fixed (the estimated time is equal to the actual). This assumption will be used throughout this thesis except Section 7.3 where it will be relaxed.

(viii) An important assumption that will be adopted is that the machines are used as single processors (one machine only in each machine centre). There is another category of problems where n jobs have to be processed on m machines in parallel (usually identical), with or without precedence constraints, due dates etc, with the objective of minimising makespan or the number of processors required.

(ix) Another assumption that is adopted is that precedence constraints exist for the operations of every job, but there are no precedence constraints for operations from different jobs. This assumption in fact excludes assembly and splitting operations of the type encountered in project planning (with CPM and PERT).

1.6 Thesis outline

The aim of this study has been to investigate the suitability and performance of a wide range of approximate methods applied to various aspects of the job-shop scheduling problem.

The first chapter has been devoted to definitions, notation, classification, criteria of performance and common assumptions related to job-shop scheduling.

Following this introduction, in the second chapter, the complexity of scheduling problems is investigated and the exact algorithms available for solving them are reviewed as two distinct groups: good optimising algorithms with polynomially bounded number of steps for the easier problems (to be named as P-class problems) and enumerative algorithms of polynomially bounded depths for the harder ones (named NP-complete). The approximate methods available are classified and reviewed as exact solutions of relaxed problems, ad-hoc algorithms and incomplete search procedures. This is followed by a discussion of methods of assessment of approximate solutions.

The third chapter deals with ad-hoc algorithms for flow-shop scheduling problems, with no passing or no-waiting allowed (the simpler form of job-shop problems). New algorithms are proposed and compared with solutions from the best available ones or from optimal procedures.

Ad-hoc single-pass algorithms for active schedules are also the subject of the fourth chapter, this time for the general job-shop problem, where a worst-case and probabilistic analysis of their performance is carried out.

The fifth chapter is concerned with incomplete search procedures as approximate methods for the general job-shop problem. A local neighbourhood search (LNS) method is described for the probabilistic and worst-case analysis of decision rules and a model is constructed describing the performance of heuristics as a function of time.

The applicability of statistical methods in LNS is discussed in the sixth chapter and in particular the use of the limiting form of the distribution of the smallest members of samples in stopping rules and bound calculations.

The dynamic and stochastic scheduling problem is treated separately with dispatching rules and simulation in the seventh chapter. The sensitivity of simulation results of a number of priority rules is investigated for different data structures (distribution, variance, loading and accuracy of time estimates). Approximate formulae are developed relating results from problems with different number of machines in parallel. Approximate formulae are also suggested, dealing with the job-shop as a complex network of queues and predicting the simulation results for different values of variance of the processing times.

The final (eighth) chapter sets the industrial context of these approximate methods in job-shop scheduling, reviews the thesis, highlights the parts that are believed to be its original contribution and puts forward suggestions for further theoretical and applied research.

CHAPTER 2

COMPUTATIONAL COMPLEXITY

2.1 Theory of computational complexity

The scheduling problem in its simpler forms of static and deterministic cases is extremely simple to describe and formulate. The problem $n/m/G/C_{max}$ has an infinite number of feasible solutions, being created by the insertion of arbitrary idle times between operations. If all possible delays are eliminated by shifting the start-times of operations as early as possible, without changing their sequence on any machine, a set SA of schedules - called semi-active schedules - is defined. The size of this set SA is limited by the number of possible sequences $(n!)^m$. The $n!$ represents the limit to the number of alternative sequences on one machine and the exponent m allows all their combinations to be included. The actual number of SA schedules is usually smaller than $(n!)^m$ because of the sequence or technological constraints. The significance of the size of this set is that it demonstrates the size of the problem and discourages full enumeration of solutions.

The set SA contains the optimal solution for any of the criteria described in Section 1.3 The reason is that for any schedule s not belonging in SA there is a semi-active one s' belonging to SA, derived from s with shifting some or all of the start times, maintaining the sequence and not increasing the value of any of the completion times. All these criteria are related to the completion times, since an improvement in their value requires a reduction of some completion time.

For the C_{max} group of measures, the optimal solution is guaranteed to be contained within A, the set of 'active schedules' $A \subset SA$. The set A results from SA by allowing 'global' shifting of start times, i.e. by allowing change of sequence, provided no job completion time increases as a result of that change. From the definition of A, it is clear that it contains an optimal solution, for any regular measure of performance.

If in the process of constructing the schedules no machine is kept idle, when it could begin processing of some operation, a set ND of 'non-delay' schedules is defined, (sub-set of A), which may not contain an optimal solution.

It is clear from the above discussion that the simpler form of static and deterministic job-shop problems is quite complex, since the smallest set that is guaranteed to contain the optimal solution is already too large. The number of alternative solutions is too large and the question arising is whether there is an efficient method that could give the optimal solution.

At this stage it is appropriate to discuss the nature of the scheduling problem in the light of recent advances in the theory of computational complexity. The scheduling problem is a combinatorial one with discrete feasible solutions, where optimisation through differentiation is not possible. There is a number of well-known combinatorial problems, similar in nature to scheduling: the travelling salesman problem, packing, assignment, transportation, shortest path of a network, shortest spanning tree, set covering, set colouring and many others. For all these non-differentiable discrete optimisation problems the fundamental question at issue is the number of operations required to find an optimal solution. Cook (1971) and Karp (1972) initiated the work on the computational complexity as part of the computer science field. A summary of this work is given below.

An algorithm is a precisely stated procedure or set of instructions that can be applied in the same way to all instances of a problem. A problem P can be fully described in a computer with a string of 0 and 1 elements, of length L, representing both the coefficients (numbers) and the operations performed with them. For large L ($L \to \infty$) the following definitions apply for the algorithms solving the problem.

Polynomial algorithm is one where the number of operations is proportional to L^k e.g. L^5, $7L^5 + 3L$ etc. (called also 'good' algorithm by Edmonds, 1965).

Non-deterministic Polynomial algorithm is one where the number of operations is proportional to k^L e.g. 2^L, $L^{\log L}$ etc. (polynomial depth algorithm).

The problems are divided accordingly to P-class and NP-class. Typical P-class problems, solved in a polynomially bounded number of steps are:

- assignment/transportation problems $0\ (n^{5/2})$
- shortest spanning tree of a graph $0\ (n^2)$
- network flow, 1 or 2 commodities $0\ (n^3)$
- shortest path, 1 or 2 commodities $0\ (n^2)$, $0(n^3)$
- P-median on tree-graphs $0\ (n^2p^2)$
- ordering n numbers in ascending order 0 (nlogn)
- minimising makespan of two-machine flow-shops and job-shops 0 (nlogn)

Examples of problems in NP-class, solved by a tree-search of polynomial depth are:

- 0-1 integer programming
- graph colouring/timetable/3-d assignment problems
- travelling salesman
- loading
- knapsack
- P-center, P-median
- set covering/partitioning/packing
- plant layout/quadratic assignment

- language recognition (find a given string within a given text)
- simplex method of LP (Klee, 1972). It is not known whether the LP problem is in P or NP class.
- general job-shop and flow-shop problem (Gonzalez and Sahni, 1978)

Besides P and NP-class problems, there are others that are thought to be more complex.

2.1.1 Reducibility and NP-complete problems

There is a subset of the NP-class problems, called NP-complete,defined by the property of reducibility. Each problem of this group can be transformed by a polynomial algorithm, simple or complex, to a problem named 'SATISFIABILITY' which itself is reducible to any other problem of the NP-complete group. (Cook 1971, Karp 1972).

The importance of the reducibility is that if a P-class algorithm could be found for one of the NP-complete problems, then P-class algorithms would exist for all of them. This is thought highly unlikely to happen, since the known NP-complete problems are the most difficult combinatorial ones:

- the general job-shop problem and most of its special cases (Ullman, 1976)
- knapsack problem
- bin-packing
- travelling salesman problem, Eucledean (undirected Hamiltonian circuit)
- directed Hamiltonian circuit (asymetric travelling salesman problem)
- partitioning a set of integers into two subsets with equal sums

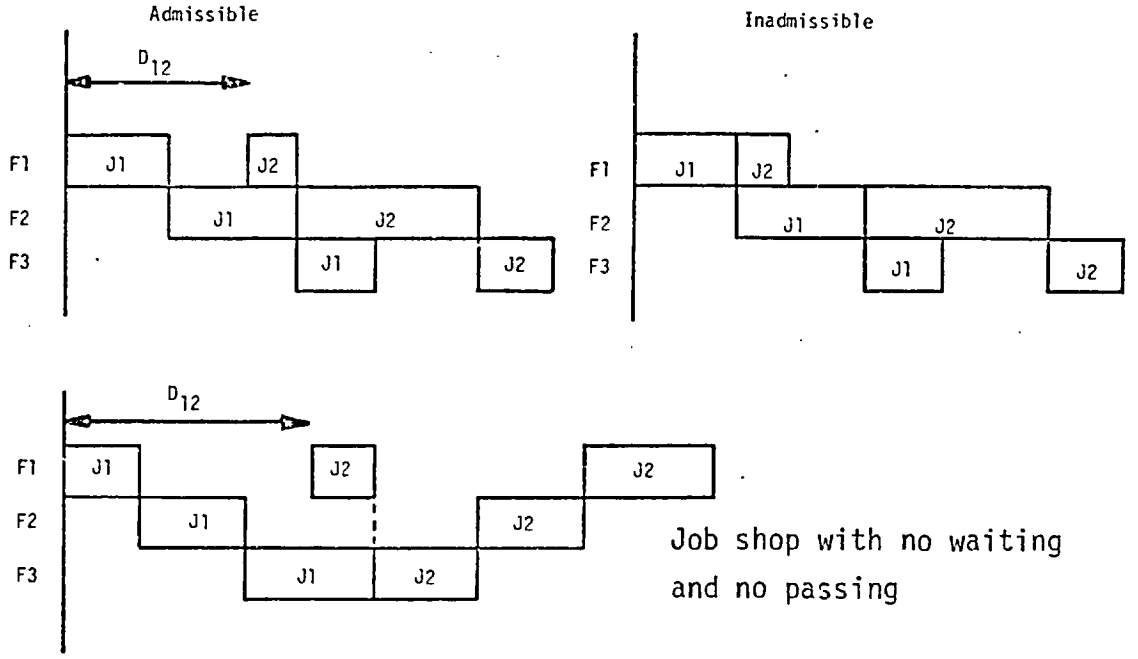
A detailed list of 25 NP-complete problems with descriptions is given in Karp (1975). It is worth adding here that for these problems the general approximation problem (i.e. find a solution with arbitrary distance ε from the optimal) has been proved to be also NP-complete (Sahni and Gonzalez, 1976).

2.1.2 Reducibility of scheduling problems

The reducibility of the general scheduling problem to the 'satisfiability' is of major theoretical importance, proving that the former is NP-complete,but in terms of efficiency of solution, it does not reduce the complexity of the problem and does not constitute any practical improvement. There are though some special cases of scheduling problems for which the direct reduction to some other well-studied and analysed combinatorial problem consitutes an efficient transformation. This is the case with the 'flow-shop, no-waiting' and 'job-shop, no-waiting, no-passing' problems which can be transformed to a directed Hamiltonian Circuit problem known as Asymmetric Travelling Salesman problem or ATSP (Wismer, 1972).

In a Gantt chart representation, a job profile in a flow-shop environment is represented as a staircase. The assumption of no-waiting (or no-waiting and no-job-passing in a job-shop) in effect means that this staircase must remain unbroken. Thus, left-shifting and passing of a complete job profile is allowed, though not of individual operations. This can be illustrated as in Figure 2.1 below.

Figure 2.1 Flow shop with no-waiting



For a given job sequence, the makespan is:

$$C_{max} = P_n + \sum_{i=1}^{n-1} D_{i,i+1} \qquad n\text{:last job in sequence}$$

where $P_n = \sum_{j=1}^{m} P_{nj}$ and $D_{i,i+1}$ is the delay incurred to job i+1 in the sequence, measured from the start time of job i.

The reduction is obtained by representing every job with a vertex (node or city) in the ATSP, where D_{ij} is the distance between the vertices i-j. An extra vertex represents a dummy first and last job, and every other vertex has zero distance from it. The distance of every vertex i to the dummy node is $P_i = \sum_{j=1}^{m} P_{ij}$. The ATSP distance matrix for a 4-jobs,m-machines problem would then be as in Table 2.1 on the following page.

An alternative reduction method based on total slack, instead of delays is described and used in Chapter 3.

The well known Bin-Packing and Knapsack combinatorial problems are also related to the scheduling one. The Bin-Packing problem can be formulated as a scheduling problem of m machines in parallel,

Table 2.1 Asymetric travelling salesman problem matrix

		To node				
		1	2	3	4	5
From node	1	-	D_{12}	D_{13}	D_{14}	P_1
	2	D_{21}	-	D_{23}	D_{24}	P_2
	3	D_{31}	D_{32}	-	D_{34}	P_3
	4	D_{41}	D_{42}	D_{43}	-	P_4
	5	0	0	0	0	-

n-jobs (tasks) with $m \leq n$ and a deadline for the completion of all tasks, where it is desired to minimise the number of machines required for processing the job-set. The knapsack problem (Lenstra et al, 1977) is also related to a number of special cases of the scheduling problem (a review of the state of the art on Knapsack problems can be found in Salkin and de Kluyver, 1975)

No 'good' algorithm has ever been found for any of the problems belonging to the NP-complete group, and it is very probable (though it has not been proved mathematically) that one does not exist. If such an algorithm were possible then all NP-complete problems would become P-class. The implication is that the general job-shop problem (static, deterministic) is solvable only by polynomial depth algorithms, i.e. some form of enumeration (tree-search).

The intensive research in the scheduling area, therefore, is unlikely to lead to an 'algorithmic' break-through. The discoveries about NP-complete problems require a change in the direction of research. It is more likely that eventually the breakthrough will take the form of the development of extremely powerful (large, fast and accurate) digital computers, which, together with the development of quantifiably approximate methods will allow the solution of practical

problems with results guaranteed to be close to optimal.

A systematic review of exact and approximate algorithms for P-class and NP-complete scheduling problems is presented in the following sections. Reviews and references on the scheduling problem and the algorithms available can be found also in Eilon and King (1967), Conway et al (1967), Elmaghraby (1968), Elmaghraby ed. (1973), Chowdhury M.A. (1974), Baker (1974), Karp (1975), Chowdhury I.G. (1976), Coffman ed. (1976), Garey et al (1976), Lenstra et al (1977) and Graham (1978).

2.2 Review of P-class scheduling problems

(i) Single machine scheduling with a finite number of jobs

- Minimise the sum of completion times, with due-dates

  Smith (1956), O(nlogn)

  Algorithm:
  Job i may be assigned the last position in the sequence only if

  $$D_i \geq \sum_{j=1}^{n} P_j \quad \text{and}$$

  $$P_i \geq P_k \quad \text{for all jobs k such that } D_k \geq \sum_{j=1}^{n} P_j$$

  (If another job were to take last position in sequence, a reduction of $\sum_{i=1}^{n} C_i$ could be achieved by transferring job i to the end of the sequence).

- Minimise the maximum lateness and tardiness

  Jackson (1955), O(nlogn)

  Algorithm :
  sequence the jobs in an order of non-decreasing due dates.

- Minimise the weighted sum of tardiness, with unit processing times and different arrival times

  Lawler (1964), $O(n^3)$

- Minimise the number of late jobs

  Moore (1968), O(nlogn), for $R_i \geq o$ Kise et al (1978), $O(n^2)$

- Minimise the weighted sum of completion times

  Horn (1972), Sidney (1975), O(nlogn)

- Minimise the maximum lateness, with simple precedence constraints.

  Lawler (1973), $O(n^2)$

- Minimise the maximum of the value of a special penalty function, subject to a number of restrictive assumptions.

  Sidney (1977), O(nlogn)

(ii) Two-machine problems

- Minimise Makespan (C_{max}) in flow-shops

  Johnson (1954), O(nlogn)

- Minimise Makespan (C_{max}) in flow-shops, without waiting

  Gilmore and Gomory (1964), Reddi and Ramamoorthy (1972), O(nlogn)

- Minimise Makespan in job-shops with one operation in each machine only

  Jackson (1956), O(nlogn)

- Minimise the sum of completion times, with two machines in parallel, unit processing times and simple precendence constraints

  Coffman and Graham (1972), $O(n^2)$

(iii) m-machine problems

- Minimise C_{max} in job-shops with two jobs.

  Hardgrave and Nemhauser (1963), Szwarc (1960), $O(m^2)$

- Minimise C_{max} with m identical machines in parallel, unit processing times, tree-like precedence constraints.

  Hu (1961), O(n) (with preemption, Gonzalez and Sahni, 1978)

- Minimise $\sum_{i=1}^{n} w_i C_i$ with m identical machines in parallel, unit processing times, time of entry $R_i \geq o$

  Lawler (1964), $O(n^3)$

- Minimise $\sum_{i=1}^{n} C_i$ (or $\bar{F}$) with m identical machines in parallel.

  Conway et al (1967), Baker (1974), O(nlogn)

  Algorithm :

  Step 1. Construct an SPT ordering/list of all the jobs.

  Step 2. To the machine with the least amount of processing already allocated, assign the next job from the list. Repeat Steps 1 and 2 until all jobs are assigned.

2.3 Exact methods of solution of NP-complete scheduling problems

The general scheduling problem, one of the group of NP-complete combinatorial problems (Coffman ed. 1976) and most of its special cases even with 1, 2 or 3 machines (proved to be NP-complete as well by Garey et al, 1976) can be solved optimally only with some algorithm of polynomially bounded depth (which in practice limits the size of problems that can be solved). This, inevitably, takes the form of an enumerative method either explicit or implicit and usually that of a tree search, where the set of solutions searched can be trimmed to a more manageable size by the use of some branch and bound method.

Although there is only one basic method, the enumerative one, there are numerous formulations encountered in the literature, summarised below, and subsequently described in some detail.

(i) Enumeration of all feasible solutions and combinatorial approach

(ii) Dynamic Programming

(iii) Integer Programming

(iv) Branch and bound algorithms (including precedence or disjunctive graph formulations)

One should bear in mind that there are limits to the size of even the simpler problems (static and deterministic) that can be solved optimally with these exact methods (as already discussed in Section 2.1), while near optimal or approximate solutions can be constructed more easily and efficiently with approximate methods or heuristics, to be discussed later (Section 2.4).

2.3.1 Enumeration of all feasible solutions and combinatorial approach

The obvious method of solution and the most inefficient, is a complete enumeration of all feasible solutions (members of the set A of active

schedules) that do not include arbitrary (unnecessary) delays and where no global left-shifting of operations can be made i.e. no operation can begin earlier without delaying any other operation . The set of active schedules is the smallest set guaranteed to include an optimal solution (dominant set) for any regular measure of performance while the set of non-delay schedules, where no machine is left idle while jobs are awaiting processing, may not contain an optimal solution . There are methods that allow complete enumeration of the set A with minimum cost, based on the principle of minimum change: given a sequence S the next one to be generated is determined by the principle of minimum cost changes.

An indication of the size of the problem is given by $O\{(n!)^m\}$ which, for a sample of problem sizes takes the following values:

Number of jobs	Number of machines	$O\{(n!)^m\}$
4	3	.138 E5
6	3	.373 E9
8	4	.264 E19
10	4	.173 E27
20	5	.852 E92
35	5	.118 E201

The set A of active schedules can be generated by the systematic partitioning procedure suggested by Giffler and Thomson (1960) (see also Brooks and White 1965, Baker 1974). This algorithm for generating all members of the set A of active schedules is summarised below.

PS_t : a partial schedule at time t

S_t : set of schedulable operations at time t

s_j : earliest start time of the schedulable operation, $j \epsilon S_t$

c_j : earliest completion time of the schedulable operation, $j \epsilon S_t$

Algorithm

Step 1. At time t=0, PS_t is empty, S_t contains all operations without predecessors.
Step 2. Determine $c^*=\min(c_j)$ for $j \in S_t$
Define m^* as the machine where c^* can occur.
Step 3. For each schedulable operation $j \in S_t$ requiring m^*, for which $s_j < c^*$ create a new partial schedule by adding j in the partial schedule PS_t, starting at time s_j.
Step 4. Update the set of schedulable operations (by removing operation j from S_t and adding the next operation of that job, creating the S_{t+1}).
Step 5. Update time to t+1.
Step 6. If all active schedules have been generated, go to 7. If not, return to 2.
Step 7. STOP.

The same set can be generated also with methods based on permutation changes. These changes can be based on minimum cost changing or lexicographic generation of allpossible permutations. Combinatorial approaches, relying on changing one permutation to another through switching around jobs, under certain conditions may produce optimal solutions. There are some special problems where optimisation techniques,based on a theorem by Smith (1956) for functions of permutations,can be applied (Elmagraby 1968, Rau 1970) but these methods cannot be used for obtaining optimal solutions of the general problem. They have been adopted though successfully for sub-optimal solutions (e.g. Neighbourhood Search) as will be discussed in Section 2.4 on heuristics.

2.3.2 Dynamic Programming

Dynamic Programming (DP) is an established technique for solving optimisation problems as sequential decision processes. It can be applied to problems with a separable objective function of the form 'minimise $F = \sum_{i=1}^{n} f_i(x_i)$' (where $x_1, x_2, \ldots . X_n \in D$ and D is the domain of values for all variables), with a single constraint and discrete variables that take integer values. This type of formulation seems to fit into a special group of scheduling problems and there has

been a number of attempts to use it. Held and Karp (1962) and Lawler (1964) have formulated the cost minimisation of a single machine problem, where costs are related to the completion times and the value of this objective function is calculated with recursive equations at every step of the algorithm. The solution method relies on the principle of optimality:

'an optimal sequence of decisions has the property that whatever the initial state and initial decision are, the remaining decisions must be an optimal sequence of decisions with regard to the state resulting from the first decision'.

This method is in fact a tree-search or branch and bound, breadth first, using dominance criteria instead of bounds. For this type of problem, the method can be illustrated as a graph with n stages of vertices and the minimum cost or minimum $\sum_{i=1}^{k} c_i$ as the shortest path from the initial to the final stage. The method always requires searching trees of about $n2^n$ nodes (Reingold et al 1977).

Another interesting dynamic programming formulation is due to Lawler and Moore (1969). It is a special case of minimisation of $F = \sum_{i=1}^{k} f_i(c_i)$ in a single machine problem, with precedence constraints (i.e. sequence of jobs is fixed) but each job may be processed in two different ways with differing costs. Dynamic programming has been used also for flow-shop problems of two-machines and sequence dependent set-up times (Corwin and Esogbue 1974). Special types of job-shop problems have been solved with DP algorithms by Sahni (1976) and Baker and Schrage (1978).

The main disadvantage of the dynamic programming formulation is that the dominance property used is not strong enough. At any intermediate stage of the solution a feasible alternative is discarded when its cost exceeds that of a complete feasible solution, which is not very efficient as a 'pruning' mechanism. Besides it is quite complicated, especially in keeping records of previous stages and intermediate decisions.

2.3.3 Integer Programming (IP)

IP Formulations

There is a vast literature on Integer Programming formulations and techniques for scheduling problems. The first IP formulation (0-1) was introduced by Bowman (1959) and is summarised below. At the point of time t, an operation k is either being processed, in which case a related variable x_{kt} takes the value 1, or else $x_{kt} = 0$.
The processing time of operation k then is

$$p_k = \sum_{t=1}^{b_u} x_{kt}$$

where b_u is a sufficiently large number or an upper bound. The number of operations is nm and the number of variables is nmb_u.
To ensure that one operation is processed at not more than one machine at any point in time t

$$\sum_{M_k} x_{kt} \leq 1 \text{ for } t = 1,\ldots.b_u$$

where M_k is the set of operations k on machine j, j = 1...m

To ensure that an operation, once started is completed without pre-emption

$$p_k(x_{kt} - x_{k,t+1}) + \sum_{i=t+2}^{b_u} x_{ki} \leq p_k \text{ for } t=1,\ldots b_u-1$$

Finally, the precedence requirements for operations are taken into account by the constraints

$$p_k x_{k+1,t} \leq \sum_{i=1}^{t-1} x_{ki} \text{ for } t=1,\ldots b_u$$

(in this case operation k precedes operation k+1)
The total number of constraints is $(2nm + m-n)b_u$
The objective function for minimising Makespan C_{max} is:

$$\sum_{t=1}^{b_u-b_o} \sum_{k\in\ell} (n+1)^t x_{k,b_o+t}$$

where $b_o = \max (P_i)$ for i=1,...n
and ℓ is the set of last operations.

At about the same time, Wagner (1959) suggested an IP formulation for the special case of permutation problems, with n^2 0-1 variables and

nm real variables. Manne (1960) has produced another formulation for the general problem. These formulations, reprinted in Muth and Thomson (1963), were not found to be very efficient.

A more efficient formulation of the problem as a mixed IP was proposed by Greenberg (1968).

s_{ik}	start time of job i in machine k (nm continuous variables)
$y_{ijk}=1$	if job i precedes j in k or else 0, thus when y_{ijk} is defined, y_{jik} is superfluous (mn(n-1)/2 0-1 variables)
M	large positive number
$r_{ifk}=1$	if operation f of job i requires machine k or else 0

Constraints:

$$\sum_{k=1}^{m} r_{ifk}(s_{ik} + p_{ik}) \leq \sum_{k=1}^{m} r_{i,f+1,k}\, t_{ik} \qquad (m-1)n$$

$$(M+p_{jk})y_{ijk} + (t_{ik} - t_{jk}) \geq p_{jk} \qquad mn(n-1)/2$$

$$(M+p_{ik})(1-y_{ijk}) + (t_{jk} - t_{ik}) \geq p_{ik} \qquad mn(n-1)/2$$

Objective function:

'Minimise Mean Flowtime' $\Sigma\Sigma(s_{i\ell} + p_{i\ell})$ where ℓ is the machine of the last operation of job i.

For 'Minimising Makespan', additional constraints are required, of the following form, where h is the last operation

$$\sum_{k=1}^{m} r_{ihk}(s_{ik} + p_{ik}) \leq C_{max} \qquad m,$$

and the objective function is 'minimise C_{max}'. This formulation for a problem of 10 jobs and 4 machines requires an IP of 220 variables and 390 constraints for 'min.$\bar{F}$' or 400 constraints for 'min.C_{max}'.

The work of Greenberg (1968) and of Pritsker et al (1969) provides more general models than the first formulations and demonstrates how a change in the definition of variables can drastically reduce the size of the problem (i.e. the number of variables and constraints). The resulting IP problems are still very large to be solved with general

IP methods. Only special structure cutting planes might be reasonably efficient, if a breakthrough were possible for the special case of job-shop scheduling.

Methods of solution of IP problems

There are two fundamental methods for solving scheduling problems formulated as IP; cutting planes and search methods.

In the 'cutting planes' method, which is based on the concept of 'relaxation' of integrality constraints, a corresponding LP is constructed whose feasible region is defined by the original constraints. The feasible LP region includes all the feasible integer solutions. The idea is to generate a number of linear inequalities that cut-out parts of the feasible region of the corresponding LP not containing integer feasible solutions, while leaving the feasible region of the IP intact. These constraints essentially represent necessary conditions for integrality. The continuous (LP) feasible solution space is modified until its continuous optimum extreme point satisfies the integer conditions.

The 'search' method basically enumerates the feasible integer points. It starts again from the continuous optimum and then partitions the solution space into sub-problems by deleting parts that do not contain feasible integer points.

Langrangean Relaxation and subgradient optimisation

The efficiency of this search can be improved by using bounds for eliminating a substantial part of the tree. A method for calculating lower bounds based on the use of Langrangean multipliers was implemented for the TSP by Held and Karp (1970) and (1971), called Langrangean Relaxation.

The basic idea with the Langrangean relaxation is that the objective function is changed to incorporate one or more of the constraints with Langrangean multipliers, which are then taken out and the relaxed problem is solved optimally.

For the problem minimise cx (x,A,B,C are vectors)

$$\text{subject to} \quad Ax \geq b$$
$$Bx \geq d$$
$$x \geq o$$

the Langrangean relaxation is

$$\min \{cx+\lambda(b-Ax)\}$$
$$\text{s.t.} \quad Bx \geq d$$
$$x \geq 0$$
$$\lambda \geq 0 \quad \text{(Langrangean multiplier/vector)}$$

It has been proved (Geoffrion, 1971 and 1974) that the optimal value of this problem is a lower bound to the optimal of the original problem, for any value of $\lambda \geq 0$

And the problem

$$\max \{\min \{cx+\lambda(b-Ax)\}\} \text{ for } \lambda \geq 0,\ Bx \geq d \text{ and } x \geq o$$

gives the best possible lower bound to the original problem. The difference between the optimal value of the original and of the last problem is called the duality gap. One technique of determining the values of λ that maximise the lower bound is subgradient optimisation. This method of solution offers itself for problems that are basically simple problems with additional constraints. By incorporating these constraints into the objective function, the problems are reduced to the simpler basic forms.

Fisher (1973) has used this method for a 0-1 formulation of the scheduling problem, where 'the basic branching mechanism used is to select a resource k and a time period t and allocate the available resource R_{kt} to the various resource-feasible subsets of the tasks

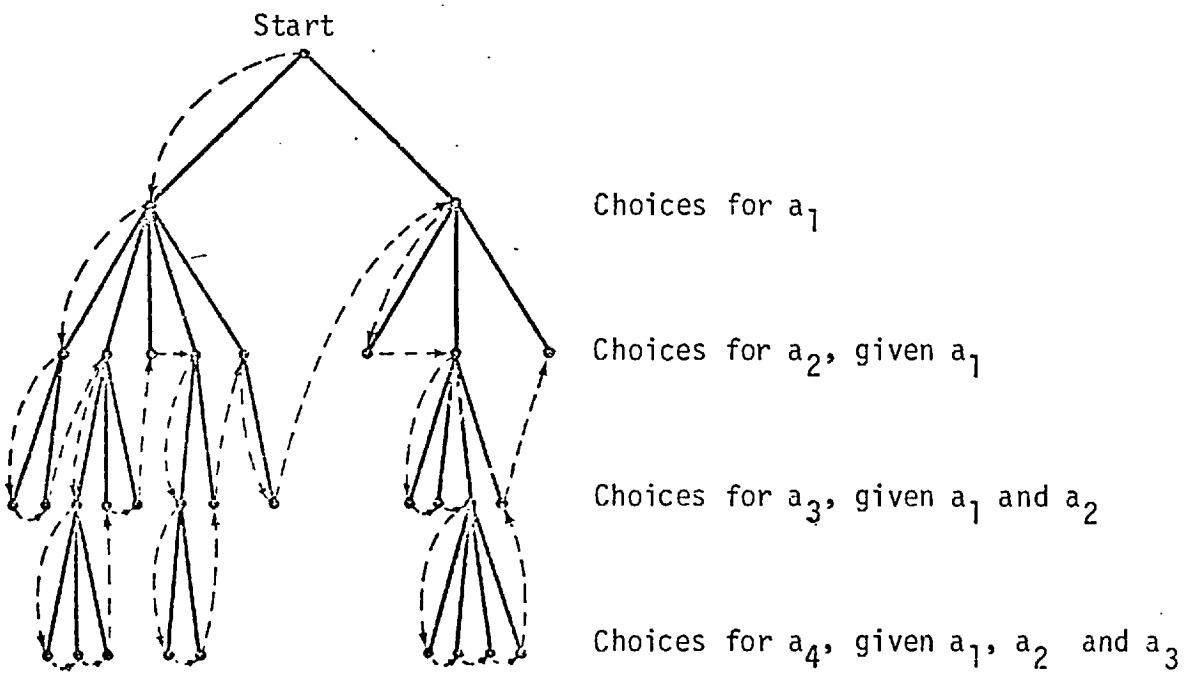
which may use resource k during time period t'. With his formulation Fisher succeeded in solving an independent scheduling problem for each job. It is worth noting that this method of Langrangean relaxation cannot be applied successfully with a partitioning method where partial sequences remain fixed.

The Langrangean Relaxation and Subgradient Optimisation method can be applied generally in problems of minimisation of non-convex functions. A good review of the method and its applications can be found in Shapiro (1977).

2.3.4 Tree-search and branch and bound methods

The underlying concept is the repeated decomposition of a problem into several partial problems of smaller size, until the undecomposed problem is either solved or proved not to yield an optimal solution. The method can be described in terms of a generalised 'backtrack search' in which any solution to the problem is a vector $V = (a_1, a_2, ...)$ with finite but undetermined elements. Each a_i is a member of a finite set A_i, with N_i elements. An exhaustive search must consider all the potential solutions i.e. the elements of $(A_1 x A_2 x ..... x A_j)$. The initial value v of the solution vector is $V = \emptyset$.
The constraints of the problem, e.g. sequence requirements for operations indicate which of the members of A_k are candidates for a_k, forming a sub-set S_k, $S_k \subset A_k$. In this way, a partial solution is built up from $(a_1, a_2, a_3, ... a_{k-1})$ to $(a_1, a_2, ... a_{k-1}, a_k)$.
If the partial solution $(a_1, a_2, ... a_{k-1})$ does not allow any possibilities for a_k, then $S_k = \emptyset$ and backtracking takes place.
This process can be represented graphically in a tree. The following diagram in Figure 2.2 is for a depth-first tree, where the nodes are visited (by a traversal) in the order indicated by the arrows (Reingold et al, 1977).

Figure 2.2 Depth-first tree search



Branch and bound is a specific type of backtrack search, where every partial or complete solution has a cost f associated with it, and the optimal solution (i.e. the one with least cost) is to be found. For any k, generally, $f(a_1,a_2,\ldots.a_{k-1}) \leq f(a_1,a_2,\ldots.a_{k-1},a_k)$ and in scheduling $f(a_1,a_2,\ldots.a_{k-1}) + C(a_k) = f(a_1,a_2,a_3,\ldots.a_{k-1},a_k)$ A partial solution may be discarded if its cost is greater or equal to the cost of a previously computed solution.

Details of implementation of such a method are given in Chapter 5. Here, it suffices to add that there are two different search strategies, the 'depth-first' and the 'breadth-first' and that potential improvements in the efficiency of the methods might be obtained by merging identical sub-trees, at the expense of keeping more complex records of the solution instances.

The published literature on branch and bound methods in scheduling is enormous. Reviews of the relevant literature can be found among others in Lawler and Wood (1966), Mitten (1970), King (1975), Reingold et al (1977) and Ibaraki (1977).

Applications of branch and bound in scheduling problems

The single machine problem of minimising the maximum lateness with different job arrival times is NP-complete. A number of attempts to solve it have been published over the last few years by Bratley et al (1971), Dessouky and Margenthaler (1972), Baker and Su (1974), McMahon and Florian (1975). Comparison of these methods on sample problems has shown that the last two and especially the last one are quite efficient.

The general cases of the single machine problems of 'minimising the sum of the weighted tardiness' (Schwimer, 1972 and Fisher, 1974), of 'minimising mean tardiness' (methods reviewed in Baker and Martin, 1974) and of 'minimising the sum of completion times' (Rinnoykan et al 1975) are also NP-complete and solvable with branch and bound. An interesting development for the problem of 'minimising total tardiness' is due to Lawler (1977) who has constructed a 'pseudopolynomial' algorithm for its solution.

The problem of minimising the sum of completion times in a two-machine flow-shop was formulated and solved by Kohler and Steiglitz (1975), using a lower bound developed by Ignall and Schrage (1965) and Lomnicki (1965). Uskup and Smith (1975) have used a branch and bound method for minimising the total cost, dependent on set-up times and Townsend (1977c)has shown how Lawler's procedure for minimising the maximum penalty in the single machine problem can be combined with Johnson's rule to produce a branch and bound algorithm for the two-machine version.

For the n-jobs m-machines flow-shop problem, Lomnicki (1965) and Ignall and Schrage (1965) independently, applied a machine-based branch and bound algorithm (for the three-machine flow-shop problem). McMahon and Burton (1967) improved the efficiency of the method by using the best of machine-based and job-based bounds. Ashour and Quraishi (1969) made a study comparing the various available methods and concluded that Lomnicki's method was probably the best, taking into account total computational costs.

The pioneering work in the n-jobs m-machines job-shop problem was conducted by Giffler and Thompson (1960) who proved that the optimal solution is contained within the set of active schedules, as described above in the 'complete enumeration' method. A basic branch and bound method for searching this set was constructed by Brooks and White (1965) and since then, many researchers have contributed towards improving its efficiency (e.g. Florian et al, 1971). More details about the bound calculation will be discussed in Chapter 5.

These basic ideas have been used in the formulation of the same problem described in terms of a precedence or disjunctive graph (Balas, 1969, Charlton and Death, 1970b, Schrage, 1970, Ashour and Parker, 1971, Florian et al, 1975). This formulation, as described and used by Ashour and Parker, is summarised below.

Formulation

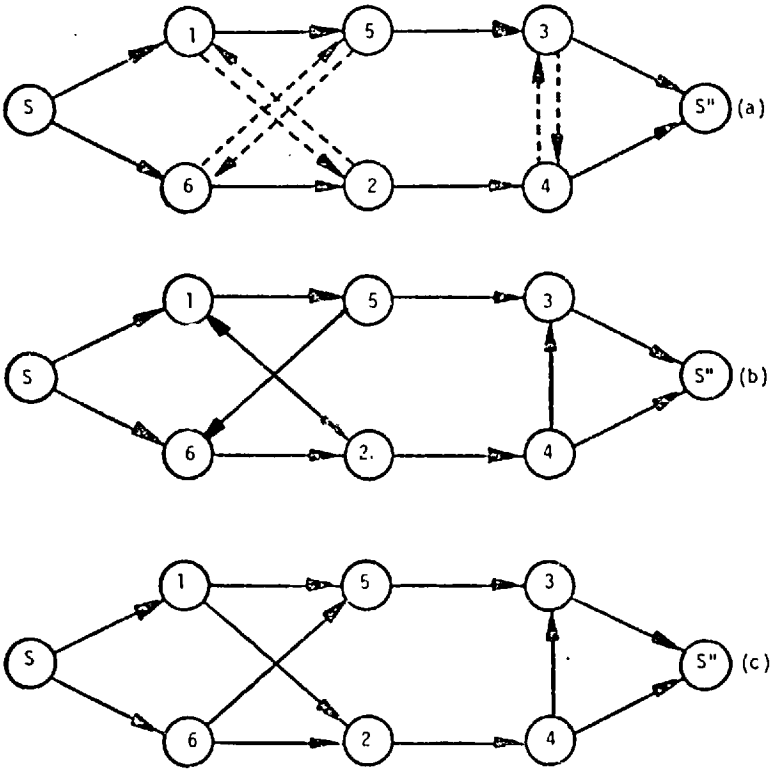
The finite directed graph G(X,A) with
$X = \{x_1, x_2, \ldots\ldots\ldots\ldots x_k\}$ vertices or nodes k=nm
$A = \{a_1, a_2, \ldots\ldots\ldots\ldots a_\ell\}$ arcs (links)
represents the scheduling problem as follows.
Every vertex x_i corresponds to a single operation.
Every directed arc a_j corresponds to a precedence requirement or decision.
The graph has also a dummy source vertex S and a sink S'.
The set X can be divided into n disjoint subsets J_i (jobs)
($J_i \cap J_j = \emptyset$ for $i \neq j$ and $X = J_1 \cup J_2 \cup \ldots\ldots \cup J_n$)
and into m disjoint subsets M_i (machines)
($X = M_1 \cup M_2 \cup \ldots\ldots\ldots \cup M_m$ and $M_i \cap M_j = \emptyset$ for $i \neq j$)
Every fixed or determinate arc corresponds to a fixed precedence required by the technological constraints of job operations processing.

The remaining arcs are indeterminate, indicating that a precedence/sequence is possible, but subject to decision. On the diagram, the latter are represented with broken lines while the former with continuous. As an example, a scheduling problem with two jobs, three machines, six operations is given in Figure 2.3 (a) below.

Figure 2.3 Disjunctive graph for a job-shop scheduling problem

Precedence of operations and Vertex identification

JOB	FIRST	SECOND	THIRD
1	$1,V_1$	$3,V_5$	$2,V_3$
2	$3,V_6$	$1,V_2$	$2,V_4$



Between vertices V_1 and V_2 there are two arcs, to indicate that any of the operations may precede the other. In a feasible schedule, every pair of these indeterminate arcs has to be replaced by one determinate only. For a schedule to be feasible, there must not exist directed 'cycles', i.e. closed paths. In this sense, the graph in Figure 2.3 (b) does not represent a feasible schedule, while the one in Figure 2.3 (c) does. The problem of minimising Makespan (C_{max}) then becomes one of finding the shortest critical path.

The solution of the problem requires examining explicitly or implicitly all the possible fixings of the indeterminate arcs, in a branching and bounding procedure.

For the problem of minimising the maximum lateness, a branch and bound solution has been proposed by Townsend (1977a). Townsend (1977b) has also constructed a branch and bound algorithm for the problem of minimising the maximum penalty, defined as a cost function of completion times.

For problems of n-jobs and m-machines in parallel, the problem of minimising the average flow time ($\bar{F}$), without precendence constraints can be solved in O(nlogn), while with general precedence constraints, it is NP-complete even in the single machine case. With two or more machines and tree-like precedence constraints, it is again NP-complete (Sethi, 1977).

For the problem of minimising the maximum lateness, for m-machines in parallel, with earliest start and due-date constraints, Bratley et al (1975) have proposed a branch and bound method.

2.4 Approximate methods of solution

2.4.1 The need for approximate methods

Scheduling problems of the real world are very complex and even their simplified versions are difficult to solve exactly, as has been demonstrated by the discussion about their computational complexity. Besides, in most of the real life problems, an exactly 'optimal' solution is not essential. In fact, a sub-optimal solution is usually acceptable, especially when some information is available about its distance from the optimal. The idea of using sub-optimal methods is not new. Heuristics have been in use for some time and their successes and failures have been subject to a lot of discussion (Conway et al 1967, Baker 1974).

It is accepted in this research project that the discoveries about non-deterministic-polynomial-time-complete problems require a change of direction of research in scheduling. Earlier, a lot of effort was directed at finding optimal or exact solutions. This is not considered to be a very fruitful direction any longer. Instead, attention in this work is focussed to simple methods and efficient algorithms that will allow generation of solutions close to optimal and an assessment of their performance.

2.4.2 Types of approximate methods

There are several types of approximate methods and heuristics, based on three different principles. These three broad categories of heuristics can be named as follows:

exact solutions of relaxed problems

ad-hoc decision rules and algorithms

incomplete search procedures, leading to a local but possibly not global optimum.

Exact solutions of relaxed problems

(i) This can be achieved by simplifying the structure of the original problem, rendering it a P-complete one if possible. An example of this idea is the heuristic of Campbell, Dudek and Smith (1970) for flow-shops without passing, where a surrogate problem in P-class is solved optimally with Johnson's method and the sequence is used as a heuristic solution. The same principle is used in relaxing constraints with the Langrangean relaxation method, in aggregating constraints to create a surrogate IP and in ignoring constraints or coefficients.

(ii) Similar effects are obtained by modifying the problem coefficients, as in the Right Hand Side (RHS) of an IP.

(iii) Another interesting concept has been used by Ashour (1970) to decompose the initial problem, solve optimally the sub-problems and then put together their solutions to yield a complete sequence.

Ad hoc decision rules and algorithms

These methods have been the first to be used as heuristics for sub-optimal decisions. They are based mainly on intuition or they are trying to recognise and imitate the subjective decision making patterns of people and substitute them by objective decision rules. In Chapter 3 a number of heuristics of this type are proposed for flow-shops and analysed. In this category, one should include heuristics such as those proposed by Palmer (1965) using a 'slope index' for each job based on the length of its operations and constructing permutation schedules according to the value of this index, and by Gupta (1972), using a similar approach, but with a completely different sequencing criterion which compares favourably with Palmer's.

The most well known heuristics in this category are the single pass priority rules (active and non-delay dispatching procedures). Once a decision is taken, it is implemented and there is no reconsideration for alternative courses of action. Their advantage is the extreme simplicity of application, which allows for decentralisation of the decision making process to local centres in a job-shop environment, and which makes them an attractive proposition especially for large-scale problems.

The priority rules are also referred to as non-backtrack tree search methods. Referring to the notions used in the description of the general branch and bound method, from the subset S_k of A_k, one member a_k is selected according to a priority rule. Each member a_k is assigned a priority value and the one with higher priority is selected. When a complete feasible solution is constructed, a heuristic solution is found and the process is terminated. A number of heuristics of this type are described and analysed in Chapter 4. Single-pass heuristics (as well as multi-pass, with backtracking) have been used for special cases of the general problem, with various criteria. Holloway and Nelson (1973, 1974, 1975) have studied heuristics for problems with due-dates, flexible routes, variable processing times, and criteria related to due-dates.

Priority dispatching (non-delay) rules are studied in a stochastic/dynamic job-shop environment with simulation and approximate formulae from queueing theory, in Chapter 7. In fact, for the general dynamic and/or stochastic problem, with its immense complexity, there is no other practical scheduling method except the single-pass priority or dispatching rules. The difference between this type of modelling, generally known as 'simulation' and the modelling of the simpler static and

deterministic problems lies in the treatment of the discrete events and time. While the arrivals in the static problem are taken to be simultaneous at time zero only, with simulation they are treated as stochastic and over a long period of time. The processing times are also stochastic in the sense that there is an estimated processing time and a realised one which may well be different. In order to obtain more reliable results with the stochastic data of this modelling, large numbers of jobs are used and no individual characteristics/results are collected but only statistical ones, usually in the form of histograms. Reviews of this approach are given in Gere (1966), Conway et al (1967), Day and Hottenstein (1970), Hollier (1968), Chowdhury (1976), Panwalkar and Iskander (1977).

The stochastic structure of this category of problems bears resemblance to the principles of queueing theory, that is, arrival distributions, service times, infinite source population (for large N) etc. This resemblance leads to the consideration of the job-shop scheduling problem as a network of queues. The use of some approximate results (formulae) from queueing theory is discussed in Chapter 7.

Incomplete search procedures and local optimisation

(i) It is possible to terminate a branch and bound tree search, before an optimal solution is found. This search, if taken to completion, would guarantee an exact (optimal) solution. The method of such an incomplete search depends on a number of parameters and decision rules. The behaviour of heuristics related to this method is studied in Chapter 5.

(ii) It is also possible to abandon parts of the search altogether,

i.e. eliminate branches when it is suspected that it is unlikely that they would lead to exact solution, and concentrate on more promising ones. This can be implemented by strengthening the bounds beyond their calculated value, e.g. by using fictitious bounds. This increases the efficiency of the search with the effect of obtaining possibly , sub-optimal solutions, (Ibaraki, 1976a,Bazaraa and El-Shafei, 1977).

Another rule that can be used for deciding when to abandon a part of a search is suggested in Chapter 6, based on probabilistic, statistical estimates of the exact solution value, by which the branch with the best estimate of an optimal solution is pursued further.

(iii) Another incomplete search procedure for local neighbourhoods is based on combinatorial analysis (not a branch and bound search). The general permutation problem of size n is to minimise an objective function $f(\pi,x)$ where π is a permutation from the solution space $P \triangleq \{\pi\}$ and x is the parameter space (set of data). For each permutation $\pi \in P$ a subset of P is defined, $N(\pi)$ called the neighbourhood $N(\pi)$ of π. An initial solution is constructed and the neighbourhood is searched with a search method S for improvements in the objective function. Examples of neighbourhoods are defined by backward or forward single insertion (e.g., abcde → acdbe) and by adjacent pair interchanges which can be searched in $O(n^2)$ number of iterations (steps). A local optimum is achieved with respect to the neighbourhood $N(\pi)$ when improvement of the objective function value is not possible. The basic difference of this type of local neighbourhood search from the branch and bound tree search is that in the former a complete solution is obtained at every iteration, whose construction is based on the previous complete solution while in the latter the solution is obtained from fixed partial solutions and eliminations.

This method has been applied successfully for sequencing in the TSP by Lin (1965), and described as the λ-optimal method in Eilon, Watson-Gandy and Christofides (1971). Baker (1974) reports the applications of such a method for the $n/1/\bar{T}$ problem. Kohler and Steiglitz (1975) have used such a method for $\bar{F}$ in flow-shops, reported also in Coffman ed. (1976).

2.5 Assessment of approximate methods

2.5.1 Heuristic performance guarantees

The main criticism for the approximate or heuristic methods has been that they are unpredictable and unreliable. This is related to the approach adopted, which has been to try a number of test problems with the heuristic, compare results with those from other heuristics, occasionally solve the same problems with an exact enumerative method, compare with the optimal solution and deduce conclusions of a qualitative nature, e.g. heuristic A is better than B most of the times, heuristic C gives values very close to the optimal etc.

Comparing the heuristic value with the optimal is not a practical proposition, nor is it very useful to know that 'heuristic values are close to the optimal'. This type of information gives some insight into the problem but one needs more information, preferably in quantitative terms. This has been the underlying idea for some recent research on the guaranteed performance of heuristics. It usually takes the form of a ratio $v(H)/v(E) \leq k$, where $v(H)$ is the value from the heuristic solution, $v(E)$ is the value from the exact solution and k is a real positive number specified for the heuristic.

A number of studies on this line have proposed heuristics with guaranteed worst case behaviour for a few problems, to be discussed below. Yet for many other problems, it appears that no efficient algorithm can guarantee any value for the above coefficient k. For the problem of minimising Makespan (C_{max}) of n jobs with precedence constraints on m identical processors in parallel, Graham (1966) proved that by scheduling the first available job from the list of schedulable jobs, as soon as a machine becomes idle (non-delay

scheduling),

$$\nu(H)/\nu(E) \leq (2-1/m)$$

When the jobs are independent, i.e. without any precedence constraints, then (Graham, 1969)

$$\nu(H)/\nu(E) \leq 4/3 - 1/3m$$

which gives a bound of 33% on the maximum error.

Another interesting result for the problem of two machines in parallel can be obtained by scheduling the 2K longest jobs (K: any integer) in an optimal procedure and the remaining short ones arbitrarily. Then, (Graham, 1978)

$$\nu(H)/\nu(E) \leq 1 + 1/(2K + 2)$$

For n jobs, the maximum number of iterations required is $2Kn + 2^{2K}$ (2Kn steps for selecting the 2K longest jobs and 2^{2K} for finding their optimal arrangement).

For the same problem, Ibarra and Kim (Graham, 1978) have developed recently an algorithm for which

$$\nu(H)/\nu(E) \leq 1 + 1/K \text{ in } O(n + K^4 \log n)$$

(for large K and n, $n + K^4 \log n << 2^{2K}$).

Another problem for which a heuristic was proved to have guaranteed performance was the bin-packing (a special form of the scheduling problem). Given m identical bins of size Q and n items of sizes $q_1, q_2, \ldots q_n$, put the items in the minimum possible number of bins. In scheduling terminology, the items are jobs, of length q_i, the bins are identical machines in parallel and Q is a fixed deadline. The same problem in another form is known as a cutting problem. In a first-fit procedure, Ullman in 1973 proved that for any precedence constraints imposed $\nu(H) \leq (17/10)\nu(E) + 2$

and if $\nu(E)$ is a multiple of 10, then

$\nu(H) \leq (17/10)\nu(E)$ (maximum error 70%)

Johnson, in a proof longer than 75 pages, showed that for any precedence constraints with decreasing values of q_i, the same first-fit method produces $\nu(H) \leq (11/9)\nu(E) + 4$. When the optimal value $\nu(E)$ is large, then this heuristic solution is quite satisfactory, producing no more than 22% error, in $O(n\log n)$, (Johnson et al, 1974).

A heuristic with guaranteed maximum error of 50% has been constructed for the Travelling Salesman Problem with symmetric distance matrix based on the solution of two relaxations of the TSP (shortest spanning tree and matching problem) with P-class algorithms (Christofides 1975 and 1976). The performance of this heuristic is bounded by

$$\nu(H)/\nu(E) \leq 3/2$$

For the TSP with general arbitrary matrix and for the graph colouring problem, it has been proved that no P-class heuristic can guarantee error less than 100% (Sahni, 1976).

These heuristics described above are interesting not only for the special problems to which they apply, but generally for the methodology used, which might give some insight on how to pursue similar work on other aspects of the scheduling problem with heuristic algorithms (Garey et al, 1978)

The main disadvantage of these heuristics is that although they guarantee a maximum error for the worst case, they do not quantify how good the solution actually is. The expected performance of a heuristic is more useful as information, especially if it can take into account the parameters of the problem. For this purpose, a probabilistic analysis is suggested below.

2.5.2 Probabilistic analysis of heuristics

The worst case behaviour of a heuristic algorithm, when known, is certainly useful information in that it guarantees some minimum quality of performance. These guarantees may be of the order of

100% error, and one is left without information about the expected performance of the heuristic.

Ideally one would like to know the probabilistic behaviour of a heuristic algorithm for all the possible instances and forms of a problem. For the general case of heuristic algorithms, it is not possible to derive this information with mathematical analysis. The only case where such an analysis has been possible,and only very recently, is for a partitioning algorithm (special structure heuristic) for large travelling salesman problems in the plane (Karp, 1977). No similar or related method can be applied for the general job-shop problem. What can be done (Chapters 3, 4, 5) is to look at a number of sizes and data structures of the problem and derive a probabilistic empirical profile for the performance of heuristics.

CHAPTER 3

APPROXIMATE FLOW-SHOP ALGORITHMS

3.1 Flow-shop problems and methods of solution

3.2 Flow-shop heuristics

3.3 New heuristics for flow-shops with no-job-passing

3.4 New heuristics for flow-shops with no-waiting

3.1 Flow-shop problems and methods of solution

The general flow-shop problem of sequencing n jobs on m machines, in order to minimise some objective function (e.g. C_{max},ΣF,$\bar{W}$), is NP-complete, as already discussed in Section 2.1. As a special case of the general job-shop problem it could be solved with the methods available for job-shop problems. But it has been felt that its special structure of maintaining the same sequence of operations through the machines could be used to simplify and improve the efficiency of the enumerative solution procedures. Thus, it has attracted considerable attention on its own. A number of branch and bound algorithms and other elimination methods based on dominance properties have been reported in the literature for the exact solution of the problem.

The branch and bound methods for minimising makespan by Lomnicki (1965), Ignall and Schrage (1965) and McMahon and Burton (1967) have already been discussed in Section 2.3, evaluated in Baker (1975a) and thus need not be presented here.

A review of the elimination methods can be found in Szwarc (1971), who has contributed to the development of dominance criteria. It is worth noting that for regular measures of performance, the sub-set of the $(n!)^{m-1}$ schedules, where the job-sequence is the same in the first two machines, is dominant. For makespan problems the sub-set of schedules, where the job sequence is the same in the last two machines is also dominant, which reduces the set to be searched to $(n!)^{m-2}$. This also means that for two-machine problems with any regular measure of performance, and for three-machine makespan problems, one needs to search only the dominant set of permutation schedules. Baker (1975b) has conducted an extensive study comparing the performance of branch-bound and of elimination methods in the $n/m/F/C_{max}$ problem, which clearly showed that the latter are inefficient.

Attention has also been paid to flow-shop scheduling problems with other criteria e.g. average completion time (Krone and Steiglitz, 1974). A review of exact and approximate methods for the problem of minimising the average flow-time $\bar{F}$ or the sum of flow-times $\sum_{i=1}^{n} F_i$, including computational results has been produced by Kohler and Steiglitz (1975). Computational results for the mean flow time problem are also given in Bansal (1977). Another seemingly different criterion, that of miminising the weighted sum of machine idle times $\sum_{j=1}^{m} w_j I_j$, for which an optimal method has been proposed by Gupta (1976), is in fact equivalent to minimising C_{max}:

$$\sum_{j=1}^{m} w_j I_j = \left(\sum_{j=1}^{m} w_j\right) C_{max} - \sum_{j=1}^{m} \left(w_j \sum_{i=1}^{n} P_{ij}\right)$$

The equivalence of these criteria renders the method of Gupta a trivial extension of the one developed by Wismer (1972), who has produced an exact method for the special case of flow-shops with no intermediate queues, based on the TSP. Gelders and Sambandam (1978) have suggested optimal and approximate procedures for another criterion, defined as a complex cost function.

Other methods for special structures of the problem have been presented that are more efficient than the general methodology. Such is the case in Gupta (1975), Smith, Panwalkar and Dudek (1975 and 1976) and Panwalkar and Khan (1976 and 1977). A dynamic programming method has been used for flow-shops of two-machines and sequence dependent set-up times (Corwin and Esogbue, 1974). Bestwick and Hastings (1976) have used a branch and bound method for flow-shops with no-job-passing, where all jobs are not processed by all machines. For the permutation flow-shop problem, an efficient branch and bound method has been reported by Lageweg et al (1978).

The landmark that has influenced substantially research on flow-shops was the pioneering work of Johnson (1954) who produced an algorithm for optimal solutions of the two-machine problem, summarised below.

Step 1 Find the minimum of available operations P_{i1}, P_{i2} from the set of schedulable jobs S, where P_{ij} is the processing time of job i in machine j.

Step 2 If the minimum processing time requires machine 1, place the job in the first available position in sequence. If it requires machine 2, place the job in the last available position in sequence.

Step 3 Remove the assigned job from the set S of schedulable jobs and return to Step 1, till S becomes empty.

Several extensions to special structures of three-machine problems have been possible. Johnson proved that his algorithm yields optimal solutions when the second of the three machines is dominated. Another special three-machine (A,B,C,) case is when the Johnson's rule applied to AB,BC, AC gives the same ordering. Then this sequence is the optimal for the three-machine case (Burns and Rooker, 1976). Szwarc (1974) gave a sufficient optimality condition for a solution obtained by Johnson's method when the three-machine problem is relaxed to a two-machine one with time lags (extended by Burns and Rooker, 1975). Szwarc (1977) has also defined a small sub-set of the feasible solutions which contains an optimal, whenever it is not empty. Recent additions to the theory of the three-machine problem are due to Gupta and Reddi (1978), Szwarc (1978), Burns and Rooker (1978).

Extensions of the basic two-machine model have been possible. Mitten (1958) showed that Johnson's method yields an optimal solution when start-lags and stop-lags are included. The basic method can be applied also when the two machines A and B are not available simultaneously. The proof of this extension has been necessary for the application of a new bounding method used in the study of enumerative heuristics in job-shop scheduling problems (described in Ch.5).

Proof: If machine B is available before A, the start time of the schedule is that of machine A, and the optimal sequence is the same as if A and B were available simultaneously. If machine B is available I_d time units after machine A (which anyway has no idle times) and $\sum_{i=1}^{n} I_i$ is the sum of idle times in machine B for the schedule produced by Johnson's method, when A and B are available simultaneously, then

(i) if $I_d - \sum_{i=1}^{n} I_i \geq 0$, the sequence produced by Johnson's method is still optimal, since it is not possible to produce any sequence with smaller makespan.

(ii) if $I_d - \sum_{i=1}^{n} I_i < 0$ then machine B can not have idle times. The critical path has no slacks and the makespan cannot be reduced by changing the sequence.

Further use of the basic Johnson's method has been made, trying to improve the efficiency of enumerative methods. Townsend (1977c) has used Lawler's procedure for the one-machine problem, combined with Johnson's rule to produce an efficient branch and bound algorithm for the two-machine flow-shop problem of minimising the maximum penalty. A two-machine based bound calculation, using Johnson's rule has been applied also to a branch and bound method, described in detail in Chapter 5.

Szwarc and Hutchinson (1977) have used the Johnson's rule as an approximate method (heuristic) for the three-machine C_{max} problem and have analysed the results statistically. The same rule has inspired also the heuristics of Palmer (1965) and of Campbell, Dudek and Smith (1970), described in the following section together with other heuristic methods.

3.2 Flow-shop heuristics

A large number of heuristics for flow-shop problems has been suggested over the years, covering different approaches (Page 1961, Palmer 1965, Campbell, Dudek and Smith 1970, Gupta 1971, Gupta and Maykut 1973, Krone and Steiglitz 1974, Bonney and Gundry 1976, Dannenbring 1977). The most important and representative ones are reviewed below and used for comparison with some new heuristics suggested.

Page (1961) has developed heuristics based on sorting techniques by individual exchanging, group exchanging, pairing and merging. In the exchanging case, starting with a given sequence (permutation), each successive pair of adjacent jobs is tested to see whether it should remain as it is or exchanged in case a lower makespan is achieved. If an exchange reducing C_{max} is obtained, the procedure is repeated. The same principle is used for exchanging the position of strings (chains) of jobs instead of single jobs. The pairing and merging of strings is based on replacing each successive pair of strings into a new ordered string, the order being the one with the best makespan. Repeating this procedure, a single chain containing all jobs is constructed.

The heuristic proposed by Palmer (1965) was inspired by a property of Johnson's method, where jobs with shorter first operations are positioned at the beginning of the solution sequence and jobs with shorter second operations are positioned at the end. To extend this concept for m-machine problems, he suggested giving priority to jobs having longer operations near the end or stronger tendency to progress from short to long operation times. A 'slope index' is defined as:

$$s_i = (m-1)t_{im} + (m-3)t_{i,m-1} + \ldots - (m-3)t_{i2} - (m-1)t_{i1}$$

or $$s_i = \sum_{j=1}^{m} (2j-m-1)t_{ij}/2$$

and jobs are ordered in decreasing slope index value.

Gupta (1971) has tried to improve this concept by defining the slope index as:

$$S_j = e_j / \min\{t_{jk} + t_{j,k+1}\}, \quad 1 \leq K \leq m-1$$

and

$$e_j = 1 \quad \text{if} \quad t_{j1} < t_{jm}$$

$$e_j = -1 \quad \text{if} \quad t_{j1} \geq t_{jm}$$

Bonney (1976) has used the idea of slope index in a different context. Two slope indices are defined for each job, a 'start-slope' and an 'end-slope' and the heuristic sequence is constructed by matching them in the best possible way, with the aim of minimising slack or machine idle times (slope matching method).

All these methods have the disadvantage of assuming some kind of standard pattern of processing times, like monotonic increasing or decreasing operation times. Bonney for example has calculated the slopes by linear regression of the start and finish times of operations. But if the job profile is irregular, it is not reasonable to describe it with a simple index. This is particularly important in small problems, where these methods perform poorly. In larger problems, the profile may approach some linear form and this agrees with the results reported by Bonney, that the slope matching performs better in larger problems.

An altogether different heuristic, inspired from Johnson's rule was suggested by Campbell, Dudek and Smith (1970). Their idea was to construct a surrogate problem by splitting the m-machines in two groups 1,....k, k+1....m, for k=1,2,....m-1. For each of the two groups, they considered the operations merging into an imaginery new longer operation, and solved the surrogate(relaxed)problem optimally with Johnson's method. This proved to be a very good heuristic and extensive studies (e.g. Bonney, 1976) have shown that it is a very good measure of comparison with other heuristics.

Dannenbring (1977) has used a weighting scheme similar to the slope index and to the Campbell et al method, with a simple improvement procedure of pair exchanges.

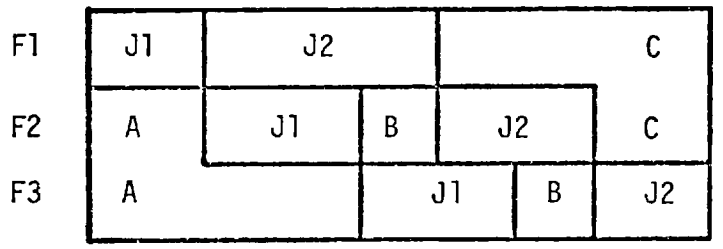
3.3 New heuristics for flow-shops with no-job-passing

Least slack heuristics (h1,h2,h3,h4)

For flow-shop problems where job-passing is not allowed, minimising total slack time between the operations, including the starting (A) and ending (C) slacks, is equivalent to minimising total makespan. From Figure 3.1 below, where A,B,C are machine idle times,

$mC_{max} = \sum_{j=1}^{m}\sum_{i=1}^{n} P_{ij} + A + \sum_{j=1}^{m} B_j + C$ and thus C_{max} is minimised when $A + \sum_{j=1}^{m} B_j + C$ is minimised. If a schedule is optimal in respect to C_{max}, then it is not possible to construct any other schedule with total slack less than that of optimal C_{max}.

Figure 3.1 Flow-shop scheduling slack times



The approach in this study is that heuristics based on slack (machine idle time), trying to create schedules with small value of $\sum_{j=1}^{m} B_j$, should not be based on a slope index. It is more reasonable to order the jobs not from their approximate slopes but from their total slacks. One would expect that by ordering the jobs with the view of selecting those that have small values of $\sum_{j=1}^{m} B_j$, C_{max} will be near the optimal. Matching the jobs with slope indices has, as already discussed, the drawback that it may be very poor for jobs with irregular profiles. Instead direct account of slacks incurred is suggested to be the decision criterion for this type of heuristic .

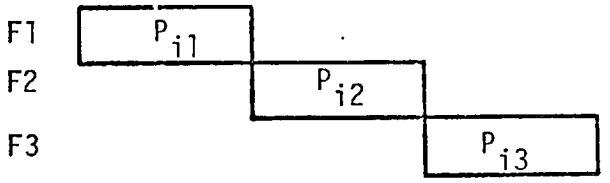
Four different functions of slack have been tried; front delay, back delay, total slack and weighted total slack where the weight coefficients indicate that delays incurred in the last machine are more

undesirable than delays in the preceding machines. These heuristics construct the solution by building up one partial schedule as a single chain, adding jobs at the end of the chain. A multiple chain approach, where many partial sequences are constructed and then joined together is not very promising, because by allowing left shifting without passing, the start profile of any partial sequence would be altered, and the decisions already taken would be invalidated. For the same reason, jobs cannot be added in the front of the single chain solution. Thus, only a single-chain sequential solution is reasonable, where, every time a job is added, the end profile of the chain is calculated and used for the next job matching-selection.

Description of the algorithm

Every job can be represented with a profile as in Figure 3.2 below:

Figure 3.2 Gantt chart of a job with three operations



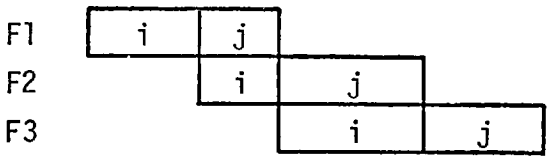
When the profile is 'unbroken', the job has 'no-waiting' between the operations. This will always be the case with the first job in the sequence. Subsequent jobs will fall in one of the four cases illustrated in Figure 3.3, on the following page.

The 'back delays' can be eliminated or reduced by left-shifting the operations while maintaining the same sequence of jobs on all machines (no job-passing). The front-delays cannot be eliminated by left-shifting.

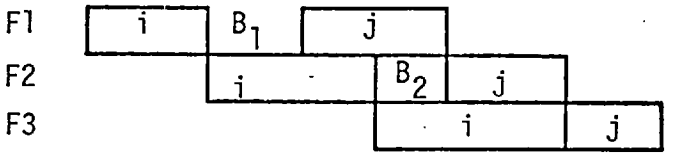
The four criteria (heuristics) used for selecting sequentially the successor j of job i, from the set of unscheduled jobs are:

Figure 3.3 Matching of job profiles

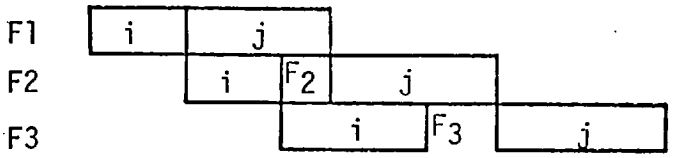
(a) Perfect matching of profiles (zero total slack)



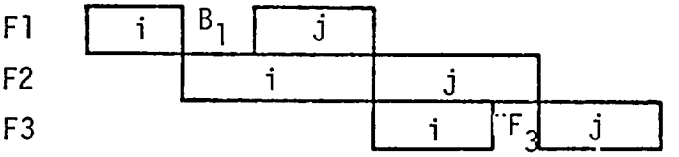
(b) Back delays B_1, B_2



(c) Front delays F_2, F_3



(d) Back and front delays B_1, F_3



h1 select the job with the least total slack

h2 select the job with the least total weighted slack

(the machine index number has been used as weighting coefficient).

h3 select the job with the least total front delay

h4 select the job with the least total back delay

Once the successor j of job i is selected its completion time on machine k is determined by the recursive relation:

$$C_{jk} = \max \{C_{ik}, C_{j,k-1}\} + P_{jk}, \text{ for all machines } k.$$

A complete sequence is constructed with this method, and two simple checks are used to see if an improvement is possible.

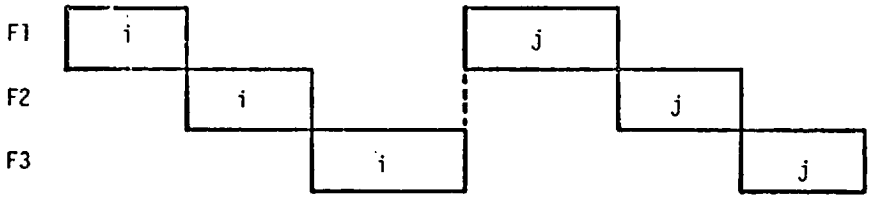
(i) Interchange of the last two jobs in the sequence

(ii) Use of all jobs successively as first job in the sequence.

A savings-based sequential heuristic (h5)

Another approach to the idea of matching jobs in a sequence is based on the amount of slack saved by a particular partial sequence. The savings are computed as the amount of slack saved by sequencing j after i, starting from the position where the first operation of job j begins at the completion time of the last operation of i (as in Figure 3.4 below).

Figure 3.4 Sequencing without left-shifting



If the schedule were constructed in that way, the makespan would be $C_{max} = \sum_{i=1}^{n} \sum_{j=1}^{m} P_{ij}$. The optimal schedule is the one that allows the savings to be maximised.

The heuristic algorithm can be described briefly as follows:

Select first job of sequence.

Select its successor by taking the job which, with all possible left-shifting, allows the highest slack savings.

Repeat for all jobs in the first position of schedule.

Computational experience

The least slack and savings-based algorithms have been programmed in FORTRAN IV for computer testing and evaluation. It has been necessary to code in FORTRAN IV also an algorithm for calculating the makespan of random sequences and the algorithm of Campbell, Dudek and Smith (computer codes in Spachis, 1978a).

Data for tests were generated for problems of 10 jobs 4 machines, 20 jobs 5 machines and 35 jobs 5 machines, with processing times from the low-variance Erlang distribution with shape parameter k=9 (see Appendix A about Erlang distribution) and from the high-variance Erlang distribution with k=1 (negative exponential), with the same average value. This distinction has been thought to be necessary because of the effects of the variance of processing times on job profiles. The computational results are given in Table A1 of Appendix A.

The performance of these heuristics has been evaluated by means of a relative ranking index. With this method, when a heuristic gives the best solution value, it is ranked with index 1. For the second best value, it is ranked with 2. In case of two or more heuristics giving the same value, they are all ranked with the same value, equal to the average (e.g. 2.5). With this method, the sum of ranks is the same for all test problems. Another criterion thought to be appropriate for the comparison of these heuristics is an 'error from the best known solution z_{min}' which is the ratio $e = (z_h/z_{min})-1$, where z_h is the heuristic solution value. Although the ratio e gives an indication of how far the heuristic value is from the best known solution, it does not give any information about the relative error, because it does not take into account the values from the other heuristics. This can be achieved by a 'relative error' criterion, defined as the ratio $e_r = (z_h - z_{min}) / (z_{max} - z_{min})$.

The average values of these criteria are given in Table 3.1. By observing the results in this table, one can see that in high variance problems, h2 (least total weighted slack) is the best for all three criteria used and all problem sizes. The savings based heuristic h5 is the poorest of all (poorer than 'random'). The 'least total back

delay' h4 also is also very poor. The second best heuristic is almost always the CDS (Campbell, Dudek and Smith).

Table 3.1 Performance of heuristics for flow-shops with no job-passing

High variance of processing times (E1)

	10 jobs 4 machines			20 jobs 5 machines			35 jobs 5 machines		
	Rank	e	e_r	Rank	e	e_r	Rank	e	e_r
Random	5.9	.176	.78	5.6	.164	.71	5.8	.240	.73
CDS	2.6	.030	.18	2.3	.034	.17	2.1	.036	.10
h1	2.8	.044	.15	2.8	.030	.22	2.3	.042	.14
h2	1.9	.008	.03	2.0	.018	.07	1.9	.030	.10
h3	2.9	.052	.21	3.8	.052	.29	3.7	.096	.30
h4	5.6	.136	.49	5.9	.182	.82	6.2	.274	.81
h5	6.3	.188	.77	5.6	.182	.78	6.0	.274	.80

Low variance of processing times (E9)

Random	5.7	.114	.82	5.8	.056	.67	6.1	.058	.87
CDS	3.3	.010	.17	2.5	.010	.13	1.7	.002	.05
h1	2.2	.002	.01	3.2	.014	.18	2.7	.016	.22
h2	2.5	.006	.13	1.6	.004	.06	2.1	.006	.14
h3	3.8	.032	.34	2.9	.014	.19	3.5	.018	.29
h4	5.6	.062	.58	7.0	.086	1.00	6.1	.062	.88
h5	4.9	.038	.39	5.0	.050	.56	5.8	.052	.76

$e = (z_h/z_{min})-1$

$e_r = (z_h-z_{min})/(z_{max}-z_{min})$

Sample size 5

In low variance problems, with Erlang k=9, a similar pattern emerges, though less clearly. In terms of ranking, h2 is the best in problems of 'medium' size (20 jobs and 5 machines). It is better than CDS in 'small problems' (10 jobs, 4 machines) but poorer than h1. At 'larger' problems (35 jobs, 5 machines) h2 is poorer than CDS and better than all the rest. The worst of all these cases are the RANDOM and h4 and then h5 (savings-based heuristic).

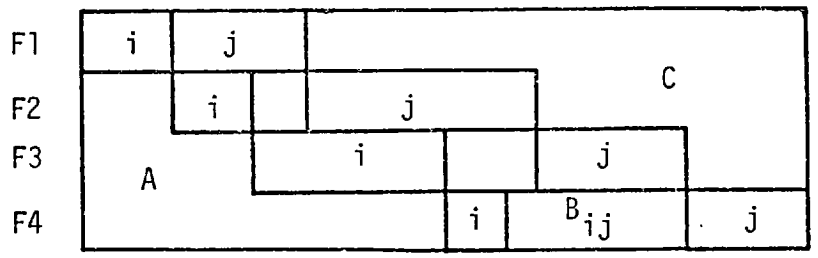
The conclusion is that h3, h4, h5 are clearly inferior, in all cases. Heuristic h2 is the best in high variance, and relatively good in low-variance where h1 and CDS can be good as well. In the low variance case, one should notice that the best three of the heuristics (h2, h1, CDS) have solution values very close to each other and that their ratios e and e_r are not significantly different.

3.4 New heuristics for flow-shops with no-waiting

The flow-shop without waiting is reducible into a directed Hamiltonian Circuit problem, known simply as Asymmetric Travelling Salesman Problem (ATSP). There are two methods available for defining the equivalent ATSP. One has already been described in Section 2.1 on reducibility of combinatorial problems. The other is based on the equivalence of two criteria of performance, namely 'minimise C_{max}' and 'minimise total machines idle times $\sum_{j=1}^{m} I_j$' or 'minimise total slack'.

Each job is represented by a vertex (node) and a dummy vertex is used as a starting and ending job for the sequence. Every pair of jobs i-j has some slack B_{ij} between i and j (see Gantt chart of Figure 3.5) which is used as the distance between vertices (nodes) i-j. The start slack A of job i is the distance from the dummy node to i, the end slack C is the distance from i to the dummy node and the total slack of schedule S is $A_f + \sum_s B_{ij} + C_e$ where A_f is the start slack of the first job in the sequence and C_e is the end slack of the last one.

Figure 3.5 Slack times for flow-shop no-waiting



The equivalent ATSP matrix is constructed with the following algorithm, where

S_{ik} start time of job i in machine k

C_{ik} completion time of job i in machine k

P_{ik} operation time of job i in machine k

n number of jobs

m number of machines

Algorithm

Step 1 $S'_{j1} = C_{im}$ for job i preceding job j

Step 2 $f = \min \{C_{ik} - S'_{jk}\}$ for all k

Step 3 $S_{jk} = S'_{jk} - f$

Step 4 $d_{ij} = \sum_{k=1}^{m} (C_{ik} - S_{ik})$

Step 5 Repeat for all possible pairs i-j, and for i=j $d_{ij} = \infty$ or - 1.

For the reduction described in Chapter 2 (with delays) replace step 4 by

Step 4 $d_{ij} = S_{j1}$

For the special cases of start and end jobs j

$d_{n+1,j} = 0$ (start)

$d_{j,n+1} = \sum_{k=1}^{m} P_{jk}$ (end)

Some good heuristics exist for the TSP, one of which is based on the construction of a shortest spanning tree, guaranteeing an error for the worst case behaviour to be 50% from the optimal (Christofides, 1976). This heuristic can be used in problems where the 'triangularity condition' holds true, but cannot be applied to the general asymmetric distance matrix resulting from the reduction of the flow-shop with no-waiting, because there is no P-class algorithm for finding the shortest spanning aborescence (the asymmetric version of the shortest spanning tree).

In this section, a number of heuristics have been designed, coded in FORTRAN IV and tested and the performance of each of them has been compared with the other heuristics, with RANDOM and with the exact solution, found by the branch and bound, depth-first algorithm of Little et al (1963). The computer codes for these routines are given in Spachis (1978a).

There are two basic methods that are suggested here, characterised by the way the heuristic sequence is built up as a chain.

(i) Sequential construction of a single chain

(ii) Multiple chain formation

Their common characteristic is that each job appears in one position only in the sequence, except the dummy job which has to be at the beginning and at the end of the heuristic solution.

The algorithms can be summarised as follows:

Step 1 Construct slack or distance matrix as described in preceding algorithm. This step sees how well two jobs i and j fit together, if i precedes j.

Step 2 Select one element of the chain (p,q)

Step 3 Exclude all elements that do not conform with the above condition, i.e. those that form a closed chain leaving jobs out. This can be achieved by excluding row p, column q, link (q,p)

Repeat steps 2 and 3 till a complete solution is reached.

The efficiency of this heuristic idea clearly depends on the distribution of processing times of the jobs. In a 'low-variance' case, the matching is likely to be good and the slack between all pairs i-j is likely to be low. By the same token the start and end slacks are likely to be fairly stable and the distribution of values of A and C is likely to be with low-variance. This suggests that two distinct cases should be considered, one with high variance (E_1) and one with low-variance (E_9). Step 2 of this algorithm is described below for a number of heuristics.

A constrained multiple chain heuristic (H1)

The basic idea in this heuristic is to select links with values constrained by a 'minimum covering level' (MCL), defined as the maximum of minima of rows and columns. This defines a surrogate problem where at least one element in each row and column of the working matrix has a finite value (open cell), and where all elements with values greater than MCL are not used (closed cells).

There is at least one row or column with open cell values equal to MCL only (since MCL is the maximum of minima). If a feasible solution is to be constructed for this surrogate problem, it has to include the link corresponding to the open cell equal to MCL (tie-breaking in lexicographic order). By selecting this link (p,q) and by excluding the elements described in Step 3 above, a new surrogate problem of reduced size is defined, from which another link of the heuristic solution is to be selected. A new MCL is defined, possibly by opening some of the closed cells. In case some row or column has no open cells at all, it is necessary to open cells up to a value higher than the previous value of MCL. The procedure is repeated n-2 times, until a complete feasible solution is constructed, using links of constrained value only, which is expected to lead to a complete solution of constrained length.

Least slack heuristics

In a heuristic approach constructing partial sequences by selecting the jobs with the best possible matching (least slack) at every stage, it is reasonable to expect that low values of C_{max} would be realised. The following methods of forming such heuristic solutions have been used.

- Multiple-chain heuristic (H2)

  Select the smallest open cell of the matrix, i.e. the least slack link. Break ties by selecting the last of the smallest values encountered in scanning the slack matrix. It is certain that this heuristic method will produce at the end a single-chain solution, since all links forming sub-tours have been excluded.

- Single-chain heuristic (H3)

  Add in front or after the existing partial sequence (chain)

the job-link corresponding to the smallest open cell of the slack matrix. Break ties as above.

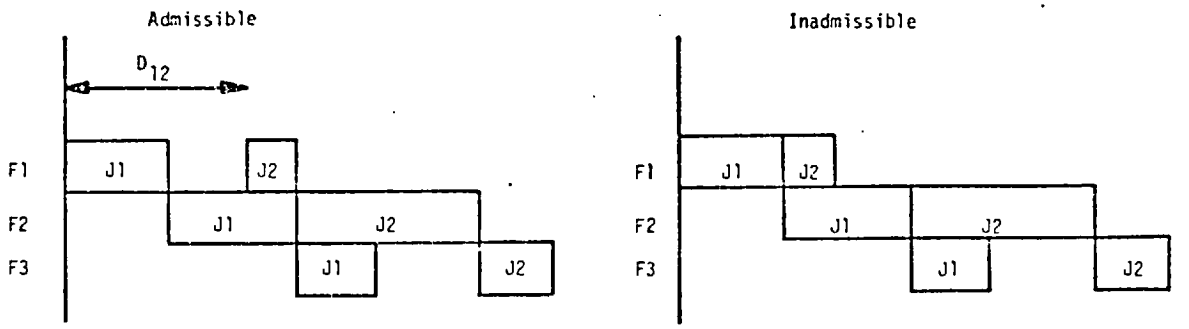
- Sequential heuristic (H4)

  Start from job i. Add after it the job j with least slack i-j. Repeat for all jobs i in first position. It is the same principle as the one in the heuristic described for flow-shop with no job-passing.

A multiple-chain savings-based heuristic (H5)

This heuristic is basically the same as that described in the preceding section for the case of flow-shops with no job-passing. The difference is that since the profiles must remain unbroken, only limited left-shifting is allowed.

Figure 3.6 Flow-shop scheduling operations left-shifting



A_j start slack for job j

C_i end slack for job i

B_{ij} slack between i-j, after left-shifting

Slack savings in this case are $s_{ij} = A_j + C_i - B_{ij}$. These savings are related to delay savings s'_{ij} as follows:

$$s'_{ij} = s_{ij}/m$$

and the solutions produced from the two different savings matrices are identical. Variations of this heuristic could be obtained by calculating a savings score, using weighting coefficients or by forming a sequential or a single-chain solution.

Computational experience

All the heuristics described above have been tested both with high and low-variance data, for problems of 10 jobs 4 machines, 20 jobs 5 machines, 35 jobs 5 machines. Heuristics H1, H2, H3 and H4 have been tested with both slacks and delays. In 30 problems tried with these four heuristics (two tests with each heuristic, one with slack matrix and one with delay matrix), the use of slacks gave better results than the use of delays in 116 out of the 120 test cases. In the remaining four cases, the solutions obtained with slacks were only marginally poorer, while the solutions with delays were not found to be the best of all. This is a clear case of dominance of the method using slacks. This superiority of slacks compared with delays probably lies in the fact that more information is used with slacks for the construction of the heuristic solution, without increasing computational costs.

These results as well as the results from RANDOM and from the exact method are given in Table A2 of Appendix A. The measures of performance used in the preceding section on heuristics for flow-shops with no-job-passing, have been used also for the case of no-waiting. The values of average ranking, error e and relative error e_r are summarised in Table 3.2, on the following page.

Observation of the results of this table shows that the overall best heuristics are the constrained multiple-chain formation H1 and the savings based multiple-chain H5, in problems of both high and low variance. The least slack heuristics of multiple-chain (H2), single-chain (H3) and sequential (H4) are almost always dominated, overall worst being the H3. No conclusions can be drawn on the effects of the variance of processing times. The relative position of H1 and H5 is not affected, while minor changes take place for H2, H3, H4.

Table 3.2 Performance of heuristics for flow-shops with no-waiting

High variance of processing times (E1)

	n=10, m=4			n=20,m=5			n=35,m=5		
	Rank	e	e_r	Rank	e	e_r	Rank	e	e_r
Random	6.0	.353	1.00	6.0	.483	1.00	6.0	.581	1.00
H1	1.8	.048	.16	2.6	.061	.12	1.9	.043	.07
H2	3.2	.100	.29	3.5	.104	.22	3.0	.056	.09
H3	4.3	.126	.38	3.7	.116	.23	4.1	.089	.14
H4	3.7	.064	.18	3.9	.088	.19	2.1	.048	.08
H5	2.0	.032	.09	1.3	.042	.09	3.9	.084	.14

Low variance of processing times (E_9)

Random	5.8	.156	.97	6.0	.176	1.00	6.0	.245	1.00
H1	1.9	.017	.14	2.5	.025	.15	2.3	.024	.10
H2	4.7	.075	.54	2.6	.034	.19	3.1	.033	.14
H3	4.1	.072	.45	4.3	.059	.31	3.7	.038	.17
H4	2.6	.035	.21	3.2	.033	.20	2.6	.025	.10
H5	1.9	.016	.12	2.2	.026	.15	3.3	.033	.14

In small problems of 10 jobs 4 machines the savings based heuristic H5 is better than H1 in terms of error and relative error, though not in average ranking. In problems of 'intermediate'size (20 jobs 5 machines) heuristic H5 is better than H1 in all counts. In the 'larger' problems of 35 jobs 5 machines though heuristic H1 is clearly the best. As second best appears the least-slack sequential heuristic H4, third in order comes the multiple chain H2 and H5 is only fourth. These results indicate that the advantages of constructing a solution with the constrained multiple chain heuristic (H1) are materialised in larger problems only where by controlling the maximum values of the links selected, a solution of low C_{max} is obtained. In contrast

the other heuristics, based on least-slack or on savings, start with low cost (contribution) links but are pushed to include very unfavourable links near the end. It is possible that with further investigations the performance of heuristic methods H1 and H5 could be improved. The constrained multiple chain heuristic (H1) involves a number of decisions which could be resolved in a number of ways other than the one used. Similarly, the savings based heuristic (H5) could be improved by investigating minor changes in the calculations of savings. A savings score could be used, calculated on weighted, quadratic and other expressions of the actual savings.

Another type of heuristic, inspired by the method of group exchanges of Page (1961) and by the 'λ-optimal' tour heuristic for Eucledian TSP, has also been considered, whereby starting from an arbitrary chain/solution λ-links are taken out, all possible combinations of the λ-sub-chains are tried and the process is repeated for all possible combinations of the λ-links. This method with $\lambda=2$ or $\lambda=3$ is reported to be efficient for the TSP. However, a preliminary investigation for flow-shops with no-waiting (examples solved manually) was rather inconclusive, probably due to the assymetric nature of the matrix, and it has been decided to drop the idea.

CHAPTER 4

PROBABILISTIC AND WORST CASE ANALYSIS OF SINGLE PASS JOB SHOP HEURISTICS

4.1 Evaluation of single pass job-shop heuristics

4.2 Description of the algorithm and decision rules

4.3 Computational experience

4.4 Worst case behaviour and probabilistic analysis of heuristics

4.1 Evaluation of single pass job-shop heuristics

Single pass heuristics for the general job-shop problem are of special importance and interest. The simplicity of implementation and the large size of problems that can be tackled with them allow many practical applications at shop floor level. The assessment of their performance can take a number of forms. The most common has been a qualitative comparison whereby one particular heuristic is either better or worse than another, evaluated on the basis of solution values obtained over a number of problems, usually with randomly generated data. An alternative method of assessment could be based on the 'Worst Case Behaviour' of a heuristic. This information if obtainable, would certainly be useful, guaranteeing a maximum error, although it would give no indication of the actual performance of a heuristic. It is believed that this expected performance is the most useful information about a heuristic in a given job-shop environment. A probabilistic analysis is proposed here as a more elaborate procedure, which will distinguish the various problem structures and will quantify for each structure the actual performance of heuristics in terms of an expected value and of a frequency distribution.

The structure of the problem depends on a number of parameters:

(i) Size as an indicator of complexity

(ii) Variance of processing service times

(iii) Sequence constraints or transfer matrix

(i) Size

The size can be described by $O\{(n!)^m\}$ which is an upper bound to the set of feasible semi-active solutions.

The following problem sizes have been investigated:

Jobs	Machines	$(n!)^m$	
4	3	.138	E5
6	3	.373	E9
8	4	.264	E19
10	4	.173	E27
20	5	.852	E92
35	5	.118	E201

(ii) Service time distribution

A general distribution G (μ,σ) may be approximated with an Erlang distribution (Appendix A), with parameter k and mean μ, assuming that the higher moments have no significant effect on the processing time values. The service times are expressed always as integers to allow faster computations since the formulation is not affected by using larger integer values instead of real numbers (e.g. 45 time units instead of 4.5). The values of variance used, as measured by the coefficient of variation $V = \sigma/\mu$, are as follows:

1.00 (Erlang, k=1)
.82 (Uniform Rectangular)
.50 (Erlang, k=4)
.33 (Erlang, k=9)
.17 (Erlang, k=36)

(iii) Precedence constraints

The simplest form of transfer matrix is that of a flow-shop with the same sequence for all jobs. At the other end, there is a completely random routing, where given a partial sequence S, the probability p that the next of the remaining r' operations is operation k, is p=1/r'. This more general case has been adopted for the experiment, where the number of alternative (possible) routes is m!, and the probability of any of these routes is 1/m!

The simplest evaluation possible is an average ranking whereby the best value corresponds to the heuristic with rank 1. The second best has a rank 2. In case of equal values, the average rank is assigned to both, e.g. 1.5.

A more elaborate measurement of the performance of heuristics, based on a probabilistic analysis of the error from the optimal, is suggested here. A feasible solution obtained with a heuristic is certainly appropriate only for the particular problem(set of jobs and machines) under consideration and does not allow any comparison with other similar problems. To overcome this difficulty, a 'normalisation' of the solution is desirable. This could be a ratio of the solution value over the exact (optimal) solution, or a percentage of 'excess' values over the yet unknown optimal. It is argued that the best is a bracket value based on the feasible solution available and specifying the maximum possible distance of the optimal. This Bracket for the Optimal Solution will be referred to as 'BOS'. BOS has the advantage of being based on the known feasible solution and not on the unknown optimal.

H represents the single-pass heuristic used

$\nu(H)$ solution value with H

b_e overall lower bound to the problem solutions

BOS bracket for optimal solution (%)

$$BOS = (\{\nu(H)-b_e\}/\nu(H)) \ 100$$

Empirical information on BOS has been derived from an experiment with the model described in the following section, where the sample size chosen for each problem structure is given in Table 4.1 below.

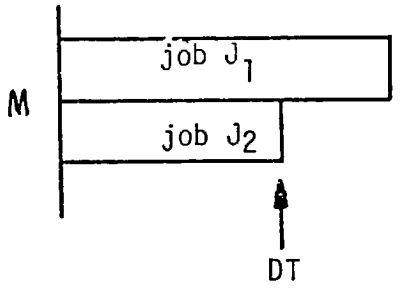
Table 4.1 Sample sizes for probabilistic analysis of heuristics

Problem		Coefficient of variation				
Jobs	Machines	1.00	.82	.50	.33	.17
4	3	10	10	10	10	10
6	3	10	10	10	10	10
8	4	10	10	10	10	10
10	4	20	20	20	20	20
20	5	20	20	20	20	20
35	5	5	5	5	5	5

4.2 Description of the algorithm and decision rules

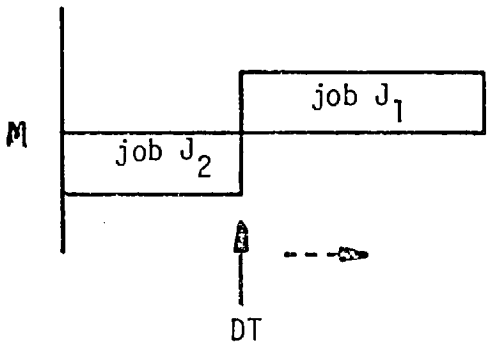
An algorithm for the generation of active schedules, implementing the method of Giffler and Thomson has been taken from King (1975), with some modifications. Before describing the algorithm it is necessary to discuss some notions and terminology to be used. The discussion can be facilitated by using Gantt charts. 'Conflict' between two or more operations arises when the jobs are contenders for processing on the same facility M at the same time (Figure 4.1).

Figure 4.1 'Conflict' jobs



A time index, called 'Datum Time' (DT) indicates in the Gantt chart whether the position of an operation in a partial schedule is fixed or not. Operations completed before DT are fixed decisions while those with $c_i \geq$ DT are tentative ones. In Figure 4.1, jobs J_1 and J_2 compete for machine M. The conflict is recognised when DT is equal to min (c_i). Resolving the conflict by deciding to give priority to J_2, is represented in the Gantt chart of Figure 4.2, and DT is updated to the next earliest completion time.

Figure 4.2 Conflict resolution



The algorithm used is described below (where N_s is the number of times DT is updated, c_i is the completion time of an operation i and p_i is the processing time of operation i) and the related flow-chart is given in Figure 4.3, on the following page.

Algorithm for active schedule generation

Step 1 Initialisation of procedures.
Define a matrix A=0 of nmN_s elements, where each row has nm elements, in m blocks, representing the m machines, and each block has n elements (operations). Define the set S of schedulable operations by entering the completion times of operations without predecessors in the appropriate matrix positions. Define the empty Partial Schedule by setting DT equal to the smallest entry $DT=\min(c_j)$, $j \epsilon S$

Step 2 Conflict recognition.
If there is one conflict or more GO TO 3.
If not, GO TO 5.

A conflict exists in a machine block, if at least one entry is equal to DT and at least another one is greater or equal to DT. All entries greater or equal DT form the conflict set S.

Step 3 Definition of 'conflict machine' m*, where the conflict occurs (if there is more than one conflict, select one).

Step 4 Conflict resolution.
Select, with some rule, called 'Heuristic', one of the conflict jobs(operations)to be processed first. Record this decision by entering its earliest completion time c_k in the same column of the next row of the matrix A. Update the earliest completion times of the remaining conflict jobs $c_j = c_k + p_j$, enter all c_j in the corresponding columns of the next row.

Step 5 Update set of schedulable operations.
For all jobs with $c_i = DT$ (implying completion of an operation) calculate the earliest completion time in their next operations c_n, at a machine different from the one where $c_i = DT$
$c_n = DT + p_n$ (p_n:processing time of next operation). Enter c_n in the appropriate column of the next row.

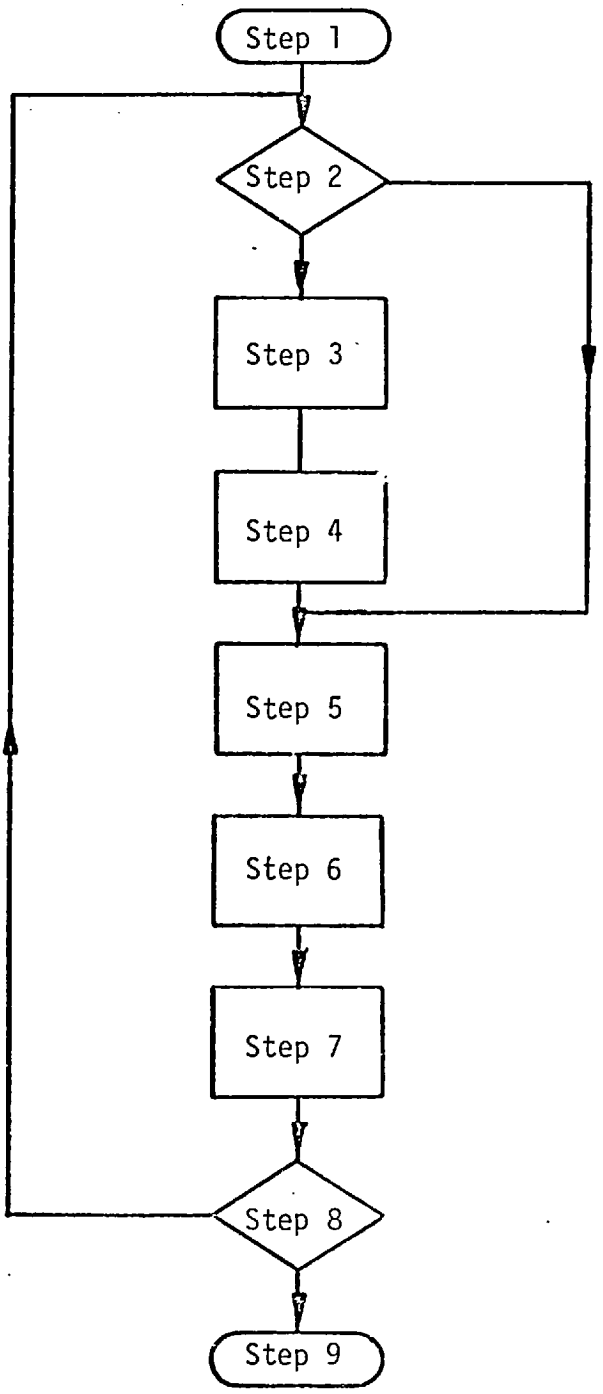
Step 6 Update the remaining part of next row.
Enter unchanged all other completion times in the same column of the next row.

Step 7 Update the datum time.
Set new DT equal to the smallest entry c_i that is greater than the current value of DT.

Step 8 Check for completeness of schedule.
A complete feasible active schedule is indicated by all entries in the current row being non-zero and by DT being equal to the largest c_i. If the schedule is not complete, GO TO 2.

Step 9 Terminate procedures.

Figure 4.3 Flow chart for single-pass active schedule generation



Description of Heuristics

The algorithm described in the previous section, resolves the conflicts that arise with one of the heuristic decision rules to be described below. Once a decision is made, then it is not reconsidered or revised later (single-pass procedure), and the rule used is called a 'single-pass heuristic'. It is apparent that single-pass heuristics are 'good algorithms', since the number of steps (iterations) required (equal to the number of conflicts) is polynomially bounded by O(nm). The following active scheduling heuristics have been used.

ECT Earliest Completion Time.

Select first the job with the earliest completion time. In case of tie, i.e. more than one job with the same smallest ECT, select the first in lexicographic order (randomly, since labelling of jobs is random).

FCFS First Come First Served

Select the job that came first to the current conflict machine i.e. the one that finished earlier its preceding operation, at some other machine. Tie breaking, as above, in lexicographic order. This is tantamount to an 'Earliest Start Time' rule and the solution obtained is the same as a non-delay scheduling one. Non-delay schedules are those where no machine is left idle, while a job is waiting for processing. The difference is that in non-delay scheduling, no conflict would be recognised, unless some jobs had the same earliest start time. Thus, if an enumerative method was to be used, some active schedules might be missed, while with FCFS, all alternative conflict resolutions are identified.

SPT Shortest Processing Time

Select the shortest operation first (tie breaking as above).

GEN A general decision rule is to assign priorities to the conflict jobs according to some function of completion times, start times, processing times, idle times and other variates, and select the one with the highest priority.

The underlying idea for this general rule is the attempt to relate desired results and the decision variables. The criteria of performance that are thought to be the most relevant to real problems are:

(i) meeting due dates

(ii) minimising work in progress

(iii) minimising idle time of facilities-resources

Since no due dates are used in this formulation of the problem, an arbitrary due date D is set, which acts as an incentive to maximise the number of jobs to be completed in the given timespan D (lateness-tardiness criteria). Minimising the work in progress is the same as minimising the time jobs spend waiting in queues ($\sum_{i=1}^{n} W_i$). Minimising the idle time of facilities $\sum_{j=1}^{m} I_j$ is equivalent to minimising makespan C_{max}. These objectives are independent and probably conflicting. An overall optimisation strategy should be one balancing these criteria by the use of some composite objective function with subjectively set weighting coefficients. The decision rule (priority function) appropriate for this type of criteria should be also composite. The following form is suggested:

For each job i a priority index P_i is set-up as the weighted sum of all criteria calculated as deviations from given values.

$$P_i = w_1P_1 + w_2P_2 + w_3P_3$$

where P_{ri} processing time of remaining operations of job i

P_{fi} processing time of finished operations of job i

P_m service time of operations available for processing in the next machine

$$P_1 = (1-\delta)(D-T-P_{ri})^{-x} + \delta(D-T-P_{ri})^{x}$$

with $\delta = 1$ if the job is already late, or else 0

$$P_2 = (T/P_{fi})^{y}$$

$$P_3 = z/P_m$$

The parameters that have to be defined are w_1, w_2, w_3, x, y, z.

4.3 Computational experience

The experiment has been carried out with data generated randomly from the processing times distributions described in Section 4.1. A summary of the results collected from this experimental study of the heuristic ECT, FCFS, SPT is presented below. Table 4.3 gives the average relative ranking for makespan C_{max}, Table 4.4 gives the average BOS and Table 4.5 gives the average ranking for average waiting $\bar{W}$ (next page).

Results from the use of the generalised composite heuristic GEN were obtained for a few problems where it became clear that the solution values could be very good when the trade-offs were balanced but only after a lot of search for defining the best combination of its parameters and coefficient values. This being a very cumbersome and rather expensive task, it has been necessary not to take the investigation of this heuristic any further in this chapter.

The cost of obtaining the solutions as presented in Tables 4.3, 4.4 and 4.5 and a host of related information is almost negligible. Table 4.2 below is indicative of the CPU times required, when a lot of extra information, not directly needed, is computed (e.g. bounds for every branch, proportions and others).

Table 4.2 Problem dimensions and computational cost of single pass heuristics

Problem Size		O(nm)	No of conflicts or decisions		CPU time in seconds CDC 7600 FTN compiler
Jobs	Machines		Typical case	Range	
35	5	175	150	148-158	1.10
20	5	100	80	78-83	0.30
10	4	40	30	27-32	0.04
8	4	32	20	18-22	0.025
6	3	18	10	9-12	-
4	3	12	6	5-8	-

Table 4.3 Average ranking for makespan values of single-pass heuristics

		Processing times distribution														
Jobs	Mach.	E_1			UR			E_4			E_9			E_{36}		
		ECT	FCFS	SPT	ECT	FCFS	SPT	ECT	FCFS	SPT	ECT	FCFS	SPT	ECT	FCFS	SPT
4	3	1.85	1.6	2.55	1.85	1.7	2.45	1.85	1.5	2.65	2.0	1.6	2.4	1.55	1.75	2.7
6	3	1.9	1.6	2.5	1.65	1.7	2.65	1.35	2.1	2.55	1.5	1.8	2.7	1.45	1.65	2.9
8	4	1.95	1.45	2.6	1.6	1.7	2.7	1.5	1.6	2.9	1.4	1.85	2.75	1.4	1.65	2.95
10	4	1.4	1.3	2.8	1.6	1.6	2.8	1.65	1.65	2.7	1.5	1.6	2.9	1.55	1.55	2.90
20	5	1.0	2.5	2.5	1.35	1.65	3.0	1.4	1.6	3.0	1.5	1.5	3.0	1.2	1.8	3.0
35	5	1.2	2.2	2.6	1.8	1.2	3.0	1.8	1.2	3.0	2.0	1.0	3.0	2.0	1.0	3.0

Table 4.4 Average BOS (Bracket for Optimal Solution %)

4	3	21.2	18.2	28.0	19.6	21.1	29.7	19.9	15.6	31.1	21.2	21.2	31.3	15.9	21.0	30.4
6	3	20.3	16.7	26.5	19.7	17.5	26.9	15.1	19.4	26.5	18.2	19.6	27.1	18.1	19.4	30.4
8	4	22.6	16.4	27.6	21.5	21.0	31.4	23.8	23.4	37.0	19.2	24.0	32.7	20.6	20.5	31.0
10	4	25.6	15.8	33.9	18.3	17.9	29.4	23.1	21.6	31.2	17.7	17.6	30.7	17.9	17.5	29.5
20	5	20.6	15.5	30.3	14.6	13.2	25.6	13.6	14.2	27.2	12.0	12.0	25.0	12.3	15.9	26.9
35	5	21.0	12.0	30.0	13.6	9.8	24.4	12.0	15.0	25.0	9.0	10.0	23.0	10.0	7.0	25.0

Table 4.5 Average Ranking for $\bar{W}$

4	3	1.3	2.5	2.2	1.25	2.6	2.15	1.35	2.05	2.6	1.6	1.95	2.45	1.5	1.55	2.95
6	3	1.2	2.7	2.1	1.1	2.5	2.4	1.46	2.08	2.46	1.5	1.8	2.7	1.3	2.0	2.7
8	4	1.44	2.28	2.28	1.4	2.0	2.6	1.05	2.4	2.55	1.25	1.95	2.8	1.65	1.6	2.75
10	4	1.15	2.3	2.55	1.45	1.75	2.8	1.15	2.25	2.60	1.3	1.8	1.9	1.33	1.72	2.95
20	5	1.05	2.38	2.57	1.4	1.65	2.95	1.2	1.8	3.0	1.45	1.55	3.0	1.85	1.15	3.0
35	5	1.2	2.2	2.6	1.8	1.2	3.0	1.8	1.2	3.0	1.8	1.2	3.0	2.0	1.0	3.0

It should be noted that the solution time is the time starting from and including the definition of the initial set of schedulable operations to the completion of the feasible (heuristic) solution. It does not include 'input' and set-up or compilation times, but includes 'output' time. The machine used is a CDC 7600, with the FTN compiler.

Discussion of results

In terms of average ranking for makespan values (Table 4.3), the rule SPT gives always the poorest value, i.e. it has the highest value of average ranking, with increasingly poor performance for increasing problem dimensions. In the smaller problem sizes, the average ranking for SPT tends to be worse for the low variance data.

Heuristic ECT performs better with low variance data for smaller problems and with high variance data for larger problems. The reverse trend seems to take place for FCFS. The emerging pattern is that for smaller problems with high variance of processing times, FCFS has the best average ranking, while with low variance, ECT is clearly better. For larger problems, with high variance, ECT is better while the result of the comparison is less clear in low variance cases.

The impact of both the problem size and variance of processing times is more noticeable on the value of the average BOS for the single pass solution.

(i) In small problems, for each heuristic, the values of the BOS are higher than in the larger problems, something one might intuitively expect. The reason appears to be that, due to precedence constraints, idleness is introduced in the machines, and there are not any jobs in the set of schedulable operations to fill

the gaps. The result is that C_{max} is substantially higher than the max $(\max_i \sum_{j=1}^{m} P_{ij}, \max_j \sum_{i=1}^{n} P_{ij})$, which is the base of lower bound calculations.

In larger problems, C_{max} is much nearer to the Lower Bound, because there are always jobs which might fill the gaps (or the machine idle times).

(ii) The value of the variance is also significant. For the same problem size and heuristic, one can see a tendency for larger BOS at high variance values, which might be explained by the consideration of job-delays or machine idle times. For a given transfer matrix, low variance implies small differences in processing times, and high variance may induce longer machine idle times which are not cancelled by subsequent shorter operations.

(iii) Apart from the above two general remarks on the impact of these two parameters on the performance of the active scheduling heuristics, a general remark is that in all cases in terms of BOS, the SPT has the poorest performance. In high variance, rule ECT has poorer performance than the FCFS, while in low variance, this is reversed. The performance of the same heuristics, measured by the average ranking for the average waiting $(\bar{W},\bar{F},\bar{C},)$ per job is somewhat different from that for BOS values, as can be seen from Table 4.5. In small problem sizes (up to 10 jobs 4 machines) ECT is the best, for all the values of the coefficient of variation V. The rule SPT is better than FCFS only

at high V, and clearly poorer in low V. In larger problems SPT is the poorest for all values of variance and ECT is the best for all V at problems of 10 jobs 4 machines and 20 jobs 5 machines, and for high values of V at problems of 35 jobs 5 machines.

4.4 Worst case behaviour and probabilistic analysis of heuristics

The maximum value of BOS observed for each problem size and distribution of processing times is presented in the following Table 4.6.

Table 4.6 Worst case value of BOS for single-pass heuristics (%)

Heuristic	Jobs	Machines	E_1	UR	E_4	E_9	E_{36}
ECT	4	3	32	34	47	39	34
	6	3	37	31	27	27	28
	8	4	39	31	34	29	31
	10	4	46	30	35	31	26
	20	5	32	23	25	22	18
	35	5	30	16	16	11	15
FCFS	4	3	39	42	22	37	32
	6	3	34	35	39	33	31
	8	4	34	41	32	30	26
	10	4	38	28	33	33	34
	20	5	28	27	26	20	23
	35	5	17	16	15	14	10
SPT	4	3	38	44	48	42	41
	6	3	46	42	38	39	36
	8	4	40	46	46	41	40
	10	4	51	39	44	39	39
	20	5	39	34	31	33	32
	35	5	31	31	27	26	29

From the figures in Table 4.6 one can see that both size and variance have some impact on the worst case value of BOS. It is worth noting that the worst case error (BOS) for all the heuristics is $\{\nu(H)-b_e\} / \nu(H)=0.56$, and that the highest values are encountered in the smaller problems while the values of BOS for large problems with low variance are fairly small. There is a clear trend for BOS values to decrease with increasing problem size, for all heuristics, problem sizes and data structures. This is of great significance, because, after all, heuristics are needed for the larger problems rather than the small ones where exact solutions can be computationally feasible.

A tentative explanation is that at large problems where more jobs are involved in each conflict, delays can be avoided by this increased availability and thus the makespan values are closer to the values of lower bounds calculated as described in the preceding section.

Another point of interest is that the higher values occur in problems with high variance of processing times, probably due to the fact that a 'wrong' scheduling decision may result in long delays.

As an alternative to the analysis of results from observation of Tables 4.4 and 4.6, a probabilistic analysis is also possible. The basic idea is that an empirical distribution function F(x) can be defined as

$$F(x) = \text{Prob}(X \leq x)$$

where X is the value obtainable by the heuristic and x is the variate which can take values $0 \leq x \leq x_{max}$, x_{max} representing the worst value obtained. Obviously, the larger the sample size used, the more reliable the probabilistic information (described by F(x)) will be. As an example, the distribution function for the heuristic ECT has been plotted, at problems of 10 jobs 4 machines, 20 jobs 5 machines and 35 jobs 5 machines, for Erlang with k=9 in Figure 4.4 and for Erlang with k=36 in Figure 4.5. The computational results used for plotting these figures are given in Appendix B.

Figure 4.4 Distribution function of BOS (%) for ECT (E_9)

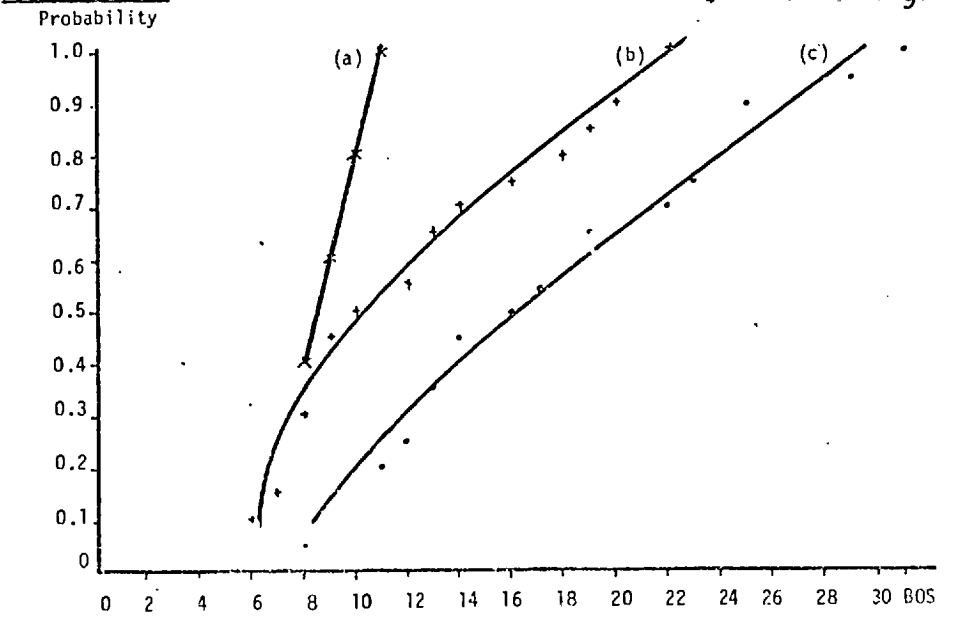
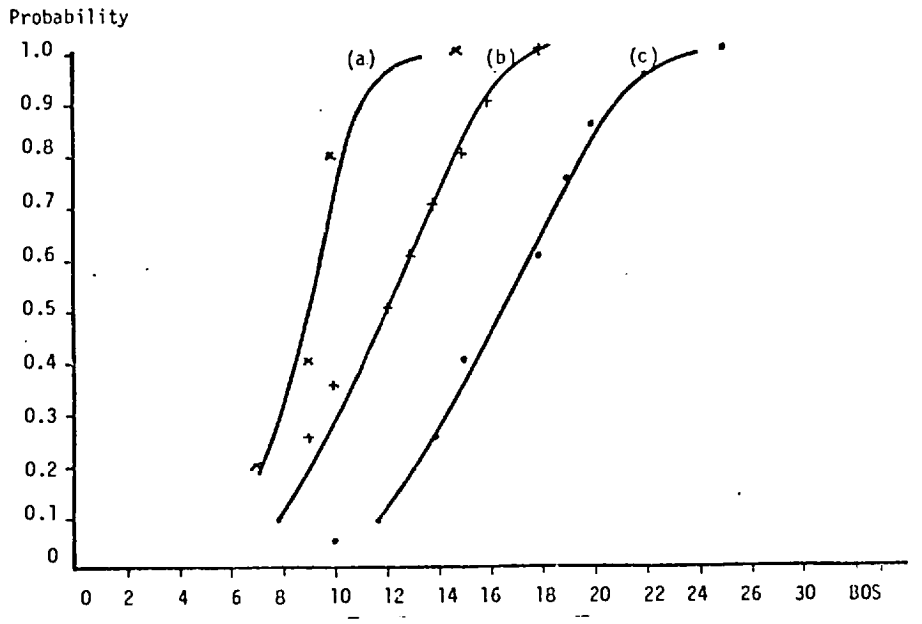


Figure 4.5 Distribution function of BOS (%) for ECT (E_{36})



(a) 35 jobs 5 machines

(b) 20 jobs 5 machines

(c) 10 jobs 4 machines

These diagrams show a clear dominance relation for the different problem sizes. For any level of probability $F(x)>.2$

$$x_{35,5} < x_{20,5} < x_{10,4}$$

or for a given value x of the BOS

$$F_{35,5} > F_{20,5} > F_{10,4}$$

For problems of 20 jobs 5 machines and 10 jobs 4 machines, where the sample size used was N=20, the difference $F_{20,5} - F_{10,4}$ is fairly stable. The implication is that some form of interpolation for problems of intermediate size is not only feasible but also meaningful . One need not expand the experiment, in order to obtain probabilistic information for other problem sizes but can use with some accuracy the interpolation results.

Another point of interest is the form of F(x) for each problem size. It appears that a linearity assumption is quite reasonable:

$$F(x) = \alpha + \beta x \quad \text{for } .2<x<1.0$$

CHAPTER 5

ANALYSIS OF ENUMERATIVE JOB-SHOP HEURISTICS

5.1 Parameters of local neighbourhood search (LNS)

5.2 Lower bounds

- 5.2.1 One-job and one-machine based bounds
- 5.2.2 Complexity and limitations of bound calculations
- 5.2.3. A routine for calculating two-machine based lower bounds
- 5.2.4 Fictitious bounds and estimates of the optimal solution

5.3 Design of the experiment

5.4 Computational experience

- 5.4.1 Lower bounds
- 5.4.2 Number of iterations and CPU time
- 5.4.3 Problem complexity
- 5.4.4 Speed of tree search
- 5.4.5 Worst case and probabilistic behaviour of heuristics in LNS with limited computational resources

5.5 Solution improvements models and stopping rules

- 5.5.1 Modelling the solution inprovements process in LNS
- 5.5.2 Stopping rules

5.1 Parameters of local neighbourhood search(LNS)

The probabilistic analysis of single-pass heuristics is useful because it provides probabilistic information about the expectation of the performance of these simple heuristics, at very low computational cost (small fractions of 1 second), as opposed to exact solutions where the required investment on computer resources may be infeasible. For cases, where higher accuracy of solution is desired and the amount of computer resources available is larger than the 1 second required for the single-pass heuristics, (of the order of 0.1 up to 1 minute) then approximate methods based on partial enumeration become feasible.

The partial enumeration is based on the search of 'local neighbourhoods'. For the general job-shop problem under investigation, in a tree-like representation of the solution space, a local neighbourhood corresponds to a sub-tree, i.e. it is defined as 'all active solutions with a common fixed partial sequence'. In terms of permutations, it is the subset of the active solution space with a number of elements fixed in position. The idea of a partial enumerative solution is to search one or more of these neighbourhoods to find the best possible solution. As in the previous chapter, the objective of the study is not only to find the best possible solution but mainly to try to describe the behaviour of the heuristic involved in quantitative terms and express the expected performance in probabilistic terms.

There are two basic methods of searching a local neighbourhood, depending on its definition. In the case it is defined by interchanging the positions of some pair of elements from a complete sequence, leaving the others as they are, the method of search is usually referred to as 'combinatorial analysis' (Section 2.3).

This method is feasible for permutation problems, where one permutation is a complete solution to the problem, and where the calculation of the value of the objective function/measure of performance is derivable from the value of the previous sequence, through a simple transformation.

The general n-job m-machine job-shop scheduling problem needs m permutations for its description and the calculation of the corresponding objective function is quite complex. Besides, one wants to search only the set of active solutions which is easily defined in graph terms but not in terms of permutations. If a pair of elements is interchanged, then in the general case, the computations required to find the new value of the objective function have to be repeated for a large part of the sequence, because no simple transformation is possible (as for example in the Travelling Salesman Problem). When the neighbourhood is described by a fixed partial sequence at the top of a 'tree' representation, then the natural method of solution is a 'branch and bound' method. A number of parameters is needed for describing a branch and bound method.

Nodes and branches

The basic idea of the method is to partition a set of solutions into disjoint sub-sets. A node represents a conflict, or a decision point, and branches are the alternative decisions/conflict resolutions.

There are two basic methods of partitioning the problem and the solution space.

(i) A binary tree has been suggested by Charlton and Death (1970a). In this method every node has a pair of branches, so that the partitioning is always generating two disjoint subtrees. Since any two operations to

be performed on the same machine cannot be performed simultaneously, the set of all schedules can be divided into two subsets, one in which the pair of operations is performed in one order and the second in which the order is reversed. The potential advantage of this method is that it might allow early settlement of crucial decisions after which the following decisions might be more obvious. This method apparently results in trees with a high number of levels (generations of nodes).

(ii) A more efficient partitioning method is the one used by Brooks and White (1965) and Florian et al (1971) and described in Section 4.2, where every node corresponds to a set of m' conflicting schedulable operations, resulting in m' branches.

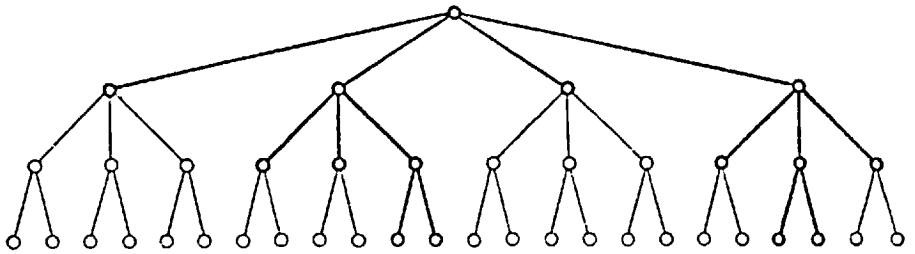
Selection of node and conflict resolution rule

It is the rule by which the node to be used next is determined. There are two basic types of branching possible, named 'depth-first' and 'breadth-first' illustrated in Figure 5.1. There is also a'best-bound-first' (branching from the active node with the least value of lower bound) which is a mixture of the other two methods (comparison of the search strategies can be found in Ibaraki, 1976b).

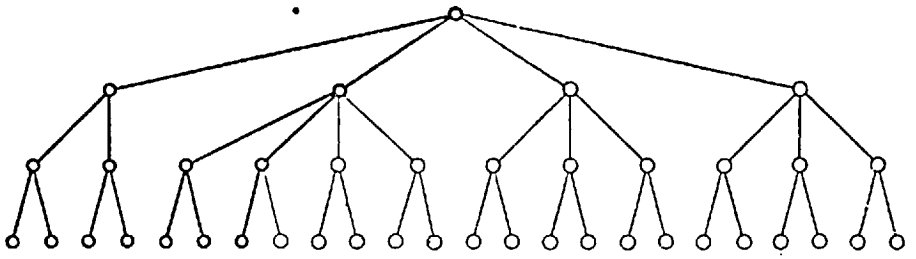
The depth-first (i.e. branching from the last active node) is more expensive in terms of computer time required, while the breadth-first (branching from the first of active nodes) is much faster but needs a lot more storage (core) in the computer. The depth-first therefore is more suitable for adaptation to computers of any capacity (even small ones). It has also the advantage that feasible solutions are constructed from the beginning and improvements take place during

the solution,while in the breadth-first, solutions are constructed at later stages, which is not suitable for the approach adopted. For these reasons a depth-first strategy has been the approach adopted in this study, with the branching rules ECT, FCFS, SPT, described in Section 4.2. Since core is not the main problem of the solution, but time, it is necessary to consider the amount of time to be invested for any problem, as another parameter of the branch and bound method. When this predetermined time is exhausted, the solution procedures terminate, and the answer will be the best value of the objective function obtained up to that time.

Figure 5.1 Tree-search methods



Breadth-first search strategy.



Depth-first search strategy.

Feasibility function

This function is used to eliminate partial schedules known not to have completion within the set of feasible solutions. In practice, it translates the precedence sequence requirements of operations into a set of schedulable operations for every instance of the problem. It determines the next executable operation for every job, if there is any.

Lower bound function

The lower bound function assigns to each partial solution (node) a value b_e representing a lower bound to all complete active solutions-descendants from that node. The lower bound is a non-decreasing function of the level (generation of nodes) in the tree representation, whose value at the final level is equal to the actual solution. The lower bound calculation methods will be discussed at length in the following Section 5.2.

Upper bound

Upper bound b_u is the value of the objective function for the best complete solution known 'a-priori' or alternatively a large number, greater than all possible solution values, if no complete solution is known. An initial upper bound close to the value of the optimal solution reduces computations substantially, as computational experience has shown.

Dominance relation and elimination rules

Every currently active node or branch has a lower bound assigned to it which is compared with the last (best) value of the upper bound of the problem, before further branching. If $b_e \geq b_u$ then all descendants of this branch will have values $v \geq b_e$ and need not be considered further.

5.2 Lower bounds

5.2.1 One-job and one-machine based bounds

The lower bounds in principle are solutions to simple surrogate problems of the original, resulting from the relaxation of some of the constraints. For scheduling problems, the bound calculations are based on the relaxation of two types of constraints.

One-machine based bounds

Relaxing the constraint that no more than one operation can be carried out on a machine at any time, in all but one of the machines, the remaining work content of that machine determines the value of the bound. This is tantamount to allowing operations in all jobs to overlap. If this relaxation is allowed, then operations on any of the machines need not have waiting times, and the related bound is

$$b' = \max \{c_{ej} + \sum_{k \in R_j} P_{ik}\} \quad , j \in M$$

where M set of machines

R_j set of unprocessed operations in machine j

c_{ej} last completed operation in machine j

This bound, referred to as 'one-machine-based' lower bound, has been developed firstly for the flow-shop by Lomnicki (1965) and Ignall and Schrage (1965) and reformulated by others.

One job-based bounds

Relaxing the constraint that no more than one operation of a job can be executed at any time, in all jobs but one, then the remaining work-content of that job is the base for calculating the lower bound. An alternative description for the same bound is to relax capacity constraints in all

machines, and determine the schedule length from the jobs work-content only:

$$b'' = \max \{c_{ei} + \sum_{k \in R_i} P_{kj}\} \quad , \; i \in J$$

where
- J set of jobs
- R_i set of remaining unscheduled (unprocessed) operations for job i
- P_{kj} processing times
- c_{ei} completion time of last scheduled operation of job i

This bound will be referred to as job-based lower bound.

A combination of these methods defined by

$$b = \max(b', b'')$$

has been proposed by McMahon and Burton (1967) as a more efficient bound for flow-shop problems.

The basic idea of a machine based bound has been implemented in the general job-shop environment for the first time by Brooks and White (1965). Later, it has been extended to cases of different job-arrival times $R_i \geq 0$, solving the problem 'Minimise Makespan in a single machine schedule with different job arrival times ' for the bound calculation, by ordering the jobs in non-decreasing order of arrival time (Florian et al 1971, and Ashour et al 1974).

Refinements in the same basic ideas can be found in Charlton and Death (1970a and 1970b), Ashour and Parker (1971) and others which basically are suggesting the following:

the machine based bound can be improved by replacing c_{ek}, i.e. the completion time of scheduled operations in a machine k by the earliest possible start time of the schedulable operations on that machine, s_{jk} which is

$$s_{jk} = \max \{c_{ek}, \max c_{pk}\}$$

where c_{pk} is the completion time of last scheduled operation of conflict jobs (in the machine preceding k).

The lower bound calculation procedure used in the local neighbourhood search experiment has been based on the above method, and is described below.

Conflict job k is scheduled on conflict machine m* with operation completion time c_h. The conflict jobs define the set S_c, the non-conflict jobs define the set S_n and the non-conflict machines define the set S_m. The remaining work content of job i is r_i, the completion time of last scheduled operation of job i is c_{ie}, the remaining work load on machine j is r_j, the completion of last scheduled operation on machine j is c_{ej}, and the earliest start of the next operation on machine j is s_j. The job-based bound b' is:

$$b' = \max(g', g'')$$

where

$$g' = c_h + \max(r_i) \text{ for } i \in S_c$$

$$g'' = \max(c_{ie} + r_i) \text{ for } i \in S_n$$

The machine-based bound b" is:

$$b'' = \max(f', f'')$$

where the earliest loading time for the non-conflict machine j is given by $\max(c_{ej}, s_j)$ and

$$f' = \max\{\max(c_{ej} s_j) + r_i\} \text{ for } j \in S_m$$

For the conflict machine m*

$$f'' = c_h + r_{m*}$$

The overall bound is calculated as:

$$b = \max(b', b'')$$

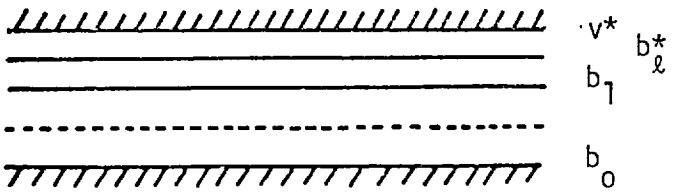
Further improvements might be possible by looking at the possible combinations of the last operations of jobs in one particular machine, which requires a step-by-step calculation of completion times of all jobs on the preceding machines, though the overall efficiency of such methods is questionable (Ashour and Hiremath, 1973). This agrees with the general principle that stronger bounds can be calculated but they may be overall less efficient than simpler ones that are dominated.

5.2.2 Complexity and limitations of bound calculations

A different approach to the calculation of lower bounds, as the one suggested by Bratley et al (1973), is based on the concept of enumeration using a NP-class algorithm. This approach does not seem to be promising, because it attempts to solve a NP-complete problem with a branch and bound method, where bounds are derived with a NP-class procedure. It is believed that only P-class algorithms can be computationally efficient and thus suitable methods for bound calculations in any problem size. It is argued here that there are limits on the values that can be calculated with such non-enumerative algorithms.

Theorem: In a scheduling optimisation problem with discrete objective function, where v^* is the optimal solution, there is a limit $b^*_\ell \leq v^*$ to the values of lower bounds beyond which stronger job and/or machine bounds are obtainable only with NP-class algorithms.

Proof



b_0: Lower bound calculated with a 'good' algorithm.

Assume that it is possible to find from lower bound b_0 the next larger value b_1 of the discrete space $\{b_0, v^*\}$ of h-elements, with a P-class (good) algorithm. Thus, in an integer formulation, which does not restrict generality, b_1 is also calculated with an overall 'good' algorithm. Repeating this procedure at most h times, the optimal value v^* would be reached with a combination of 'good' algorithms, since h is also polynomially bounded (because $h \leq v^* - b_0 << \sum_{i=1}^{n} P_i$) Thus, this limit b^*_ℓ for bounds obtainable with 'good' algorithms exists, and it is not possible to obtain better bounds unless an enumerative method is used.

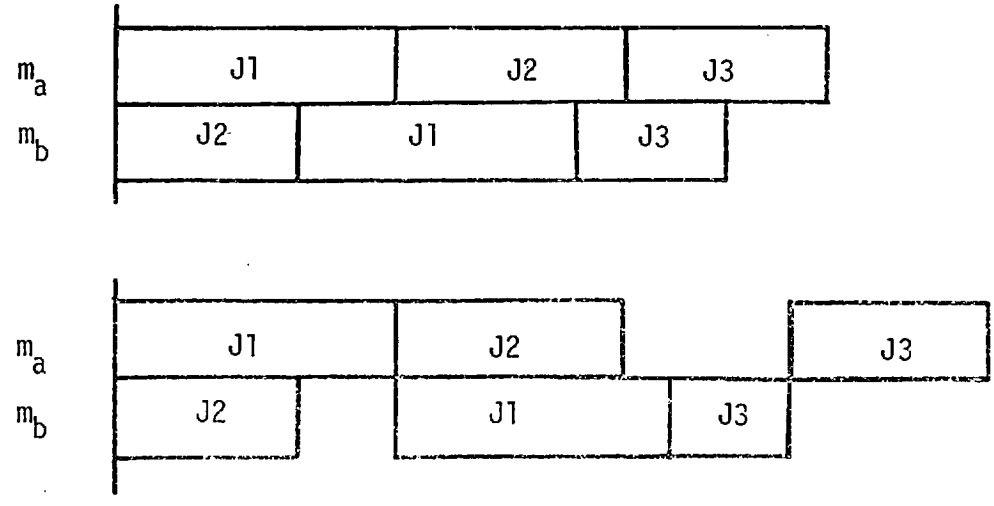
For the bounds calculated by relaxing job or machine constraints, it is to be expected that the best that can be obtained with a 'good' algorithm is in fact what the good algorithm for the largest possible problem can give. The higher order problems solvable in P-class are $n/2/F/C_{max}$ (Johnson, 1954) $n/2/G/C_{max}$ (Jackson, 1956) and $2/m/G/C_{max}$ (Szwarc, 1960).

5.2.3 A routine for calculating two-machine based lower bounds

The basic idea of this bounding procedure is to relax the machine capacity constraints in all but two machines m_a and m_b (instead of one machine as in the bounds of Lomnicki and Ignall and Schrage). For this pair of machines m_a and m_b Jackson's method, with O(nlogn) has been used, based on Johnson's rule,to calculate an optimal schedule for their remaining operations, as if the jobs involved have no operations in any other machine. The optimal solution of this surrogate problem can be used as a lower bound to the original m-machine problem. This method is based on a principle similar to the one used by Campbell et al (1970) and Townsend (1977c) whereby Johnson's rule has been used as a heuristic for the original problem, producing optimal solutions for a surrogate one.

The value of this optimal two-machine schedule cannot be lower than the load of any of the two machines considered individually. This is illustrated graphically below in Figure 5.2.

Figure 5.2 Two-machine based lower bound



This calculation is repeated for all possible pairs of the m machines. Their number is

$$\binom{m}{2} = \frac{m!}{2!(m-2)!} = \frac{(m-1)m}{2} \text{ or } O(m^2)$$

and the best (highest) value is used as lower bound. The same idea, in an alternative formulation would be to relax the job technological constraints (i.e. not to allow jobs to have two operations processed simultaneously) in all but two jobs and find the optimal schedule for this $2/m/G/C_{max}$ problem by Szwarc's method $O(m^2)$. The number of possible pairs is $O(n^2)$. This method has been considered to be poorer because, at problems with n>m, it is unlikely to have bottle-neck jobs, while it is almost certain to have bottle-neck machines. The implementation of this bounding procedure has met two basic problems. If one wants as strong a bound as possible, one should take into account that the operations for the pair of machines m_a, m_b may have different 'entry' times $R_i \geq 0$, in which case the problem $n/2/F/C'_{max}$ with $R_i \geq 0$ is NP-complete, i.e. not solvable with Johnson's rule. However it is possible to use its relaxation with $R_i = 0$ for all i, in which case C_{max} will be a lower bound to C'_{max}. The other problem arises from the fact that the machines m_a and m_b, due to decisions (conflict resolutions) taken earlier on, may become available for processing with a difference dt in time. For this case, it has been necessary to show that both Johnson's rule (Chapter 3) and Jackson's method (Appendix C) apply for determining an optimal sequence.

Having resolved these difficulties as described above, test problems have been tried and the result was that the simpler bounding method was overall at least as good and probably better than the two-machine based one. This result verifies what other researchers have found, namely that stronger bounds are not necessarily better, because the increased computational costs involved cancel all possible advantages.

5.2.4 Fictitious bounds and estimates of the optimal solution

An alternative to trying to improve bound calculations is the idea of using 'fictitious bounds' (Bazaraa and Elshafei, 1977). Given a lower bound b_e and a good upper bound b_u, one can put forward the hypothesis that the optimal solution value is $v = b_e + (b_u - b_e)a$, where $0<a<1$. The tree is then searched for constructing a feasible solution $v \geq v^*$ (v^* is the real optimal), discarding all branches with lower bound values greater than v. If such a solution is found, then the upper bound value is updated to $b_u = v$, a new value is assigned to the estimate of the optimal v and the process is repeated, till no feasible solution of value v is found. In this way the interval where the real optimal lies is narrowed until eventually the real optimal will be located and constructed. This method obviously requires a substantial amount of search, needed for the successive estimates of the optimal (for the repetitions of the tree search with different bounds).

The shortest and most efficient tree search with a given method of lower bound calculation, is one where the value of the optimal solution v^* is known in advance, and the branch and bound method is used only in order to construct an optimal schedule. In such a case the value of the optimal solution would be used as the upper bound for the dominance relation ($b_u = v^*$). As soon as the lower bound of a partial solution becomes greater than v^*, the associated branch is rendered inactive (pruned). This point will be discussed again, in the light of the statistical results presented in Chapter 6.

5.3 Design of the experiment

The aim of this experiment has been to compare simple enumerative heuristics with a depth-first branch and bound method, in a wide range of problem structures, in order to find probabilistically the bounds of performance and determine the most appropriate solution strategy in each case. Another objective has been to relate the complexity (size of solution space A) of the problem to its characteristics and dimensions, namely to the number of jobs n, to the number of machines m and to the variability (coefficient of variation V) of the processing times. The main criticism for the branch and bound methods in general has been that they are unpredictable and expensive. Thus, one of the objectives of this study has been to investigate this unpredictability and to try and quantify it, if possible (predict behaviour by modelling solution improvements).

The typical branch and bound method makes an exhaustive search of the tree and it is terminated when an optimal solution is found. This occurs when

(i) the upper bound becomes equal to the overall lower bound of the solution (ie equal to the least lower bound of the active nodes).

(ii) the complete tree has been searched.

This method guarantees that an optimal will be found, but in larger problems, the amount of computational resources required may be enormous and therefore prohibitive. Besides, it is questionable whether the optimal solution is really needed, given that the data representing the processing times are only estimates of the actual service times. The use of assumptions about machine availability and breakdowns, labour availability etc. is another reason for not insisting on optimality. The above reasoning leads to the acceptance

that a solution with guaranteed distance from the optimal is a far more efficient usage of the computational resources and far more realistic as a task for real life job-shop scheduling problems. One could then redefine the problem as 'find a solution which guarantees that the value of the optimal solution is within a distance from the best known solution defined by a bracket for the optimal solution (BOS) of e%, regardless of the time required, which might result in a very early termination (before the expendable time is exhausted) or in a prolonged search (beyond desired expense). In terms of the BOS specified for the optimal solution, as defined in the previous chapter, it is reasonable to decide in advance the amount of computer time to be invested in a search. The problem is thus reformulated as 'find the best possible BOS (bracket for the optimal solution) within the amount TL of computer resources (time)'.

More general stopping rules might take into account not only a desired BOS of e% and a TL but also a cost related to the inaccuracy of the solution, the proportion of the tree searched as an indicator of the probability of finding better solutions, the rate of solution improvements etc., to be discussed later in Section 5.5. The decision rules to be evaluated and compared are the enumerative heuristics based on the branching rules ECT, FCFS, SPT in a LNS. Their performance will be measured by the value of BOS obtained in a given TL.

Table 5.1 on the following page, shows the different problem structures to be investigated, the sample sizes of the experiment and the time limit allowed TL (CPU time in CDC 7600 seconds).

The LNS is by nature a dynamic process in which, whenever a solution better than the current upper bound is constructed, it is adopted and used for the search of the remaining neighbourhoods of the tree.

The efficiency of the LNS as a heuristic depends on the speed of these improvements and therefore some method of evaluating this speed or rate of improvements is needed. The values of the upper bound can be seen as values of a discrete function of time or of the number of iterations. It is to be expected that the number of iterations and CPU time are related and thus either of them can be used as the independent variable of the above mentioned discrete function.

Table 5.1 Sample sizes and computational budget for LNS

Problem Dimensions		Sample size					Time limit (seconds)
Jobs	Machines	Distribution of processing times					
		E_1	UR	E_4	E_9	E_{36}	TL
4	3	10	10	10	10	10	1
6	3	10	10	10	10	10	5
8	4	10	10	10	10	10	15
10	4	20	20	20	20	20	5
20	5	20	20	20	20	20	10
35	5	5	5	5	5	5	60

The efficiency of the heuristics will be judged also on the proportion of the tree that can be searched within the specified computational resources (speed of search). The heuristic that searches the largest part of the tree is expected to give the best solution value. The calculation of this proportion requires an estimate of the total tree size T_d and of the number of branches that remain unsearched, R_d, where d is the depth of the tree. This can be achieved by a statistical method, or with a simple recursive calculation based on observation of the solution instance (both described in Appendix C)

t_i number of branches of active node at level i

r_i number of unsearched branches of active node at level i

T_i total number of branches at level i ($T_0 = 1$)

R_i total number of unsearched branches at level i ($R_o = 0$)

The recursive formulae are:

$$R_{i+1} = R_i + T_i r_i$$

$$T_{i+1} = T_i t_i$$

and $$P = 1 - (R_d/T_d)$$

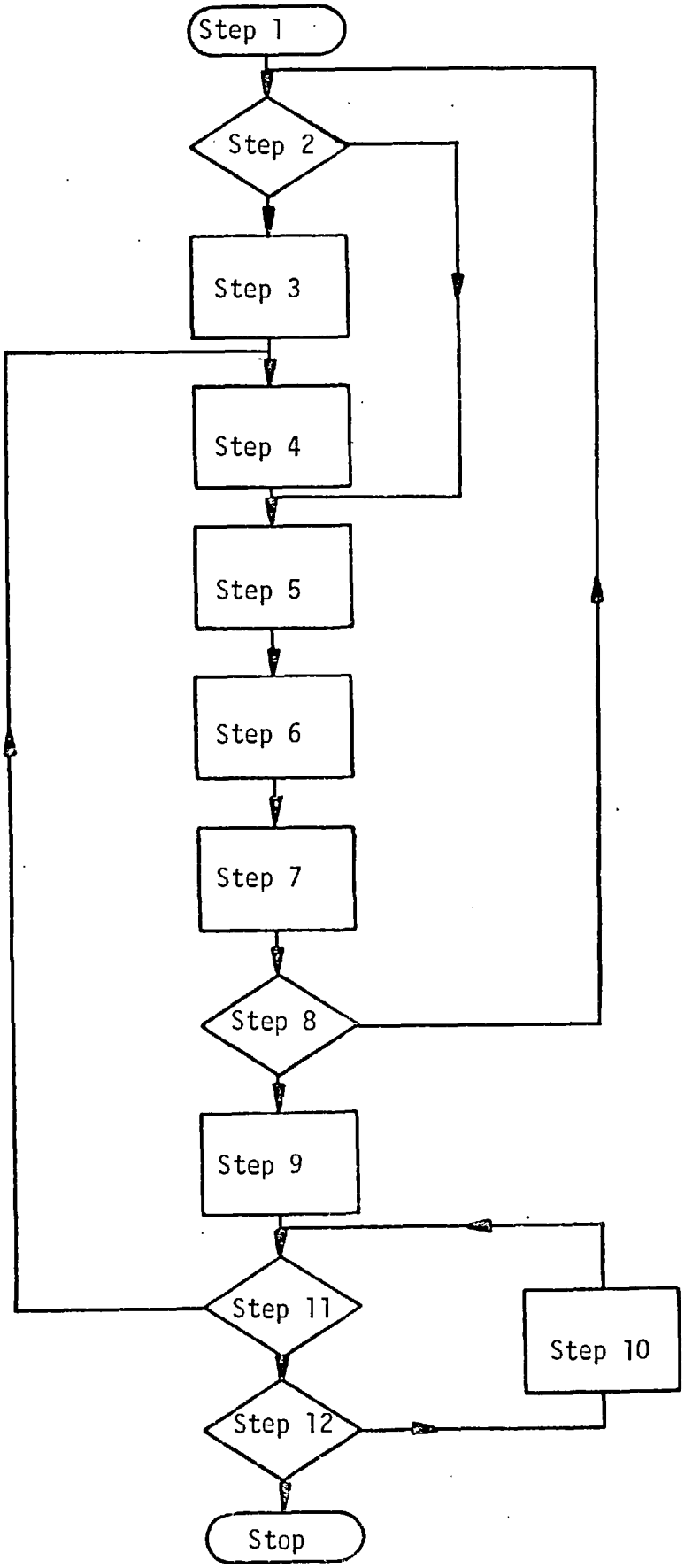
This calculation is needed not only for estimating the complexity of the problem and for the comparative evaluation of heuristics but also for stopping rules or decisions (to be discussed in Section 5.5).

Description of the model

The algorithm described in Chapter 4 generates only one active schedule. In order to be able to generate all the active schedules, extensive modifications are required for record keeping. At every conflict (node), information has to be stored for the alternative decisions possible, to allow back-tracking and re-branching. Modifications are also required for the dominance relations and elimination rules ('pruning' or 'fathoming' of the tree), and other record keeping needs. Any lower bound from those described in the previous section can be used, without effect in the structure of the method. The flow-chart of the algorithm used is given in Figure 5.3.

Elimination (fathoming) of a branch takes place according to the dominance relation, whenever a particular branch is not going to produce solutions better than the best known solution (value of upper bound). This is checked at a number of points. At Step 4, as soon as a job is selected for the conflict resolution, the lower bound (b_e) of the corresponding branch is compared with the current upper bound (b_u). If $b_e \geq b_u$, then further search in the descendants of that branch is pointless, and elimination takes place.

Figure 5.3 Flow-chart for job-shop scheduling with a local neighbourhood search method.



At Step 7, where the completion time C_p of the current partial schedule is determined, if $C_p \geq b_u$, then the complete feasible schedules based on that current partial schedule cannot have C_{max} lower than b_u, and again elimination is necessary.

In the two cases described above and also whenever a complete solution is constructed the current active conflict node is searched to see whether it remains active, in which case the remaining branches are investigated (at step 4), or whether it has become inactive (all options exhausted) in which case back-tracking to the immediately above node level takes place (Step 10). More details of this model and the FORTRAN IV codes of the routines used are given in Spachis (1978b).

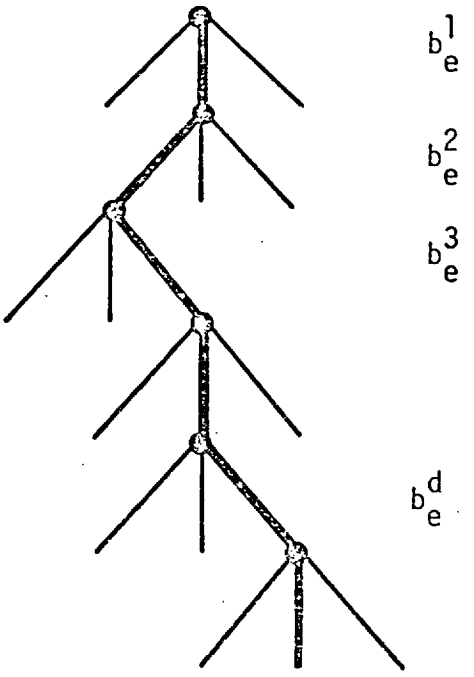
5.4 Computational experience

5.4.1 Lower bounds

The two bounding methods described in Section 5.2 were tried on test problems, and the conclusion was that the simpler method, based on one-job and one-machine was, in terms of overall efficiency and simplicity of implementation, as good and probably better than the two-machine one. Thus the former has been adopted for the LNS experiment.

A direct evaluation of the one-machine and one-job based lower bound can be carried out by comparing the smallest lower bound value b^*_e calculated at the first conflict (first node of the tree) with the optimal solution. The value of the lower bound can be expressed as a non-decreasing discrete function of the depth of the tree, across any traversal of the graph representing the complete tree (Figure 5.4), because $b^1{}_e \leq b^2{}_e \leq b^3{}_e \leq \ldots\ldots \leq b^d{}_e$, where d is the depth of the tree.

Figure 5.4 Lower bounds across a tree traversal



The least of these lower bound values across a traversal is

$$b^1_e = \min(b^k_e) \quad \text{for} \quad k=1,.....d$$

and the least of the lower bound values at the first node is b^*_e. This is an overall lower bound to the solutions of the problem. A feasible solution with value $v=b^*_e$ is apparently an optimal solution. The question that arises here is, how often is this least lower bound value realizable as a feasible solution, i.e. how often is it an optimal solution. From the experiment described in the preceding section, the following results have been obtained, summarised in Table 5.2 below (sample sizes as in Table 5.1).

Table 5.2 Feasibility of minimum lower bounds b^*_e (Frequency of F,NF,U)

Job-shop size		TL	Distribution of processing times														
			E_1			UR			E_4			E_9			E_{36}		
Jobs	Machines	secs	F	NF	U	F	NF	U	F	NF	U	F	NF	U	F	NF	U
4	3	1	5	5	0	3	7	0	3	7	0	4	6	0	1	9	0
6	3	5	7	3	0	8	2	0	7	3	0	5	5	0	6	4	0
8	4	15	5	1	4	9	1	0	5	2	3	3	4	3	4	4	2
10	4	5	10	0	10	11	0	9	10	0	10	12	1	7	14	0	6
20	5	10	6	0	14	7	0	13	10	0	10	10	0	10	2	0	18
35	5	60	2	0	3	2	0	3	4	0	1	1	0	4	0	0	5

Note F: there is a feasible solution with value b^*_e (optimal)

NF: there is no feasible solution with value b^*_e

U: it is not known whether there is a feasible solution with value b^*_e or not.

One can see from Table 5.2 that for smaller problems a complete search of all the neighbourhoods of the tree has been possible within TL, while for increasing problem size the proportion of incomplete search cases increases. From the test problems where the search has been complete

(i.e. problems of 4 jobs 3 machines and 6 jobs 3 machines) it is clear that the number of cases, where b^*_e is a feasible (optimal) solution, increases with size. This could be attributed to the method of lower bound calculation, based on the summation of processing times, without taking into account machine idle times, which may produce less tight bounds for smaller problems, as already discussed in Section 4.3.

The variance of the processing times as can be deduced from Table 5.2 is also significant for the quality of b^*_e. The pattern that emerges for small problems (Table 5.3) is that for increasing variance, the proportion of cases where $b^*_e=v^*$ increases. This can be justified by the nature of the data structure. In high variance data, there are jobs and/or machines whose work content (total processing times) is dominant or exceeds the others by a substantial amount. This means that they are crucial in determining C_{max}. At the same time they are crucial for the lower bound calculations. Thus, there are feasible solutions with value equal to the least of lower bounds. The minimum makespan for problems with low variance data depends heavily on the sequence induced delays which are not taken into account in the lower bound calculations, thus it is less likely that b^*_e will be the value of a feasible solution.

The same reasoning applies for larger problems, where a time limit has been imposed on the search, and for a number of problems it has not been possible either to obtain a feasible solution equal to b^*_e or to complete the search. It is to be expected that the dispersion of solution values is lower in problems with low variance of processing times, i.e. that the frequency of each class of solution values is higher than in problems with larger variance. This implies that whenever b^*_e is realizable as a feasible solution, in low variance problems, such a solution, although rare,

is likely to be constructed earlier in the search than in high variance problems.

The information obtained on the quality of the bounds is significant not only from the point of efficiency of the search but also for purposes of stopping decisions. Early in the search a good solution v is constructed which is greater than the least of lower bounds ($v>b^*_e$), and the search continues, until either it is completed or a better solution $v'=b^*_e$ is constructed. It would be useful to have some information on the probability of having an optimal $v^*>b^*_e$, with the view of using it for decisions on whether to continue the tree search or not.

This type of information can be presented as the frequency of feasibility of b^*_e, as in Table 5.3 below, where the impact of size and variance on the quality of the lower bounds is summarised. It is worth noting that out of 375 test problems investigated with limited computer resources, the b^*_e has been the value of a feasible (optimal) solution in about half of them.

Table 5.3 Frequency of feasibility of b^*_e

Number of machines	Distribution of processing times				
	E_1	UR	E_4	E_9	E_{36}
3	12/20	11/20	10/20	9/20	7/20
4	15/30	20/30	15/30	15/30	18/30
5	8/25	9/25	14/25	11/25	2/25
TOTAL	35/75	40/75	39/75	35/75	27/75

5.4.2 Number of iterations and CPU time

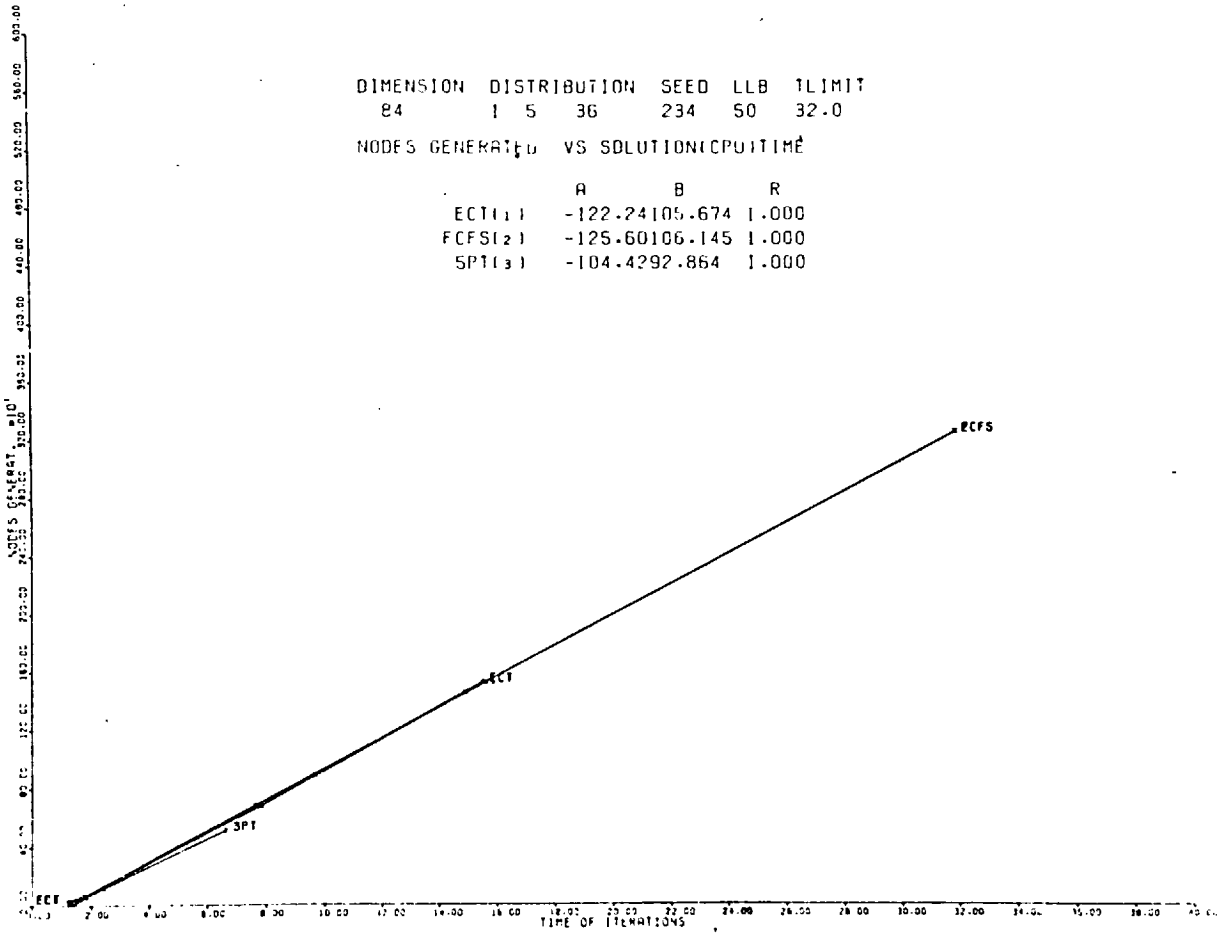
The number of nodes generated during the tree search can be described in terms of a monotonically increasing function of time. For all practical purposes, the function can be assumed to be

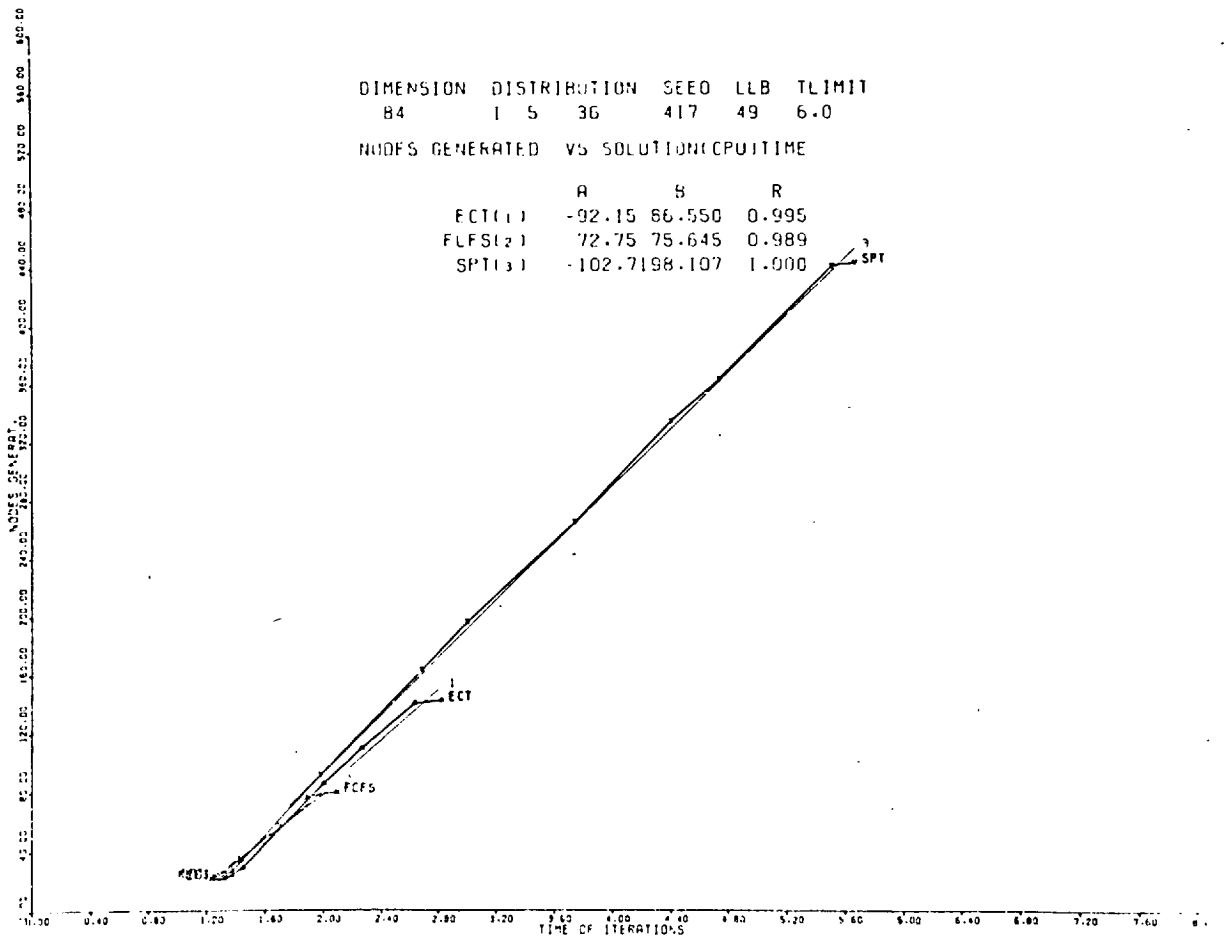
continuous given the very large number of nodes involved, and the very small time interval between two successive nodes. In order to establish the relationship between the number of nodes generated and the CPU time, results from a range of problems have been investigated. The problems investigated were of 8 jobs 4 machines, (E_{36}) 10 jobs 4 machines (E_1,E_9,E_{36}) 20 jobs 5 machines (E_9) 35 jobs 5 machines (E_9,E_{36}). The results have been plotted with the aid of computer graphics and a sample of these plots is given in Figure 5.5 on the following page. In these graphs, the meaning of the names and figures printed in the heading is given below.

Name	Numerical example	Interpretation
DIMENSION	84	Job-shop of 8 jobs 4 machines
DISTRIBUTION	1	Erlang type of distribution of processing times
	5	expected value (average) of processing times
	36	k parameter of Erlang distribution
SEED	234	Seed for random number generator, code for the data used in test problem
LLB	50	minimum of lower bounds b^*_e
TLIMIT	32	Time limit TL imposed on the LNS
A		regression constant α (intercept) for linear regression of nodes against time
B		regression coefficient β (slope)
R		correlation coefficient r

Taking into account all information up to the last solution improvement available, the relation between nodes and time of the example problems of Figure 5.5 is clearly linear, with a correlation coefficient r equal to 1. In fact, only four of the 105 solutions involved had r lower than .98. Thus the effect of backtracking in terms of time for subsequent nodes generation is insignificant.

Figure 5.5. Number of iterations as a function of CPU time





The particular problem set of 35 jobs 5 machines, E_9, was tried also including the number of nodes near the end of the time allowed, which did not represent a solution improvement. The correlation coefficient was again practically equal to one and the difference in the regression coefficient β was insignificant.

The conclusion is that, provided there is a reasonable number of solutions to use, for all practical purposes, the number of nodes is linearly related to the CPU time elapsed, and thus can be treated as equivalent.

5.4.3 Problem complexity

The partitioning of the active solution space A by the branch and bound method already described, defines a tree with d generations (levels) of nodes and size {A}. As already discussed these quantities are bounded as follows:

$$d < nm$$

$$\{A\} < (n!)^m$$

The size of the set A is a measure of the overall complexity of the problem and the depth of the tree is significant for the amount of computer storage required in branch and bound depth-first solutions. Information on these quantities has been collected for a number of test problems with 10 jobs, 4 machines, processing times from E_1,UR,E_4,E_9,E_{36} distributions each solved with heuristics ECT, FCFS, SPT and RANDOM with the objective of finding whether there is any relation between d, {A} and the above mentioned parameters of the problem.

Analysis of results for each of these problems showed that the depth d is fairly stable at different traversals of the corresponding tree, varying in the typical case from 28 to 32. The range of values of d obtained for a problem is not identical for all heuristics.

Some variations in the range of d have been observed also for problems with the same distribution of processing times as well as for problems with differing V. The conclusion is however that d is fairly stable. No correlation appears to exist between d and V, and there is no evidence that partitioning A by different heuristics results in significantly different depth patterns.

Similar conclusions can be drawn for the estimates of the size of the set A. For each solution, the estimates of {A} are fairly similar at the different solution instances, typically of the order of E14. The estimates resulting from the use of different heuristics in each problem are also very close to each other, which validates the method used for estimating the tree size. Besides there is no evidence that different data structures (i.e. different V) result in different tree sizes. The following Table 5.4 presents the typical values of d and A for the different problem sizes.

Table 5.4 Size of set of active schedules

Jobs	Machines	d	{A} (E)	$(n/m)^d$ (E)
4	3	6- 8	3	1-2
6	3	11-14	5	4-5
8	4	18-22	8-9	6-7
10	4	26-32	13-15	11-13
20	5	77-83	40-48	46-49
35	5	150-160	92-108	>100

If the number of branches per node is taken to be the same all over the tree (which does not happen in practice), then the ratio $n/\bar{m}$ is the most plausible value, given that the routing is random and initially the n jobs are divided equally to the m machines for processing. For a tree with d generations of nodes, the number of branches at the last level will be $(n/m)^d$ (see Appendix C). For

the experimental values obtained for d, the estimates $(n/m)^d$ are given in Table 5.4. This model is certainly crude but as the figures in Table 5.4 suggest, it gives a reasonable a-priori estimate for the order of the size of the set A.

5.4.4 Speed of tree search

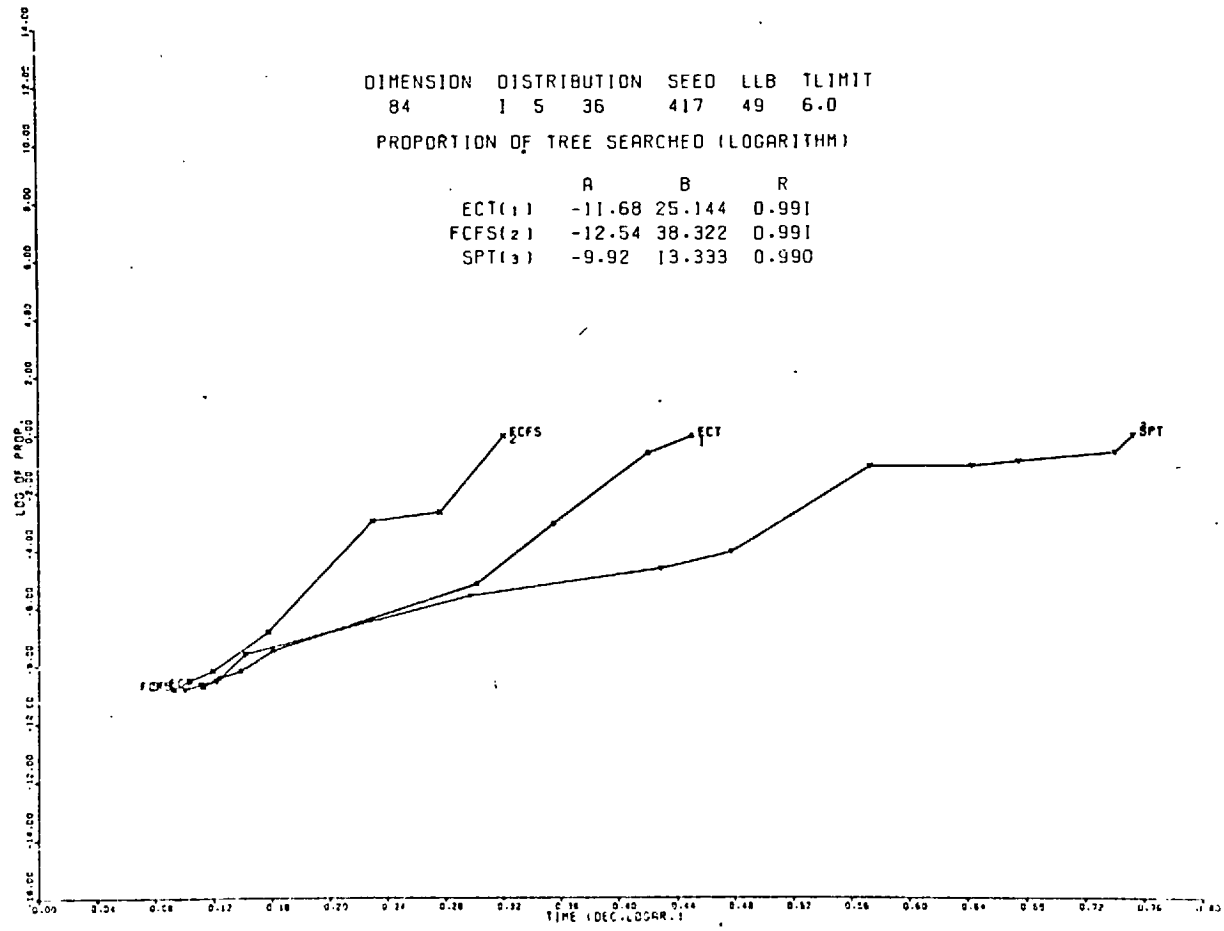
The proportion of the tree that has been searched up to a point in time t is an increasing function of time P(t), whose values can be calculated with the method already described in Section 5.3.

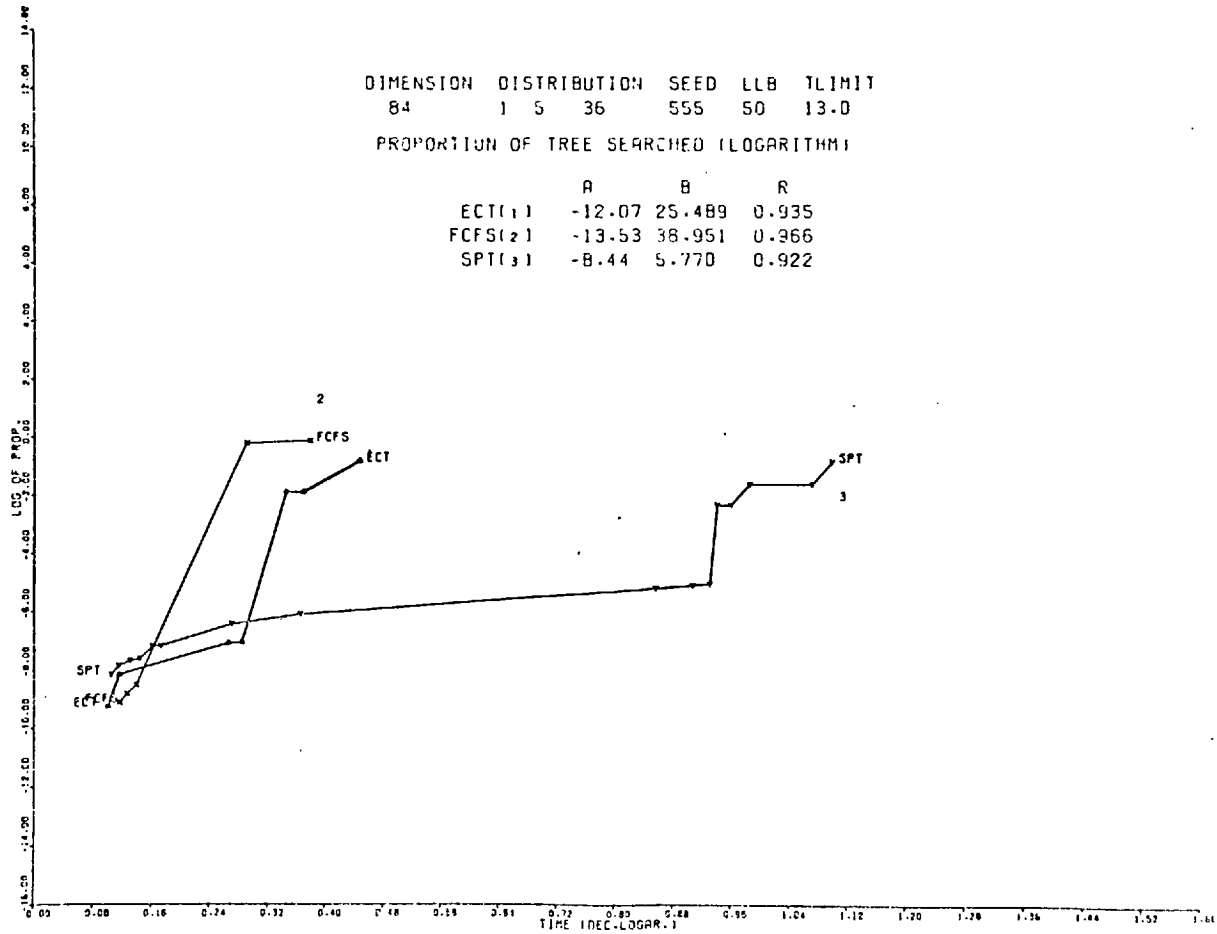
A sample of problems with 8 jobs and 4 machines has been used for detailed study of the function P(t). The first feasible solution constructed corresponds to a proportion of the tree of the order of E-10. The value of the proportion searched increases rapidly as a result of elimination of branches at various levels in the tree. The enormous range of values which P(t) takes during a problem solution does not allow any meaningful diagram to be constructed, unless logarithmic scales are used.

The task of plotting a large number of diagrams with different data has been carried out by computer plotting facilities. Some representative results are given in Figure 5.6, where the points of the function P(t) corresponding to solution improvements are linked by straight lines. If these points are used for a linear regression of the form $\log P = \alpha+\beta \log t$, the values of the correlation coefficient are quite high. The lowest value is $r=.7$ and $r>.9$ in 25 out of the 30 test cases. This result though should not be interpreted as an exactly linear relation between logP and logt, but rather as an approximate description of the trend.

In the expression $P = 10^{\alpha}t^{\beta}$ for the solution of a problem with one of the heuristics ECT, FCFS, SPT the value of β indicates the

Figure 5.6 Proportion of tree searched as a function of CPU time





speed of the tree search with that particular heuristic. The values of β could be used as a criterion of performance of the heuristic, where higher β means faster tree search and thus better performance of the heuristic. Alternatively, the heuristics can be ranked by simple inspection of the graphs shown in Figure 5.6, where the best heuristic is the one which searches a given proportion of the tree in the shortest time.

The information on the proportion of tree searched over time is useful also for stopping decisions in LNS, allowing an evaluation of the potential outcome of an extended search. If at time t, near the time limit, the proportion searched is very small and the extrapolation of the trend shows that a complete search would require a lot of extra time, the LNS is terminated. If on the other hand P approaches 1.0, an extended search might be considered.

5.4.5 Worst case and probabilistic behaviour of heuristics in LNS with limited computational resources

The BOS performance of heuristics is described by an average ranking and by some characteristics (parameters) of their distribution function. The ranking of the heuristics for each problem has been based on the values of C_{max} obtained, i.e. the lowest value gives rank 1 to the corresponding heuristics etc. In many cases, especially in smaller problems, all the heuristics achieve the same value, the optimal. In these cases, the ranking is based on the speed with which the value is obtained, measured either by CPU time or by the number of iterations. The average rankings given in Table 5.5 have been calculated either directly from experimental results or from diagrams as the ones presented in Figure 5.9 of Section 5.5.

Table 5.5 Average Ranking for Makespan

		n=4 m=3	n=6 m=3	n=8 m=4	n=10 m=4	n=20 m=5	n=35 m=5
E_1	ECT	1.90	1.65	1.70	1.80	1.85	1.40
	FCFS	1.60	1.90	1.40	1.60	1.35	1.60
	SPT	2.50	2.45	2.90	2.60	2.80	3.00
UR	ECT	1.60	1.45	1.30	1.55	1.65	1.80
	FCFS	1.80	2.10	2.00	1.60	1.45	1.20
	SPT	2.60	2.45	2.70	2.85	2.90	3.00
E_4	ECT	2.05	1.45	1.70	1.45	1.50	1.40
	FCFS	1.35	2.20	1.80	1.85	1.60	1.60
	SPT	2.60	2.35	2.50	2.70	2.90	3.00
E_9	ECT	1.45	1.70	1.70	1.50	1.50	1.20
	FCFS	1.95	2.00	1.80	1.75	1.50	1.80
	SPT	2.60	2.30	2.50	2.75	3.00	3.00
E_{36}	ECT	1.70	1.50	1.70	1.75	1.35	1.80
	FCFS	1.40	1.60	1.50	1.40	1.65	1.20
	SPT	2.90	2.90	2.80	2.85	3.00	3.00

Note

n is the mumber of jobs

m is the number of machines

E_1, UR, E_4, E_9, E_{36} are distributions of the processing times

ECT, FCFS, SPT are the heuristics used

The main conclusions drawn from the results of Table 5.5 are that SPT is clearly the poorest , in all circumstances. No valid comparisons can be carried out for a given value of V across the various

dimensions, because optimality is not always achieved. Although some consistency of relative performance of the heuristics is to be expected (see also Section 5.5), the arbitrary nature of the time limit used may affect the results. It is possible that different rankings would be obtained at different time limits, although in the typical case, the heuristics have the same rank over time. For a given problem size, at low variance (E_{36}) FCFS has better average ranking than ECT, as well as at high variance (E_1). For intermediate values of V (UR,E_4,E_9) ECT has the edge.

The parameters used for describing the probabilistic behaviour of the heuristics are the worst case bracket for optimal C_{max} and average BOS (Tables 5.6 and 5.7).

Table 5.6 Worst case of BOS of LNS at TL (%)

ECT

n	m	E_1	UR	E_4	E_9	E_{36}
4	3	0	0	0	0	0
6	3	0	0	0	0	0
8	4	11	0	10	14	13
10	4	20	14	12	15	12
20	5	16	15	10	10	11
35	5	13	12	6	5	5

FCFS

4	3	0	0	0	0	0
6	3	0	0	0	0	0
8	4	11	0	10	14	13
10	4	26	11	17	18	11
20	5	17	10	13	10	15
35	5	9	7	5	5	6

Table 5.6 (continued)

SPT

n	m	E_1	UR	E_4	E_9	E_{36}
4	3	0	0	0	0	0
6	3	0	0	0	0	0
8	4	11	0	10	14	14
10	4	25	14	20	21	24
20	5	19	26	20	19	22
35	5	18	21	22	18	20

Observation of Table 5.6 shows that at smaller problems, the worst case bracket is always zero, because optimality is achieved with all heuristics. For larger problems ECT and FCFS have roughly the same values of worst case performance, which is always better than SPT. Higher values are seen in high variance problems.
It is very encouraging to see that for larger problems, the worst cases have smaller brackets than the medium-size problems. It is thought that this is not due to the time limits used but to the size of the problems.

Table 5.7 Average BOS of LNS at TL (%)

ECT

n	m	E_1	UR	E_4	E_9	E_{36}
4	3	0	0	0	0	0
6	3	0	0	0	0	0
8	4	2.4	0	2.4	2.4	3.7
10	4	3.9	3.0	4.4	4.0	3.0
20	5	7.7	5.3	3.0	3.2	5.7
35	5	5.2	6.1	2.0	2.7	3.1

FCFS

4	3	0	0	0	0	0
6	3	0	0	0	0	0
8	4	2.9	0.6	2.4	2.4	3.7
10	4	5.5	3.2	5.3	3.7	2.7
20	5	3.6	4.1	4.3	3.7	7.0
35	5	5.4	3.4	2.2	3.6	3.0

Table 5.7 (continued)

SPT

n	m	E_1	UR	E_4	E_9	E_{36}
4	3	0	0	0	0	0
6	3	0	0	0	0	0
8	4	3.1	0	2.4	2.4	3.8
10	4	6.4	6.1	7.9	7.5	8.7
20	5	12.1	12.9	12.0	12.5	16.4
35	5	11.4	16.4	14.4	15.0	17.4

The results in terms of average BOS are particularly encouraging (Table 5.7). The practical implications are that given a problem of size n-jobs m-machines and the variance of the processing times, one can refer to these results, select the heuristic to be used and have an indication of its performance from the average bracket as well as of some upper bound for the amount of the limited computer resources specified.

5.5 Solution improvement models and stopping rules

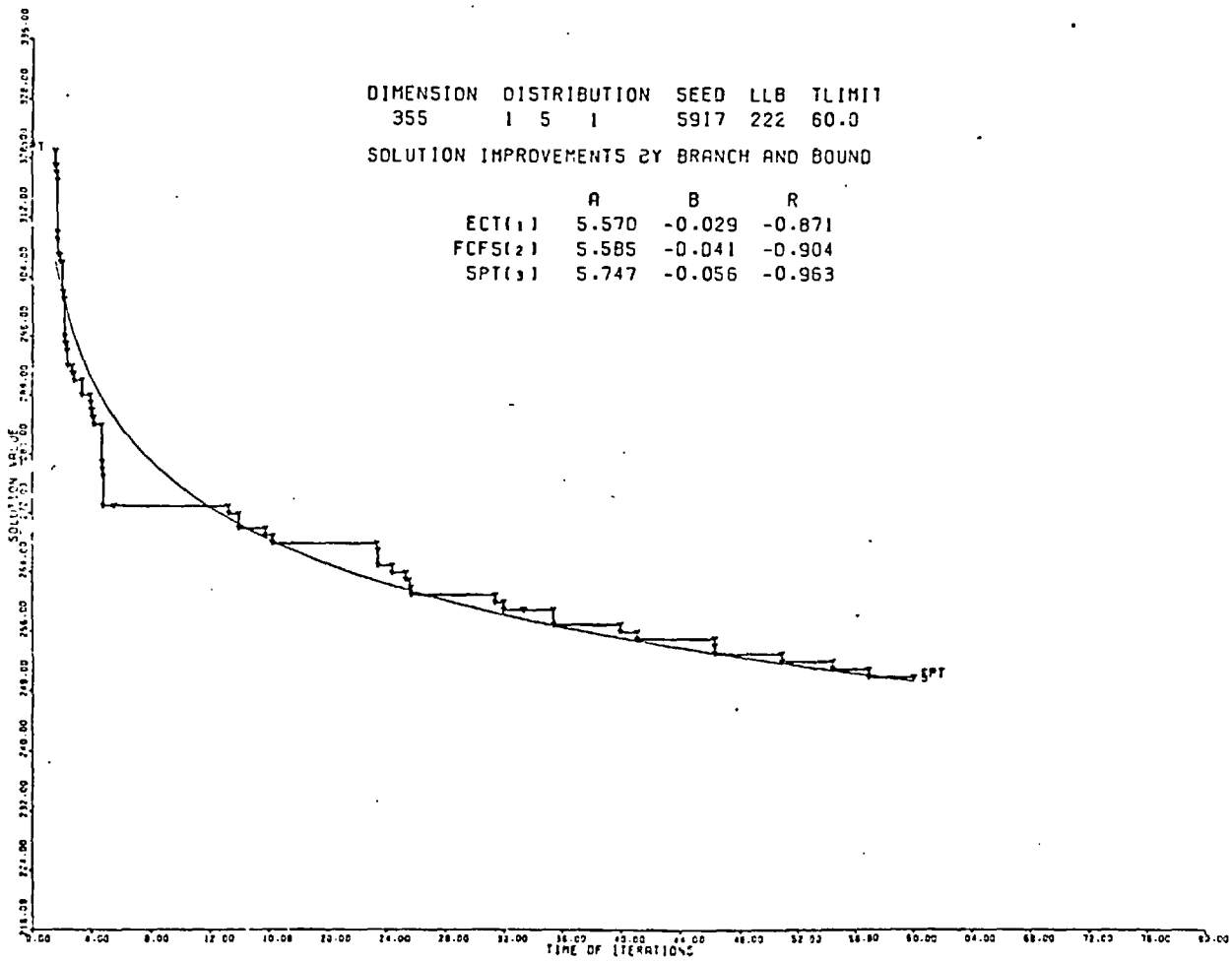
5.5.1 Modelling the solution improvements process in LNS

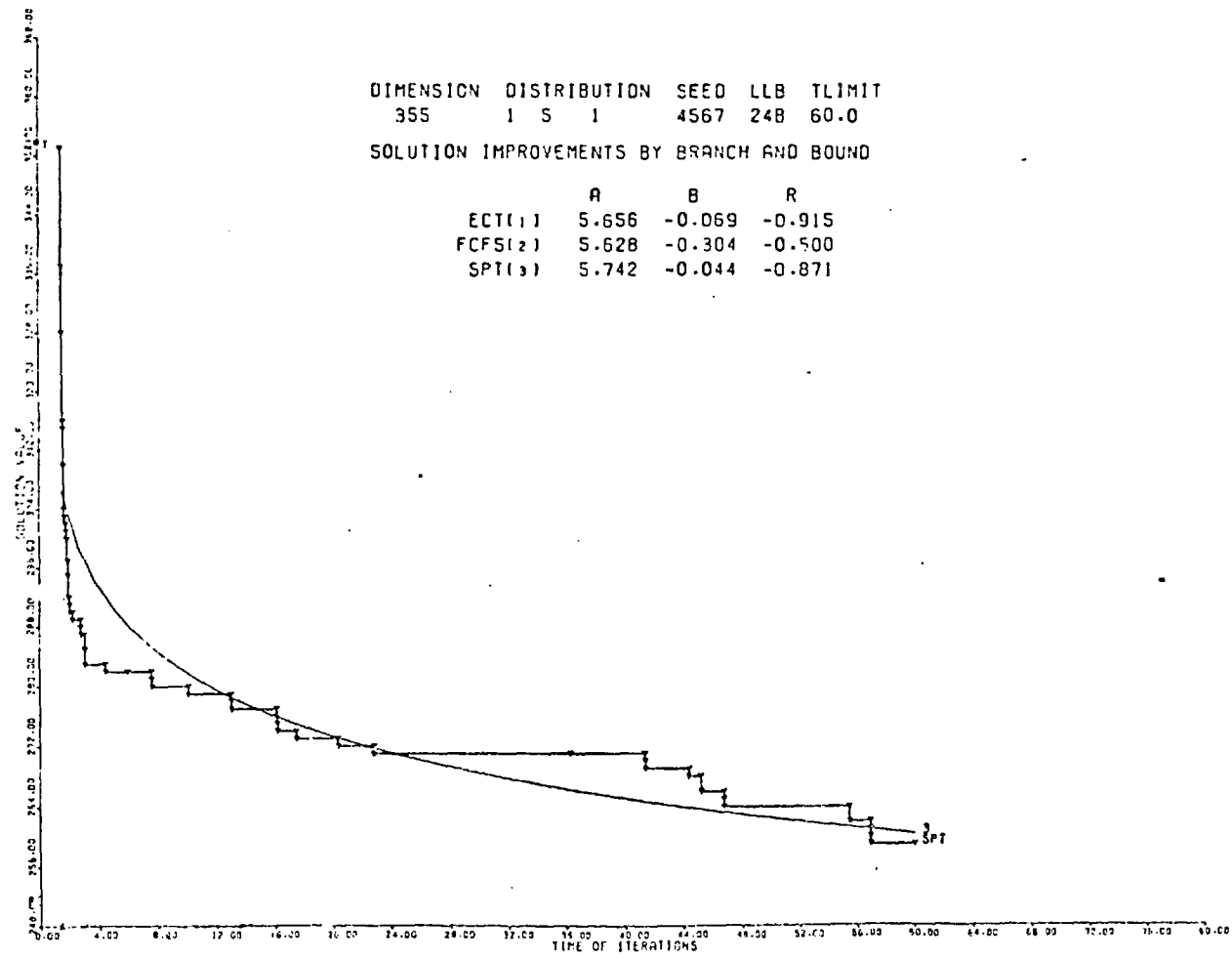
The local neighbourhood search (LNS) produces solutions whose C_{max} value decreases with increasing number of iterations. It has already been established in Section 5.4.2 that for all practical purposes, CPU time and number of iterations are equivalent, being linearly related. Thus the LNS process can be represented as a decreasing step function of time f(t), where the solution values (or the bracket for the optimal solution - BOS) are discrete and the variate t can be treated as continuous. This function f(t) has been plotted with the independent variable of CPU time on the horizontal axis and the solution value (or BOS) on the vertical axis. A typical solution improvement pattern can be seen in Figure 5.7 below, where the continuous line is an approximation of the step function.

The 'staircase' representation of Figure 5.7 is particularly useful for the comparisons and ranking of heuristics, carried out as described in Section 5.4. It is suitable also for demonstrating the effects of using fictitious lower bounds, i.e. when instead of the calculated value b_e, the bound $b_f=b_e+\Delta b$ is used. Thus the search of the tree is accelerated and the solution terminates when $v=b_e^* + \Delta b$ or when a solution $v'>b_e^* + \Delta b$ is constructed and the complete tree is searched. In such a case there is a loss in accuracy $\varepsilon \leq \Delta b$ and the savings in time are Δt. The two processes for a given problem can be represented as in Figure 5.8 on the following page.

The pattern of the LNS solution improvements is very much the same in all problems investigated. As can be seen from the typical chart of Figure 5.7, there is a rapid rate of improvements in the beginning of the search, slowing down later on with increasing

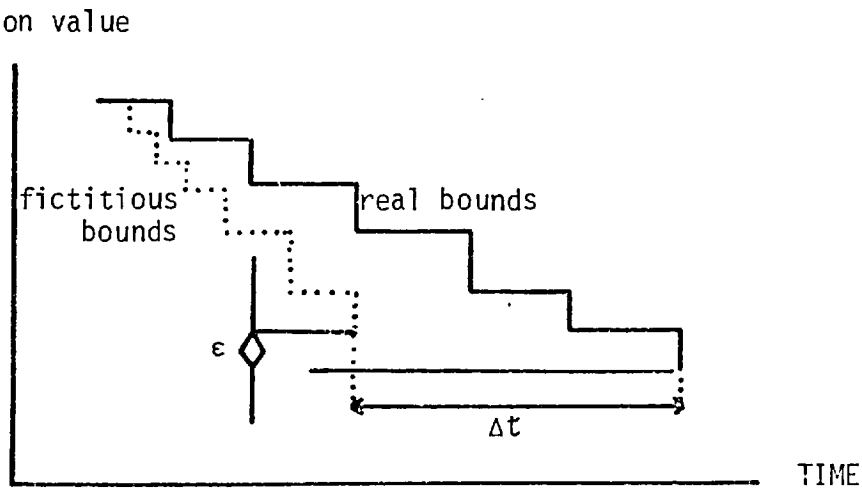
Figure 5.7 Typical solution improvements patterns





time. The fact that there are increasingly long intervals without any improvement is due to the nature of the search. A good solution is found within a local neighbourhood in which most of the other solutions have inferior values. A long search proves that there is no better solution in that neighbourhood. By backtracking higher up in the tree, and reversing some critical initial decisions (conflict resolutions), eventually another neighbourhood is reached and the search there produces a series of new improvements. The typical case is one where the intervals without any improvement become longer and the improvements obtained, if any, are smaller.

Figure 5.8 Effects of fictitious bounds



During this search, it is possible that a solution is constructed with value $v=b_e^*$ (b_e^* is the least of lower bounds for the problem). In this case, the solution obtained is an optimal and no further search is required. If this does not happen, and the search is allowed to continue, then a solution $v'>b_e^*$ is constructed, after which no further improvement is obtained with an exhaustive tree search. This solution v' is then an optimal. From the computational experience of this study, v' is usually constructed fairly early in the search and then a large amount of time is spent trying to prove that v' is an optimal. Which of the two cases takes place in a particular problem depends on the quality of the lower bounds

calculations as already evaluated in Section 5.4.1 , where a pattern of the frequency of $v=b_e^*$ has been discussed.

In large problems optimality is achieved practically only when $v=b_e^*$, since exhaustive enumeration for the case of $v'>b_e^*$ is not possible within the limited computational resources TL. It is usual that, even when a solution with $v=b_e^*$ exists, it is not found within TL. Thus, one can assume that at least one more solution improvement is possible (i.e. one solution with value less than the upper bound), if the search is allowed to extend beyond the time limit, which would guarantee optimality of solution.

It is argued that this event could be predictable, to a certain extent. The method by which it could be predicted is one in which a model is constructed, describing the dynamic behaviour of the LNS with a given heuristic. The discrete empirical function f(t) for $t_o<t<TL$ can be approximated with a continuous function F(t'), for $t_o<t'<TL^*$, where $TL^*>TL$ is an extended time limit, and t_o is the time of completion of the first feasible solution.

The approximation can be as good as it is desired, provided polynomial models of higher order are used. It is not very helpful though to use higher order polynomials, because the number of parameters (coefficients) estimated from the 'fitting' will be high, and no meaningfull comparisons can be made across different problems. Ideally one would prefer a one-parameter fitting, describing only the shape (rate of improvements). This is not possible because the starting points of the step-functions of various problems, although bounded (see Section 4.3) can vary considerably. The next simplest fitting will obviously include a second parameter for the location of the approximate curve (starting point).

The fitting that would be theoretically more correct would consider every point of the horizontal lines of the step-function, representing the value of the upper bound at every iteration of the solution. This is impractical, because of the number of iterations involved. Therefore, it is necessary to use only representative instances of this step-function. A number of approximations is possible:

1. The points of the solution improvements only are approximated by a curve $y = e^{\alpha+\beta x}$
2. As above, with $y = e^{\alpha}x^{\beta}$
3. Mid-points of the horizontal sections only are used for fitting $y = e^{\alpha+\beta x}$
4. Mid-points, $y = e^{\alpha}x^{\beta}$

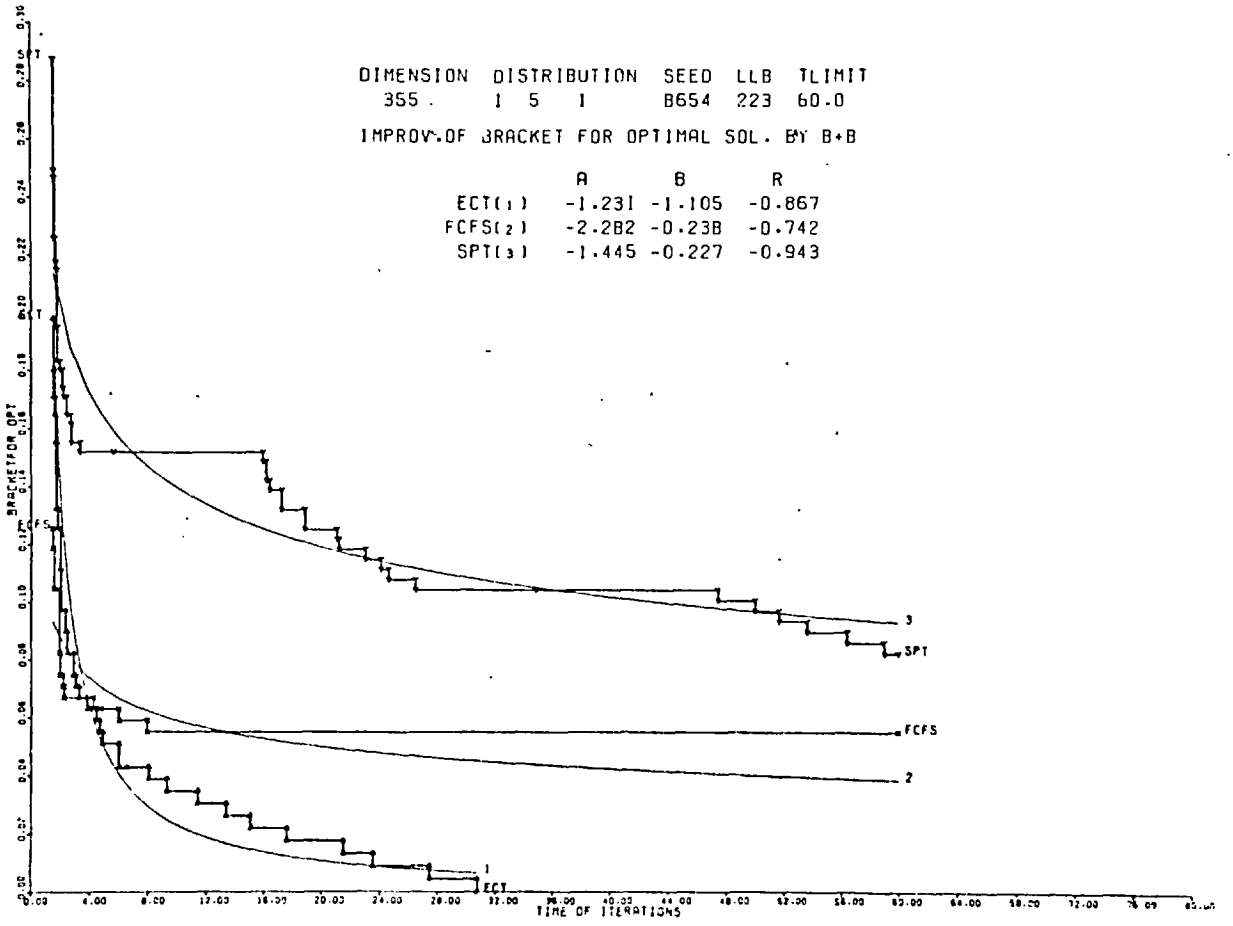
In order to take into account also the interval during which the value of the upper bound is not improved, without considering every single iteration, the first and the final point of the horizontal lines are used for fitting

5. $y = e^{\alpha+\beta x}$
6. $y = e^{\alpha}x^{\beta}$

A number of problems has been tested for each of these fittings (an example is given in Appendix C) and it is thought that the most appropriate one is the one taking the two points from each solution value and fitting a curve $y = e^{\alpha}x^{\beta}$ by defining α and β with a least square linear regression of $\log y = \alpha+\beta \log x$.

The bracket for the optimal C_{max} (BOS) has been used because it is thought to be more useful than the actual solution values, allowing comparisons of different problems. This model has been used to describe the improvements of the bracket for optimal solutions (BOS) of the test problems and the correlation coefficients were satisfactorily high, as can be seen in Table 5.8 (some typical cases are illustrated in Figure 5.9 on the following page).

Figure 5.9 Typical patterns for improvements of the bracket for the optimal solution BOS



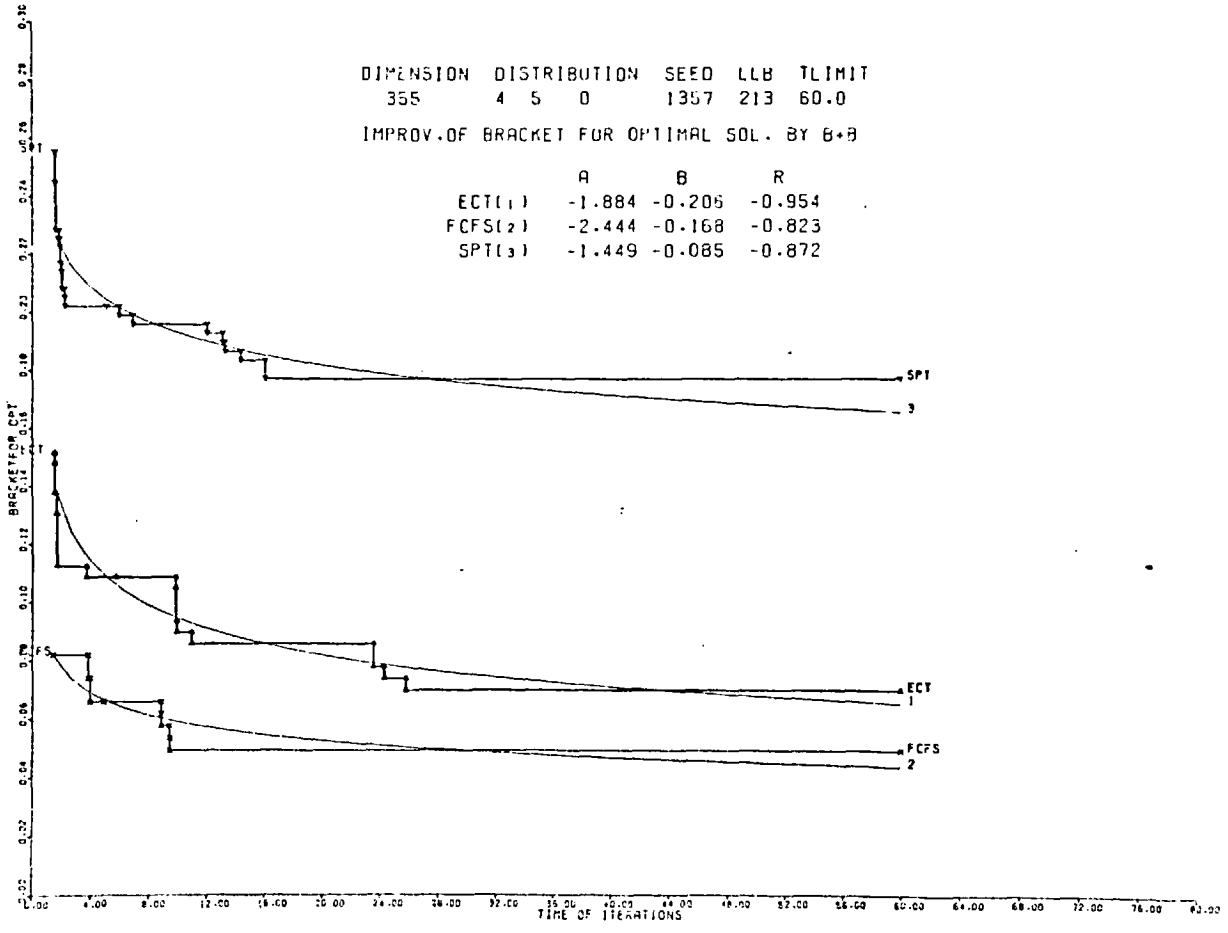


Table 5.8 Quality of continuous approximation of BOS improvements function

Problems		% of cases with r greater than		
Jobs	Machines	.6	.8	.9
8	4	97	60	34
10	4	91	45	30
20	5	90	62	41
35	5	92	73	33

The value of the regression constant α represents an imaginary bracket value at time zero, and therefore α has no physical meaning. The function $f(x) = e^{\alpha}x^{\beta}$ is in fact defined for $t_o \leq x \leq TL$ (t_o: time of completion of first feasible solution). The value at t_o, $f(t_o) = e^{\alpha}t_o^{\beta}$ corresponds to the initial solution and the value at TL $f(TL) = e^{\alpha}(TL)^{\beta}$ corresponds to the last (best) solution in the LNS. It should be added here that the values of α and β are not independent of the time limit used, though their values would not change dramatically for different TL.

Given the similarity in the pattern of the solution improvements, a range of values for α and β might be established for each type of problem. A distinction is necessary though between problem-heuristics that reach optimality within the given time limit and those that do not. One cannot expect to find similarity in the graphs of these two groups. In fact, the whole idea of using a model of solution improvements is related to those problems where optimality is not reached within TL.

For problems with UR, E_1, E_4, E_9 and E_{36} processing times, 10 jobs 4 machines (two groups of 10 problems each with different time limits) and 35 jobs 5 machines (5 problems), the mean values of β and the standard deviation are given in Table C1 of Appendix C. As can be seen from these results, the values of the regression coefficient

vary within a range of coefficients of variation usually lower than 0.5 and not exceeding 1.0. For problems of 10 jobs and 4 machines, 10 out of the 15 different averages are lower for the larger time limit, indicating that the efficiency of the search diminishes with time. This reflects the fact that solution improvements become increasingly difficult with time. The values of the average β show also the differences in performance of the heuristics, although the patterns of change with the variance of processing times is less clear. The smaller values of $\bar{\beta}$ at the larger problems should be expected, as a consequence of the larger problem size and of the time limit used. These result are not comparable with those of Table 5.5 on average ranking and of Table 5.7 on average BOS, because they include also the cases where optimality was achieved, while those reported in Table C1 are for cases of incomplete search.

5.5.2 Stopping rules

The usual criterion for stopping a tree search has been the construction of an optimal (exact) solution. This, as already discussed, can be very expensive if not infeasible, and certainly it is very inefficient. In a large number of cases (see Table 5.2, on how often the least of lower bounds is a feasible-and therefore optimal-solution) an optimal solution is constructed early in the tree-search and then an exhaustive search is needed to prove its optimality. Another case of inefficiency takes place when a near-optimal solution is constructed early in the search and exhaustive enumeration takes place to obtain relatively minor improvements, whose value is questionable, as discussed in Section 5.3.

The method adopted in this chapter has been to find the best possible solution within given resources, quantify its quality relative to the optimal and to the other heuristic solutions obtained, estimate the complexity of the problem, the extent of search that allowed its

construction, and get some indication of the improvements that can be expected to be realised with further search.

The first implication of the above information is that when the tree search in a particular problem approaches the prespecified time and the proportion of the tree searched, without obtaining an optimal, is very small, then the probability of obtaining a substantially better answer with a limited extension of the search is very small. If on the other hand, a large proportion of the set of active solutions has already been searched, and no optimal solution has been found, then it is possible to decide an extension, in order to exhaust fully the tree and either find an optimal solution, or prove the optimality of the last solution. Another implication is that the results up to the current instance of the solution can dictate the extent of further search, in a sequential decision making process. There is some subjective benefit associated with an expected solution improvement and a cost related to the expected amount of time required to obtain the solution improvement. The decision rule for stopping would then be based on the comparison of the values assigned to these two quantities.

The model described in the previous section for solution improvements over time can be used as an estimate of the time (cost) required for one more improvement. It is clear that the same amount of improvement becomes more expensive as the bracket of the distance from the optimal BOS is reduced. An alternative formulation that is possible is to express the solution improvements as a function of the proportion of the tree searched (and not as a function of time).

The feasible solution space A has a probability density function $f(x) = \text{Prob}(x-\frac{1}{2}<X<x+\frac{1}{2})$ where, in the formulation adopted, x takes

integer values only (x has discrete values in any formulation). The branch and bound search of a proportion P of the tree results in a solution value v_b. The frequency of values $z<v_b$ is $F(v_b) = \int_{e}^{v_b} f(z)dz$ and if one assumes that values are distributed randomly within the solution space (which is not quite true, but can be acceptable as a hypothesis, if large parts of the tree are considered), then the probability of a better solution in the remaining part of the tree is $(1-P)F(v_b)$. The idea of using statistical properties of the solution space not only in stopping rules but also for other purposes will be discussed in detail in the following chapter.

CHAPTER 6

STATISTICAL METHODS IN LOCAL NEIGHBOURHOOD SEARCH

6.1 Sampling methods in job-shop scheduling

6.2 Distribution of values of active schedules

6.3 Estimates of optimal solution value

6.4 Applications in stopping decisions and bound calculations

6.1 Sampling methods in job-shop scheduling

The Local Neighbourhood Search method with heuristic decision rules for conflict resolution described in the previous chapter is a potentially powerful tool for solving approximately deterministic job-shop scheduling problems. The efficiency of the LNS method for a given decision rule depends on the quality of the lower bounds calculated and on the time of termination of the search. The latter is particularly crucial, as already seen in Section 5.2, where in about half of the test problems the least of lower bounds b_e^* has not been realizable as a feasible solution. In these cases, either an enormous amount of computational resources may be required to prove optimality by exhaustive search, or in the case of incomplete search, a poor value of the bracket for the optimal solution BOS may be obtained.

If the value of the optimal solution is known somehow in advance, then the implementation of stopping decisions can improve the efficiency of the LNS. Such information can be used also for lower bound purposes: each node in the tree search corresponds to a fixed partial sequence, and all the unscheduled operations form a surrogate problem, whose optimal solution can be used as a lower bound for the solution of the original problem. It is believed that statistical sampling methods can be used for the purpose of estimating the value of the optimal solution in a tree search method. The background to the use of statistical methods in job-shop scheduling will be reviewed below with the objective of developing and testing suitable estimators of the optimal solutions.

The most crude way of using statistical methods in scheduling problems is random sampling of solutions from the set A of active schedules.

This set, as already seen in Section 2.3.1 , is very large even for the more modest problem sizes, and the probability of selecting an optimal solution randomly is very small. Taking a sample of N random solutions and selecting the best of them is not satisfactory as an approximate method, for there is no way of determining a suitable sample size N and for the low confidence in the results.

The first serious attempt to use sampling methods for solution generation, was conducted by Heller(1960). In a large scale investigation, he found experimentally that the distribution of feasible schedule makespan times in a flow-shop approaches the normal distribution, and tried to explain it theoretically with an'approximation to the central limit theorem for a simple periodic Markov chain!

The above result led this study to the formulation of the hypothesis that the distribution of makespan values in job-shop scheduling problems, can also be approximated by a Normal distribution $N(\mu,\sigma)$. This hypothesis has been tested in a number of problems, by constructing random solutions, with the help of the basic algorithm already described and used for active schedule generation in Chapter 4, and with a random number generator for deciding at every conflict node which branch to follow.

The hypothesis of normally distributed makespan (C_{max}) values of the problems of Table6.1has been tested with the X^2 test and with the Kolmogorov-Smirnov non-parametric test, as described briefly below (Hoel,1971).

X^2 -test: A number k of classes is defined by the lowest x_{min} and highest x_{max} of the integer random solutions, C_j, for j=1...N, the observed values being v_i, i=1...k. The intervals are defined by $v_i \pm \frac{1}{2}$, the corresponding observed number of occurences is O_i and the expected E_i. The quantity $X^2 = \sum_{i=1}^{k}\{(O_i-E_i)^2/E_i\}$

is approximated by a X^2 distribution with a degree of freedom d.o.f.=k-1-n_p, where n_p is the number of parameters estimated from the sample (for a Normal distribution n_p=2) provided that $E_i \geq 5$, $k \geq 5$.

Kolmogorov-Smirnov test: In this test two samples are tested to see if they are from the same population. The observed cumulative frequency is compared to the expected, and the maximum difference D is recorded and compared with the critical values which depend on the sample size.
(Table D1 in Appendix D)

As estimators of the parameters of the Normal distribution, the mean and the adjusted standard deviation of the random sample values have been used. The test problems processing times have been taken from Erlang distributions with the same average. Two different values of the Erlang parameter have been used, k=1 and k=9, in order to investigate the effect of the variance on the distribution of the values of the set A. The results of this experiment are summarised in Table 6.1 below.

Table 6.1 Test of assumption of normally distributed makespan values.

Problem	Dimensions n	m	Proc.times distrib.	Sample size N	$\bar{C}_{max}$	σ	V	X^2-test level of signif.	K-S test max.diff. D
P_1	8	4	E_1	400	91.45	8.52	.093	.0032	.0371
P_2	8	4	E_1	400	85.15	8.77	.103	.0000	.0863
P_3	8	4	E_1	400	75.42	5.61	.075	.4000	.0392
P_4	8	4	E_9	400	69.59	4.39	.063	.0001	.0406
P_5	8	4	E_9	400	61.23	4.99	.081	.0025	.0772
P_6	8	4	E_9	400	66.13	4.12	.062	.0435	.0383
P_7	20	5	E_9	400	138.02	7.31	.053	.1437	.0452
P_8	35	5	E_9	120	230.33	8.85	.038	.3390	.0384

The resulting values of χ^2 and the corresponding levels of significance (Table 6.1) show that the hypothesis of Normally distributed makespan values cannot be rejected at a level of significance of 2.5% in 6 out of the 8 test cases. The Kolmogorov-Smirnov test results, summarised in Table 6.1, show that the hypothesis of normally distributed values cannot be rejected at a level of significance of 1% and in 6 out of the 8 cases at 20%. It should be noted here that in problem P_2 several sample schedules had a value equal to the a-priori lower bound to the optimal solution, and thus it is not surprising that the distribution is not approaching the normal. It is worth adding that the mean μ and standard deviation σ of the sample values, and therefore the coefficient of variation $V=\sigma/\mu$, are fairly stable for different sample sizes in each problem.

For the same test problem size of 8 jobs and 4 machines, there is a substantial difference of the $\bar{C}_{max}$ values for E_1 and E_9 distributions of processing times, as one might reasonably expect. The values of the coefficient of variation V show also some consistent small difference, indicating lower dispersion of values about the mean, for lower data variance. For increasing problem size, V is again decreasing, probably due to the larger number of operations, allowing some fluctuations of the makespan values to be absorbed.

The hypothesis, that the values of C_{max} are approximately normally distributed, although not rejected with the tests described above, has the fundamental drawback that it assumes an assymptotic behaviour. This is not true since the values of active schedules are bounded from below by the optimal v_b^* (or a lower bound b_e) and from above by the worst possible case v_w^* (or an upper bound b_u). The normality assumption does not give any indication of what the optimal v_b^* or the

worst case v_w^* might be. Besides, the observed frequencies are somehow skewed, deviating from the symmetric bell-shaped form of the Normal. This is not unexpected, because the full set A of active schedules, described by a frequency histogram (an empirical distribution) may not be described exactly by a Normal or any other simple theoretical distribution.

6.2 Distribution of values of active schedules

A more accurate description of the empirical distribution of values of the set of active schedules, involving n-1 parameters, where n is the number of classes defined by an upper bound and a lower bound of set A, was suggested by Cunningham and Turner (1973) and Randolph et al (1973), based on the Bayesian theory. This method is summarised below.

The population from which the sample is drawn has a distribution, known as 'state of nature' and described by θ. An estimate of this constant 'state of nature' can be measured by means of a prior probability distribution $b(\theta)$. Following the observation of an experimental outcome x, the revised probability distribution (or posterior probability distribution) is according to Bayes theorem

$$b(\theta/x) = \{b(\theta).p(x/\theta)\}/\{\sum_{all\theta} b(\theta).p(x/\theta)\}$$

A sequential sampling of N elements allows N revisions of the posterior distribution. If θ were not the description of a distribution but a simple variable, then the computational effort would be manageable. Revising though a distribution described by a vector $\bar{p}$ of probabilities p_i, i=1...n (where n is the number of classes) is much more complex and computationally expensive, which makes the practical efficiency of this excellent theoretical idea questionable. Assuming Normal distribution for A certainly would help shorten the calculations, but, as seen in the preceding section, is not exactly correct. The above mentioned authors have used a Dirichlet distribution with n-1 variates:

$$b_o(p_1,\ldots,p_{n-1}) = \{\Gamma(\sum_{i=1}^{n} k_i+n)/\Gamma(k_1+1)\ldots\Gamma(k_n+1)\}p_1^{k_1}\ldots p_{n-1}^{k_{n-1}}(1-\sum_{i=1}^{n-1} p_i)^{k_n}$$

where k_i are positive integers and n is the number of classes assumed for the distribution values. The posterior distribution is again of

the same type, called conjugate to the prior. If the sample value corresponds to class i then k_i is updated to k_i+1. Otherwise the value of k_i remains the same.

A good description of the empirical distribution of active schedule values is possible with the above method, using a large number of parameters. It is though unlikely that more than one problem could be described with the same combination of parameter values. Thus the main drawback of the method remains the length of computations involved.

If one does not insist on the exact description of the set A, then approximate descriptors, better than the Normal and simpler than the Dirichlet, are possible with theoretical distributions of more than two parameters. A simple theoretical distribution which seems to be more suitable than the Normal for describing the values of active schedules is the 'Weibull'. This is a three-parameter distribution which may approach the shape of the Normal, but which is bounded from below, i.e. Prob(x<a)=0 for some value of a. The probability density function of the Weibull (Hastings and Peacock, 1975) is:

$$f(x) = c/b\ ((x-a)/b)^{c-1}\ \exp[-((x-a)/b)^{c}]$$

where a location parameter

b scale parameter

c shape parameter

The cumulative distribution function is:

$$F(x) = 1 - \exp[-((x-a)/b)^{c}]$$

where for c=1, the Weibull distribution reduces to a negative exponential.

The hypothesis of Weibull distributed sample values was tested for the problems of Table 6.1. The calculation of estimators of the

Weibull parameters can be based on the limited number of discrete (integer) values that the location parameter a may take:

$$b_e \leq a \leq x_{min}$$

a : integer.

b_e : lower bound to the values of set A.

x_{min}: lowest value of the sample.

The method used to calculate the scale b and shape c parameters of the Weibull for every value of a, has been based on the relatively simple form of the cumulative distribution function

$$F(x) = 1 - \exp\ [\ -((x-a)/b)^c]$$

$$\rightarrow \quad \exp[-((x-a)/b)^c] = 1 - F(x)$$

$$\rightarrow \quad -((x-a)/b)^c = \ln(1 - F(x))$$

$$\rightarrow \quad c[\ln(x-a)-\ln b] = \ln[-\ln(1 - F(x))]$$

$$\rightarrow \quad \ln[-\ln(1 - F(x))] = -c\ln b + c\ln(x-a)$$

For

$$Y = \ln[-\ln(1 - F(x))]$$

$$X = \ln(x-a)$$

$$\alpha = -c\ln b$$

$$\beta = c$$

$$\rightarrow \quad Y = \alpha + \beta X$$

This form offers itself for a least square linear regression, given that F(x) can be estimated from the sample available. The regression is repeated for all values of a, and every time b,c are calculated from $c=\beta$ and $b=\exp(-\alpha/\beta)$. The best estimate of the optimal solution is the value a with the highest regression coefficient r.

This method has been applied to the test problems of Table 6.1 with seemingly very good results. The resulting values of the correlation coefficient r given in Table 6.2 below, are very high indeed, approaching 1.0 in most of the cases. These results though should be accepted with some caution, because by taking logarithms twice,

significant differences may become smaller, in which case, the real fitting is not as good as the value of r might suggest.

For this reason, the parametric X^2 and the non-parametric K-S tests were used to test the hypothesis that the sample values are Weibull distributed with parameters a,b,c estimated as above. Table 6.2, gives some additional information for the problems investigated (least of lower bounds b_e^*, best known solution from tree search v_b and minimum of random sample x_{min}) and summarises the results of this experiment.

Table 6.2 Hypothesis testing of Weibull distributed values of active schedules.

Problem	b_e^*	x_{min}	v_b (* optimal)	Highest r	a	c	X^2-test level of signif.%	K-S test max Diff. D
P_1	63	73	*67	.998	71	2.5	3.1	.0314
P_2	71	71	71	.814	71	(0.7)	(0.0)	(.3257)
P_3	51	58	57	.981	55	3.9	12.5	.0538
P_4	51	59	*55	.996	56	3.3	0.01	.0468
P_5	50	51	*50	.998	50	2.3	0.3	.0770
P_6	48	55	54	.997	52	3.8	2.8	.0305
P_7	120	123	*120	.999	121	2.5	16.3	.0302
P_8	209	210	210	.977	209	2.2	16.7	.0907

If one accepts that a higher level of significance for the X^2 or a lower maximum difference for the Kolmogorov-Smirnov test implies a 'better fit', then comparison of results in Tables 6.1 and 6.2 indicates that the Normal assumption is better than the Weibull only for 3 out of the 8 test problems with each of the two tests. One may also notice that the values of the Weibull shape parameter c for the best

fit lies between 2.2 and 3.9, with smaller values for the larger problems P_7 and P_8. For this range of c-values the Weibull distribution is bell-shaped and fairly symmetrical, looking not very different from a Normal.

For the test problems P_1 - P_6 additional random solutions have been generated for a sample size of N=2000, to be used as described in the following sections. This increased sample size has been used here to test again the hypothesis that the active schedule values are from a Normal or a Weibull distribution. The hypothesis has been rejected with both the X^2 - test and the Kolmogorov-Smirnov test. This was to be expected because,although the empirical distribution of the values of active schedules can be approximated with a Normal or a Weibull, it is not exactly either of these. This becomes apparent with large samples (eg. N=2000) where the maximum difference D of the K-S test should not exceed $1/\sqrt{N}$ and where the X^2 value should be small. In fact in this experiment D did not become zero and the same proportional difference of observed and expected values gave a small X^2 value in small samples and a larger one in large samples.

6.3 Estimates of optimal solution value

Distribution of smallest members of samples of active schedules

The assumption that the values of set A are from a Normal or Weibull distribution has the drawback of being relatively inaccurate. The calculation of a more accurate descriptor, like the Dirichlet, is very complex, as discussed in the preceeding section. One way of going round these problems is to use properties of samples, like the central limit theorem, without assuming any particular form for the distribution of the parent population. According to the central limit theorem, the averages of N samples, each of size M, are normally distributed with the same mean and standard deviation $\sigma / \sqrt{M}$, where σ is the standard deviation of the parent distribution.

A more useful concept, applied for the Travelling Salesman problem by Golden (1977), is that the distribution of the minima x_i of the N samples bounded from below by the optimal solution, approaches a Weibull (Fisher and Tippett, 1927, Weibull, 1951). This concept has been used in this chapter, where the parameters of the Weibull disribution of the minima of samples of active schedules have been estimated with two different methods: one based on the principle of maximum likelihood (Mood and Graybill, 1963) and the other on regression (as discussed in the preceeding Section 6.2).

Maximum likelihood estimators of Weibull parameters

The likelihood function of N random variables $x_1,\ldots,x_N$ from a population described by $f(x,\theta)$ is the function

$$L(x_1,x_2,\ldots,x_N,\theta) = \prod_{i=1}^{N} f(x_i,\theta)$$

Many likelihood functions (e.g. θ describing Normal or Weibull distributions) satisfy regularity conditions, so that the maximum likelihood estimator is the solution to the equation $dL(\theta)/d\theta = 0$.

For the Weibull distribution,

$$L = (c/b)^N [\prod_{i=1}^{N} (x_i-a)/b]^{c-1} \exp[-\sum_{i=1}^{N} ((x_i-a)/b)^c]$$

which is maximised for the same values of a,b,c, as the function

$$\ln L = N\ln c - N\ln b + (c-1)[\sum_{i=1}^{N} \ln((x_i-a)/b)] - \sum_{i=1}^{N} ((x_i-a)/b)$$

for

$$\partial \ln L/\partial a = 0$$

$$\partial \ln L/\partial b = 0$$

$$\partial \ln L/\partial c = 0$$

The maximum likelihood estimates of a,b,c are calculated by solving the system of these three simultaneous equations. This is a rather difficult task from the computational point of view, if the general methods of numerical techniques are used. This general methodology does not take into account some special features of the formulation of the problem, namely that the values of all members of A and the lower bound are integers, which limits the solution space for the parameter a to a small number of discrete (integer) values. In fact parameter a must lie between the lower bound b_e and an upper bound defined as

$$b_u = \min[x_i] \quad \text{for} \quad i=1,2,\ldots,(Nm)$$

$$b_e \leq a \leq b_u$$

From

$$\partial \ln L/\partial b = 0 \rightarrow [-Nc/b] + \frac{c}{b^{c+1}} \sum_{i=1}^{N} (x_i-a)^c = 0$$

$$\rightarrow \quad b = g(a,c) = [(\sum_{i=1}^{N} (x_i-a))/N]^{1/c}$$

Substituting in L,

$$L = [\frac{c}{g(a,c)}]^N \; [\frac{1}{g(a,c)^N} \prod_{i=1}^{N} (x_i-a)]^{c-1} \exp(-N)$$

which is simple enough to allow calculations for all possible values of a, and for a wide range of c ($0 < c$)

To avoid infinite and indefinite results, a is adjusted by a small quantity ε (ε=.001). Following these calculations, the largest value of L indicates the best estimates of the Weibull parameters a,b,c.

The method described above has been applied for the test problems P_1-P_6, using a range of sample sizes (M) and of number of samples (N). For N=10 samples of M=40 independent members each (sample size of 400) the values of L calculated for all six problems have been very small varying between E-7 to E-18, except for $a+\varepsilon=x_{min}$, where the fluctuations due to changes in the shape parameter c have been enormous. The same problems have been investigated with the parameter a corrected by ε (for all values of a) and the results remained the same (i.e. the effect of ε is insignificant). The most likely values of a and c resulting from this method are given in Table 6.3 and are discussed in some detail below:

- P_1 Results inconclusive (same value of L for many a's).
- P_2 No analysis possible since $x_{min}=b^*_e$.
- P_3 Results compatible with LNS (optimal solution unknown).
- P_4 Results do not agree with known optimal solution.
- P_5 Results agree with known optimal solution.
- P_6 Results compatible with prior information (optimal unknown).

While the sample size M=40 is considered to be sufficient, the number of samples N=10 is thought to be rather small for allowing high confidence in the results. Thus, it has been necessary to use larger samples of up to 2000 random solutions, in order to estimate the parameters of the Weibull distributed sample minima. In this extended experiment, the number of samples N has been taken equal to 10, 20 and 40, resulting in reduced values of the likelihood function (ranging from E-12 to E-35) and the sample size M has been taken equal to 50, 100 and 200. The most likely values of the Weibull parameters a and c are given in Table 6.3 below.

Table 6.3 Maximum likelihood estimators of Weibull parameters.

Problem	Lower bound b_e^*	Best known solution v_b	x_{min}	N=10 M=40 a	N=10 M=50 a	N=20 M=50 a	N=40 M=50 a	N=20 M=100 a	N=10 M=200 a
P_1	63	67*	73	68 c=5.0	72 c=4.5	72 c=2.5	72 c=2.5	73 c=0.8	73 c=0.4
P_2	71	71*	71	(71) c=0.5	(71) c=0.5	(71) c=0.5	(71) c=0.5	(71) c=0.5	(71) c=0.5
P_3	51	57	58	51 c=9.0	51 c=9.0	51 c=9.0	51 c=9.0	51 c=8.0	51 c=8.0
P_4	51	55*	59	59 c=0.5	59 c=1.0	59 c=1.0	59 c=1.0	59 c=0.8	59 c=0.4
P_5	50	50*	51	50 c=4.0	51 c=0.5	51 c=0.5	51 c=0.5	51 c=0.4	51 c=0.4
P_6	48	54	55	51 c=5	55 c=0.5	53 c=3.0	52 c=4.0	55 c=0.8	55 c=0.4

As can be seen from Table 6.3, the results of this method are relatively unstable in the sense that for different sample sizes the most likely values of the location parameter a vary in 3 out of the 6 problems. One explanation might be that the differences of the likelihood fuction value are very small, of the order of E-30, and at this range, computational results are not accurate enough. In 3 out of the 6 test problems the numerical results obtained with the increased samples do not agree with the independent information available. It is thought that larger samples might allow more conclusive and accurate results but the handling of larger samples would render the method impractical. As an alternative, a different method has been used to find the most likely values of a, using least square linear regression, as in Section 6.2.

Least square regression estimates of Weibull parameters.

The experiment has been conducted with the 2000 random solutions available for each of problems P_1 to P_6, for N=40 samples, each of size M=50 and the results are summarised in Table 6.4. The number of samples has been set to N=40, because the X^2-test (which is generally more reliable than the K-S test) requires at least 5 distinct classes

of 5 members each, i.e. a mimimum of 25 items. Taking into account that some classes may have less than 5 members and therefore have to be merged, it has been decided to set N=40.

Table 6.4 Linear regression estimates of Weibull parameters (with N=40, M=50).

Problem	Highest r	Parameter Estimators a	c	X^2-test level of signif.%	K-S test Max. Diff. D
P_1	.996	65	7.3	7.3	.1430
P_2	Regression not feasible				
P_3	.823	51	4.9	0.0	.2892
P_4	.983	55	4.2	-	.1463
P_5	.975	50	2.0	-	.2820
P_6	.979	48	6.4	0.3	.1977

The results of the above Table 6.4 are satisfactory in the sense that the most likely value of a is compatible with the independent information available in all except one problem (P_1). In this case the difference of a=65 and the known optimal v_b^*=67 is very small.

6.4 Applications in stopping decisions and bound calculations

The importance of the statistical methods described in the preceding sections for scheduling problems lies, not in their theoretical interest, but in their application to enumerative methods. The basic idea underlying all these statistical methods is that for a given problem P, the values of active schedules can be classified in a finite number of classes, defined by an upper bound and a lower bound. Each particular class v has either zero or positive probability of containing at least one member. The Weibull location parameter a corresponds to the class with lowest v that is likely to contain at least one member. This member is to be treated as an estimate of the value of the optimal solution to the scheduling problem. It should be stressed that this method may suggest a particular value v as the most likely optimal, but it does not say how to construct such a solution. The actual construction has to be based on some form of enumerative procedure (e.g. branch and bound, tree search), which is the methodology used in the previous chapter. The additional insight offered by these statistical methods is the probability associated with the unknown solution sequence with value v. (Erdos and Spencer,1974).

There are two ways by which estimates of the optimal solution value can be used to improve the efficiency of LNS. They can be applied for stopping decisions and bound calculations. One of the main problems in branch and bound is that the least lower bounds calculated may be infeasible and therefore 'poor' compared with the actual yet unknown optimal solution. A lot of computational effort may be wasted trying to emumerate a tree, not knowing whether the best known value is the optimal or not. There are many cases where an optimal solution is constructed early in the search and the rest of the time

is spent trying to prove it is optimal by exhaustive search. Such an enumeration can be computationally very expensive, even in the case where the least of lower bounds is equal to the optimal solution. This is where the statistical methods can be of help. By obtaining an estimate a of the optimal solution, one may see that a solution with that value has already been constructed, and instead of proving optimality by exhaustive enumeration of the remaining tree, it is possible to terminate the search procedures, assuming that the Weibull parameter a is not an over-estimate of the real optimal solution.

If the best known solution v_b is greater than a, it is possible to obtain an estimate of the frequency of feasible solutions with value lower than v_b. The frequency or probability of non-empty classes lower than v_b expresses the expectation of finding solutions better than v_b and can be used subjectively for stopping decisions. An alternative application of the statistical estimate of optimal solution values is in the sphere of bounds. The efficiency of the statistical methods as well as of the tree-search depends to a large extent on the 'quality' of the lower bounds b_e that can be calculated. An indication of their quality is given by the discussion of how often they are realizable as feasible solutions, i.e. whether the class $v=b_e$ is empty or not. (see Section 5.2). A useful hypothesis is to assume, for given b_e and x_{min} (or b_u) that at least one of the classes $b_e, b_e+1, \ldots, b_e+\delta(b_u-b_e)$ for $\{0 < \delta < 1 \atop b_u'=\delta(b_u-b_e)\}$ is non-empty, and therefore reduce drastically the computational task, by using a 'fictitious' upper bound b_u'. If the hypothesis is rejected, then the revised lower bound becomes $b_e'=b_u'$ and the procedure is repeated. If not, then a new upper bound is found, used for further search (Bazaraa and Elshafei, 1977). Instead of using fictitious bounds,which are defined randomly, it is more efficient to use a statistical estimate of the optimal solution as an upper bound in a tree-search procedure, trying

to establish a feasible solution less than or equal to a. If such a solution is found, it can be either accepted or one could go further by trying a reduced value of a, progressively, until feasibility is violated, which would prove optimality. If no feasible solution with value a is found, one can increase progressively the value of the upper bound used, until feasibility is obtained and therefore optimality.

The sampling method for estimating the value of the optimal has also another potential use. One can obtain an estimate of the optimal solution of a sub-problem, defined by a branch and bound partitioning method as a node corresponding to a fixed partial sequence and use this estimate as a lower bound. In such a case, from the set of active nodes, the one with the lowest estimate can be selected for further search in a best-bound-first procedure (tie breaking by selecting the nodes with the highest likelihood). This would be a strong bound, but one should take into account the computational cost of calculating this estimate of the optimal solution for the sub-problems.

CHAPTER 7

APPROXIMATE METHODS FOR STOCHASTIC AND DYNAMIC SCHEDULING PROBLEMS

7.1 Queueing theory and simulation in scheduling

7.2 Design of the experiment

7.3 Sensitivity of simulation results to the processing times structure

- 7.3.1 Effects of changes in the distribution function
- 7.3.2 Effects of changes of the variance of processing times on the performance of scheduling rules at fixed load factor.
- 7.3.3 Effects of changes of processing times variance with variable load factor.
- 7.3.4 Effects of inaccuracy of processing times estimates.

7.4 Evaluation of a new composite global scheduling rule

- 7.4.1 Description of the scheduling rule
- 7.4.2 Calibration and evaluation of the composite rule

7.5 Job-shops with identical machines in parallel

7.6 Approximate formulae for networks of queues

7.1 Queueing theory and simulation in scheduling

The dynamic and deterministic job-shop, defined by different job arrival times ($R_i \geq 0$) is obviously more complex than the static one. P-class algorithms are available only for a few special cases of the single machine problem: C_{max} with precedence constraints, $\bar{C}$, L_{max} with precedence constraints and unit processing times, $\bar{T}$ with unit processing times, $\bar{C}$ with m-machines in parallel and unit processing times. It has been established that for the majority of single machine problems and all the m-machine problems, the dynamic case with $R_i \geq 0$ is clearly NP-complete (see Chapter 2).

For the stochastic problem, static or dynamic, there is no enumerative method that can give any answer at all. There are though a few limited cases where an analytical approach can be applied, by considering the processors as queueing systems. (Bruno, 1976)

The queueing theory approach can not give deterministic results. Probabilistic results can be obtained for certain problems under steady-state conditions. Mathematical analysis of the transient state is very complex even for the simpler models (Cox and Smith, 1961, Lee, 1966). Thus one can not have exact or optimal solutions, and the probabilistic ones are clearly approximate.

The classification used for the static-deterministic scheduling problem (n/m/routing/criterion of performance) is not appropriate for the stochastic and dynamic features of the queueing models, where one needs information on the arrival pattern (frequency, batch-size etc), on the processing times (deterministic or stochastic) and on the availability of facilities in parallel, which are described by the parameters p_5, p_6, p_7. Additionally, one needs to know the queueing discipline p_8, the size of the population of jobs p_9 and the maximum queue size allowed

p_{10}. The queue is thus described by the six parameters

$$p_5/p_6/p_7:p_8/p_9/p_{10}$$

Analytical results available for the average waiting time of some simple models FIFO/∞/∞ are given below, where the load factor (or ratio) is defined as $\rho=\lambda_a/r\lambda_s$,r is the number of identical servers in parallel, λ_a is the arrival rate and λ_s is the service rate (Kleinrock, 1975 and 1976). In queues with Poisson (negative exponential) arrivals and service times, the expected waiting time W is:

$$\text{M/M/1} \qquad \lambda_s W=\rho/(1-\rho)$$

$$\text{M/M/2} \qquad \lambda_s W=\rho^2/(1-\rho^2)$$

Analytical results are also available for some'non-Poisson-queues' (where p_5 or p_6 are not negative exponential) which are more complicated.

$$\text{M/D/1} \qquad \lambda_s W=\rho/(1-\rho)$$

$$\text{D/M/1} \qquad \lambda_s W=z_0/(1-z_0) \quad \text{for} \quad 0<z_0<1$$

$$\text{root of } z=\exp\{-(1-z)/\rho\}$$

$$\text{M/D/2} \qquad \lambda_s W=\frac{2(1-\rho)+(1-z_0)(2\rho^2-1)}{4\rho(1-\rho)(1-z_0)} \quad \text{for} \quad -1<z_0<1$$

$$\text{root of } z^2\exp\{2\rho(1-z)\}=1$$

A well known case is the M/G/1: GD/∞/∞ (GD: general discipline) where the expected number in the system L_s is calculated by the Pollaczec-Khintchine formula (Taha, 1976)

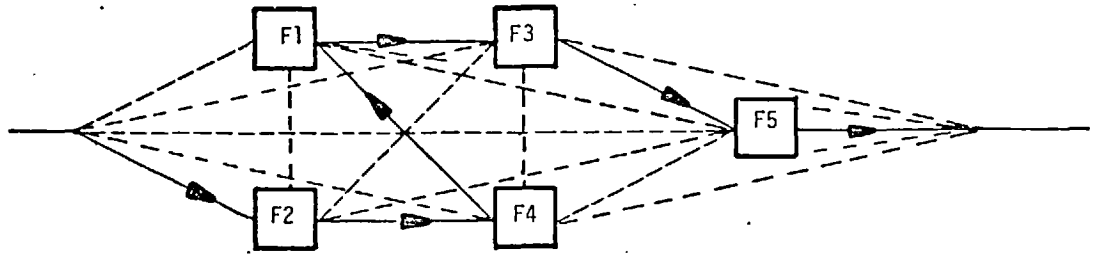
$$L_s=\lambda_a E(t)+\lambda_a^2\{E^2(t)+\text{Var}(t)\}/2\{1-\lambda_a E(t)\}$$

The operating characteristics for any service time distribution such as gamma or constant service times can be obtained directly from the above formula.

The more realistic job-shop problems with batch arrivals and more than

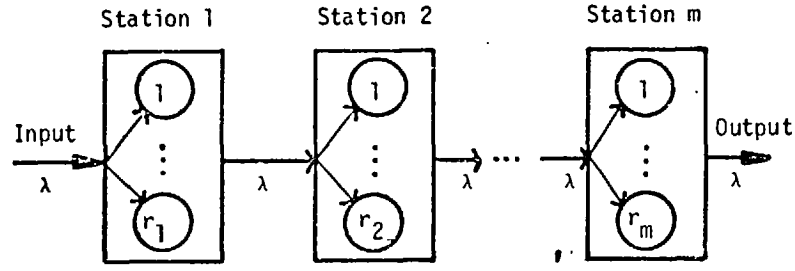
one processor are complex queueing systems with servers in parallel and in tandem and are represented by networks of queues. Figure 7.1 below shows a system with five queues and one of the possible routes.

Figure 7.1 Network of queues



The analytical results for networks of queues are very limited. For queues in series or in tandem the most important result is based on a theorem stating that for Poisson input and service times and general queue discipline, the output from the processor is also Poisson (Saaty, 1961, Taha, 1976). The same result can be extended to the case of more than one identical machines in parallel. This model can be used for studying flow-shops like the one illustrated in Figure 7.2, with negative exponential interarrival times of single jobs and service times.

Figure 7.2 A flow-shop like queueing system



Rao (1976) has suggested an analytical method for a three-stage series production system with identical Erlang service times, reducing drastically the number of simultaneous equations that have to be solved.

For many queueing systems however, analytical results are either unknown or too complex to derive, and approximate methods have been used with success for a number of models, usually single stage ones (Kingman 1970, Page 1972, Cosmetatos 1974 and 1975, Rosenshine and Chandra 1975, Marchal 1976, Cosmetatos 1976 and 1977). Some of these formulae are given in Appendix E.

Although the approximate results simplify the study of a number of queueing problems, they cannot offer any answer to the more general job-shop problem represented by a network of queues (for which a method based on approximate system formulae will be considered later in this chapter). The only method available that can be applied to problems of higher complexity is simulation.

Simulation experiments for scheduling problems have been reported for the first time by Rowe and Jackson (1956), Jackson (1957), Baker and Dzielinski (1960) and since then a lot of research has been devoted in simulation of job-shops especially in the study of queueing disciplines (Gere 1966, Conway et al 1967, Hollier 1968, Chowdhury1976, Panwalkar and Iskander 1977). Simulation experiments with priority dispatching rules have been conducted also for special structure job-shops, e.g. in a parallel processor shop (Moodie and Roberts, 1968). The decision rules applied for selecting a job from the queue of each machine, are what has been described in Section 4.2 as non-delay rules (dispatching rules or loading rules),i.e.as soon as a machine is available and jobs are waiting for processing, the machine is not left idle. A number of parameters related to the characteristics of the job-shop problems are of importance in simulation.

(i) Scheduling rules, known also as dispatching, priority or loading rules.

There are many rules encountered in the literature and in practical applications. Some are extremely simple and some quite sophisticated. Reviews of these rules can be found in Conway et al (1967) and in Panwalkar and Iskander (1977).

Commonly used loading rules can be classified as local or global. Local rules use information available locally at a particular machine centre, where the decision will be implemented, regardless of information on the rest of the job-shop. Global rules use both local information and from other sources, requiring therefore a more elaborate information system. The loading rules can be classified also as static, where the priority of a job does not change over time, and as dynamic where the priority is a function of time. Examples of simple rules are given below.

RANDOM: random selection of job to be processed first.

FIFO : first in, first out.

LIFO : last in, first out.

SI : shorter imminent operation.

SI* : SI with a maximum permitted waiting time in queue.

COVERT: c/t, according to descending value of waiting cost to processing time.

EDD : Earliest (minimum) due date first.

LROPS : least remaining operations.

Many studies have been devoted to evaluating and comparing the rules in various job-shop environments. The SI rule came out as a very efficient rule and very simple to implement (Conway et al 1967, Jones, 1973, Rochette and Sadowski, 1976). It produces low waiting (or flow) times but has the drawback that long operations tend to be left unprocessed for very long periods of

time. To counterbalance this disadvantage a 'truncation' has been superimposed on the SI rule, so that when the waiting time of a job in a queue exceeds a prespecified amount of time, the job is given priority to all other ordinary jobs waiting in the same queue (Conway et al 1967, Buffa and Taubert 1972, Eilon and Cotterill, 1968, Oral and Malouin, 1973,Eilon and Chowdhury, 1974).

More variations of this basic idea can be found in Hottenstein (1970) and Jones (1973). Aggarwal and McCarl (1974) have tried a composite cost-based rule and Hershauer and Ebert (1975) have tried another form of a composite priority rule, comparing them with standard rules. A new composite rule has been designed and tested in this study (in Section 7.4).

(ii) Arrival patterns and job-shop loading.

There are many conceivable models describing arrivals in job-shops, where arrivals take place singly or in batches. Single arrivals have been used in a number of simulation studies, where the interarrival times were assumed to be from a negative exponential or other distribution (Conway, 1965, Nelson, 1967, Hottenstein, 1970, Jackson, 1963, Conway and Maxwell, 1962). This model has no practical interest, because single arrivals in real life job-shops are very rare. Batch arrivals are much more realistic, especially when the interarrival times Δt are fixed, corresponding to the industrial practice of receiving orders over a fixed time period and releasing them to the shop daily or every week.

The batch size Q can be variable or fixed, in which case the number of jobs dispatched (but not their characteristics) is deterministic, corresponding to the case where an agreed number

of orders is accepted, determined by an overall target loading of the shop, and where excess orders are refered to subsequent time periods. Eilon and Chowdhury (1975) have studied the relationship of waiting times, for a number of dispatching rules, for deterministic and Poissonian batch sizes, and found that the latter yield a much higher mean waiting time, as one might expect. Elvers (1974) has conducted an extensive study with batch sizes Q determined by sixteen different probability distributions with the same mean, and his conclusions imply that the shape of the distribution is not a significant variable in evaluating the relative effectiveness of dispatching rules. Deterministic batch sizes and interarrival times have been used in this study, since the relative performance of heuristics has been established to be independent of the shape of the distribution.

(iii) Job-shop size.

Baker and Dzielinski (1960) have conducted a study in which they reported that the shop size has no significant effect on the relative performance of the scheduling rules. This conclusion has been confirmed by Conway et al (1967) where it has been suggested that experiments with a job-shop of six machines are adequate to show the complexities that are likely to arise in larger job-shops.

(iv) Due-dates.

The methods by which due-dates are determined have attracted considerable attention on their own. A number of methods for determining due-dates have been investigated by Eilon and Hodgson (1967), Conway et al (1967), Ashour and Vaswani (1972), Holloway and Nelson (1974), Eilon and Chowdhury(1975), Day and Hottenstein (1975). The main conclusion is that some methods can produce

better results for specific scheduling rules and that changes in the parameters involved in defining the due-dates can control effectively the values of the criteria of performance.

(v) Routing of jobs (transfer matrix).

A range of routes is possible in job-shops. In the limiting case of the general job-shop all routes through the machines are equally likely. At the other end of the range, all jobs follow the same path through the machines (flow-shop). The routing is affected by the amount of flexibility possible in a given job-shop. It is possible that a particular operation can be performed by more than one machine, when the most suitable processor is not available or when a job is very urgent. Another type of flexibility is one where the operations sequence of a particular job can be altered, according to some rules (Chowdhury, 1976). These subjects have not attracted considerable attention up to now, and it is felt that 'flexibility' is an open subject for further research.

(vi) Processing times.

The estimates of processing times of operations in a given job-shop can be described by some statistical distribution, with defined average and standard deviation. The actual processing times may be equal to the estimates or they can vary according to some distribution. The effects of using processing times from different distributions or from the same distribution but with a range of variances have not been thoroughly investigated up to now. The sensitivity of simulation results to errors in estimating processing times is only partially known, for cases of minor differences between expected and actual times. It has been felt

that the structure of the processing times is a topic of special interest and thus it has been investigated in depth in Section 7.3.

(vii) Machine centres with identical processors in parallel.

This is another area that has not attracted attention and the results obtained in this study are discussed in Section 7.5.

7.2 Design of the experiment

The above mentioned studies have been carried out by means of a discrete event simulation in a digital computer(see Schmidt & Taylor,1970,Fishman,1973 on simulation). The model used comprises 5 work-centres with one machine each. The arrivals of deterministic batches of jobs (batch sizes of 1,10 and 35 jobs) in the shop take place at regular intervals, depending on the overall job-shop load (load factor ranging from 0.6 to 0.95). As soon as a batch of jobs is dispatched to the shop, due dates are assigned to the jobs, allowing for in-process waiting up to the total processing time they require, i.e. $D_i = R_i + 2\sum_{j=1}^{m} P_{ij}$ for all jobs i in the batch.

The estimated processing times are from the family of Erlang distribution and equal to the actual ones (the effect of actual processing times differing from the estimated, will be discussed in Section 7.3.4). A range of values of the Erlang parameter k has been considered: 1,2,3,4,5,9,15, 30,∞. The smaller values (data with high variance) represent cases of diversified product lines, consisting of items which vary extensively in their demand pattern of facilities. The high values of Erlang k correspond to homogeneous operations in length.
The routing of jobs in the job-shop is assumed to be completely random- all routes through the machines/centres are equally likely-and inflexible, in the sense that a certain route may not be altered in any case. The number of operations per job is randomly distributed between 1 and 5 (uniform rectangular distribution).

The overall load factor of the shop is the ratio of the total machine time required to process a set of jobs over the total available machine time (for the one machine system it is $\rho = \lambda_a/\lambda_s$). Defining a desired load ratio is equivalent to determining the batch interarrival times Δt.

The real value of the load ratio can be calculated at the completion of the experiment and it might be slightly different from the desired load ratio. In this study, the term 'load ratio' ρ will be used for the desired value only and should be seen as equivalent to a specific value of batch interarrival times.

The main measure of performance used is the average waiting time i.e. the first moment of the distribution of job waiting times. A lot of additional information has been collected in the form of histograms, averages and variances of waiting (flow) time, lateness and queue size.

A number of experiments has been carried out in this study with the model described above, using routines of a general job-shop simulation package (Pace, 1969, JSS, 1972 and Eilon, 1973).

There is a number of sources of possible error associated with any simulation experiment, and with the collection of information. The 'run-in' period (until a steady state is reached) may introduce some error, which can be counterbalanced by deleting a number of observations at the beginning of the experiment. The length of the transient period that should be deleted can be calculated as in Wagner (1972) or Fishman (1973) or found empirically by collecting relevant information in a number of trials. Another source of error is due to the timing of the collection of information. Values are added in histograms when a job is completed only, and some error is introduced at the termination of the experiment where jobs near completion are ignored (run-out period). Both types of error can become negligible, if a large sample of jobs is simulated.

Another source of error is due to the auto-correlation that exists between the values recorded in successive observations. It is though not uncommon to ignore these effects, when one is interested in comparing

the same measure of performance for a number of dispatching rules or load factors (Pace, 1969, Chowdhury, 1976). Chowdhury (1976) in his study, using the same job-shop simulation computer package (JSS, 1972), has calculated confidence intervals for the results and found them satisfactorily low for runs between 5000 and 10,000 jobs. In this study, after some preliminary empirical investigation the sample size N has been taken equal to 10,000 jobs (one run only), large enough to absorb the run-in and run-out periods effects and to offset the effects of autocorrelated sample values for values of load factor up to $\rho = 0.9$.

7.3 Sensitivity of simulation results to the processing times structure

The objective of this study is to analyse the effects of different data structures for processing times on the performance of a number of scheduling rules. As additional parameters, the overall loading of the job-shop, and the size of the batches have been used. It is accepted that the size of the shop does not affect seriously the performance of the rules (Baker and Dzielinski, 1960) and that the effects of having arrivals with stochastic (Poissonian) batch size instead of deterministic need not be considered here, since they have already been studied in Eilon and Chowdhury (1975).

7.3.1. Effects of changes in the distribution function

For a given distribution function f(x) describing the processing times P_{ij} in a job-shop, changes in the expected value E (P_{ij}) will not affect the relative performance of scheduling rules. A change in the expected value will result in a proportional change of waiting times, flow times etc, but ratios like $\bar{W}/E(P)$ and $\bar{F}/E(P)$ will remain the same.

It is expected that distributions with the same first and second moments, i.e. with the same mean value and standard deviation, will not produce significantly different results in simulation. This hypothesis is derived from the fact that, in a Taylor expansion of the function, the derivatives of order higher than two play a minor role only (Appendix E). Thus, if two functions have the same values of first and second moments (μ,σ), their difference is expected to be insignificant for simulation purposes.

This hypothesis has been tested by comparing results from Erlang

and Normal distributions, with the scheduling rules FIFO, SI, SI*, load factor $\rho = 0.8$, deterministic batch arrivals with batch size Q=35, mean processing time μ=20 time units, and 10,000 jobs with random routing. The rule SI* is used with truncation defined by a control parameter U. Jobs are given ordinary priority as long as the quantity $D-(t+P_r) - U$ is positive (where t is the current time and P_r is the expected remaining processing time). If this quantity becomes negative they are given top priority in the queue. The value of U has some effect on the average waiting time, as can be seen from Serghiou (1973) and from the following results of a trial with normally distributed processing times ($\mu = 20$, $\sigma = 6$, $\rho = 0.8$).

U	0	40	80	120	160
$\bar{W}$	204	206	212	214	214
σ	159	169	165	165	165

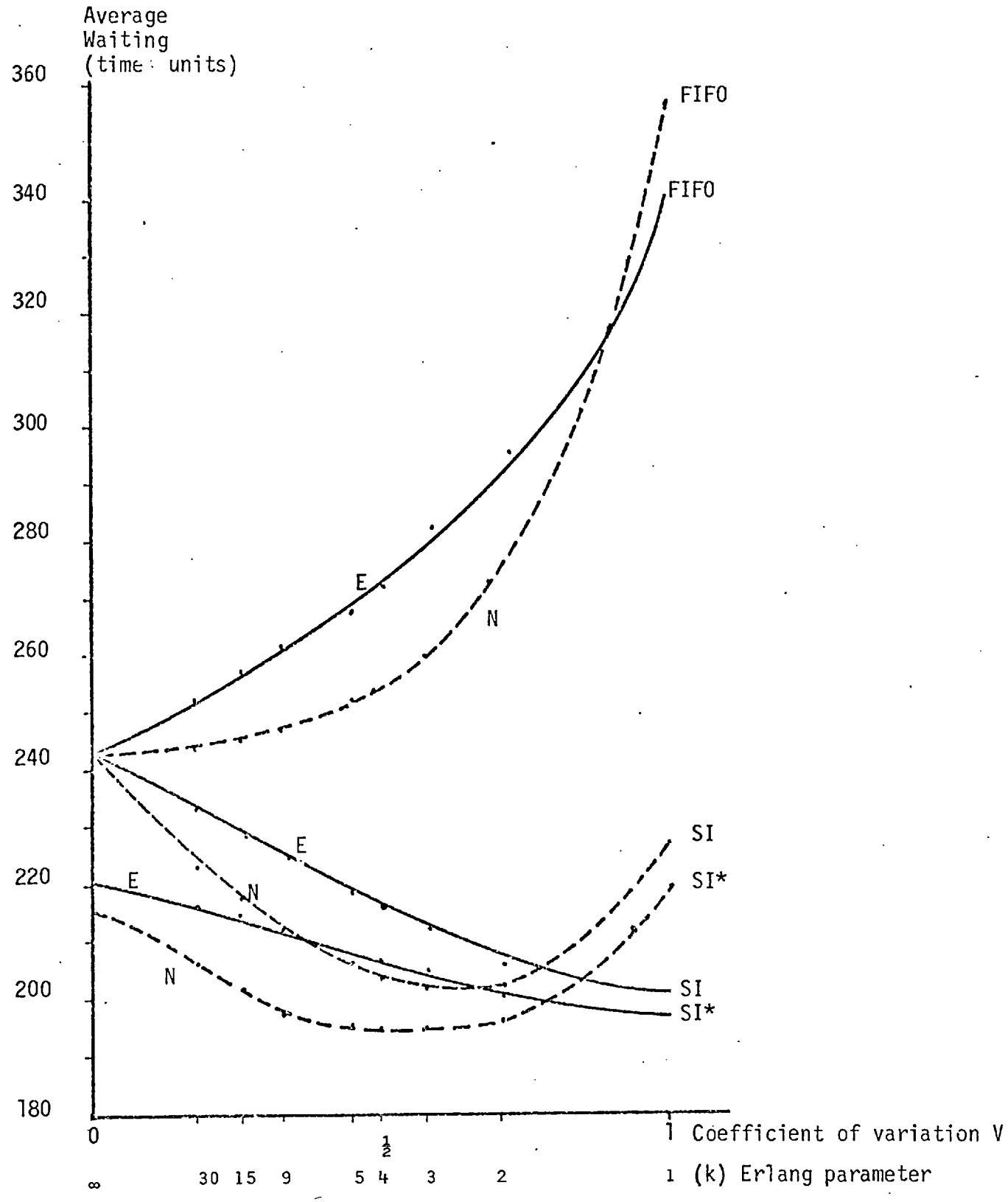
In this study the value of the control parameter U has been taken equal to zero. Table 7.1 below shows the values of the variance parameters used for average processing time $\mu = 20$.

Table 7.1 Coefficients of variation for Erlang and Normal distributions

Erlang parameter k	$V=k^{-1/2}$	Normal, σ
1	1.000	20.000
2	0.707	14.142
3	0.577	11.547
4	0.500	10.000
5	0.447	8.944
9	0.333	6.667
15	0.258	5.164
30	0.182	3.651
100	0.100	2.000
∞	0	0

The results of this simulation experiment are given in Tables E1 - E4 in Appendix E and summarised in Figure 7.3 on the following page.

Figure 7.3 Comparison of average waiting times with Erlang and Normal distribution of processing times (Q=35, ρ=0.8)



As can be seen from Figure 7.3 (and from Tables E1 and E2 in Appendix E), the average waiting times resulting from Erlang and Normal distribution of processing times are very close, except for the case of very high variance (V = 1). The simulation results, for all scheduling rules, show a maximum difference of 5%, and usually lower. This verifies the hypothesis that the important descriptors of the processing times distribution are the mean and standard deviation of the distribution of processing times and not the higher order moments. The larger difference observed in the case of very high variance (k = 1, V = 1 and σ = 20) should be expected, because of the method used for determining processing times; all processing times less than 1 (or negatives) are set equal to 1. This distortion produces a Normal-like distribution with real σ less than 20 and this is reflected in the localised irregularity of the simulation results at V = 1.

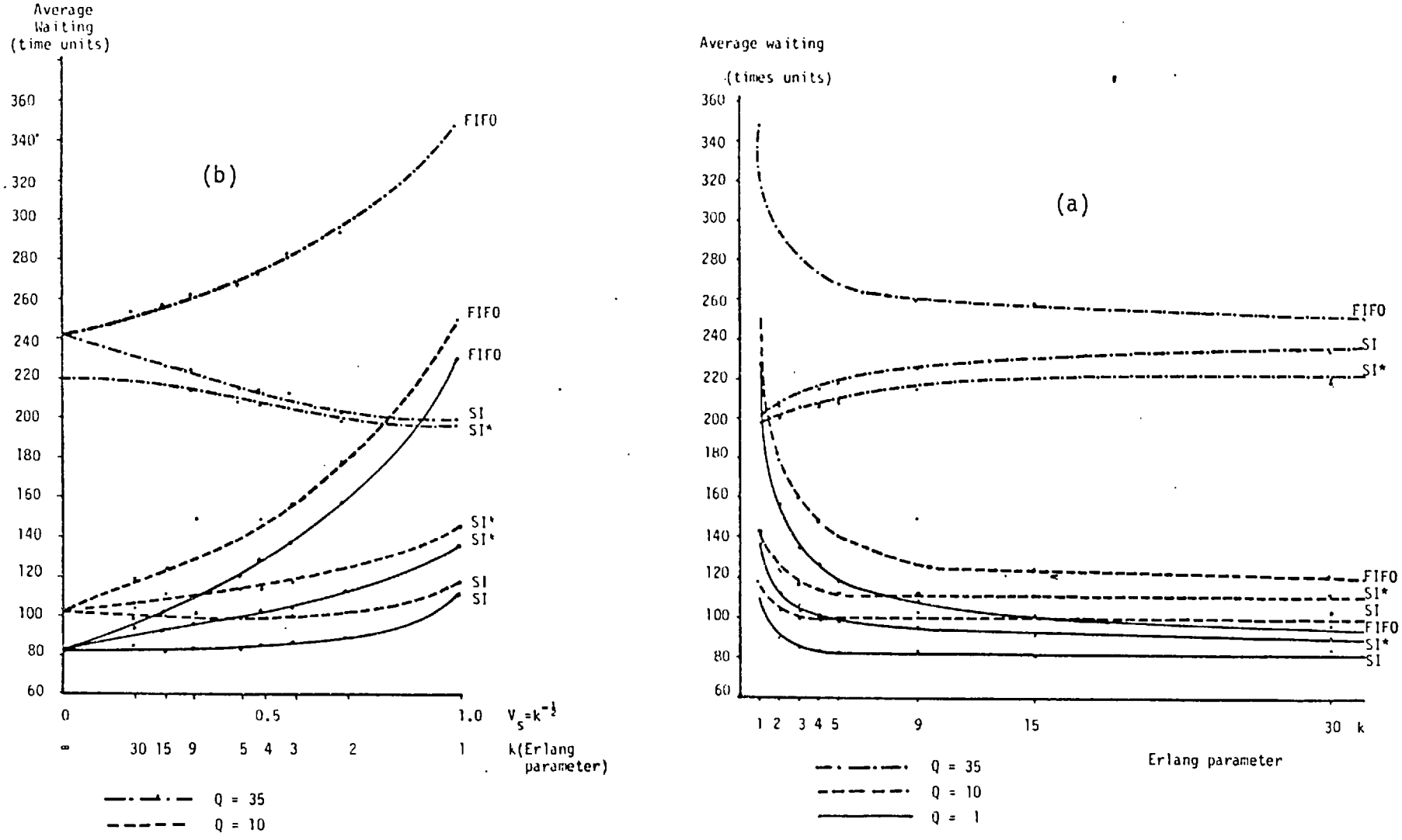
These preliminary results indicate clearly a relationship between the variance of a distribution and the performance of heuristics in simulation. This has been the subject of an extensive experiment, discussed on the following pages.

7.3.2. Effects of changes of the variance of processing times on the performance of scheduling rules at fixed load factor.

It has already been established in the study of static and deterministic problems in Chapter 4, that the variance of processing times has some impact on the performance of heuristic scheduling rules. This effect has been studied in more depth with simulation for some basic rules (FIFO, SI, SI*). The study has been based on the Erlang family of distributions, since it has been shown above that other distributions with the same mean and variance would not produce significantly different results. It has been

thought necessary to include two additional parameters in this experiment, the desired overall load factor for the job-shop and the batch size of the deterministic arrivals. The effect of changes in the variance (or in the Erlang parameter k) in conjunction with changes in the batch size, at constant load factor $\rho = 0.8$ have been investigated at Q=1, 10 and 35, with one run of 10,000 jobs each. The load factor has been set to $\rho = 0.8$ as a realistic high utilisation target value, avoiding the instability of near congestion cases ($\rho \to 1$). The batch size of Q = 1 (single arrivals) has been used as a limiting case with theoretical interest, Q = 35 has been selected as a case of large batch arrivals, in accordance with the job-set n = 35 used in static and deterministic scheduling (Chapters 4 and 5), while Q = 10 has been set as an arbitrary intermediate value. The results are given in Tables E2, E3 and E4 in Appendix E and summarised in charts. The average waiting time with the dispatching rules FIFO, SI, SI* has been plotted against the value of k in Figure 7.4(a) (next page). The use of the value of k on the horizontal axis has the drawback that $k = \infty$ cannot be included in the graph. The coefficient of variation $V = k^{-1/2}$ is more appropriate for plotting the average waiting, as can be seen in Figure 7.4(b) on the following page. As can be seen from Figure 7.4, for the average waiting time ($\bar{W}$ equivalent to $\bar{F}$) in smaller batch sizes (Q = 1 and Q = 10) SI performs better than SI* which performs better than FIFO, for all values of k, and the $\bar{W}$ performance of each rule improves with increasing k (decreasing V). In larger batches (Q = 35) FIFO is again the poorest and its $\bar{W}$ performance improves with decreasing V. There is though a reversal of performance for SI and SI*. The scheduling rule SI* gives always the lowest average waiting. Its performance however deteriorates with decreasing V.

Figure 7.4 Average waiting for Erlang processing times (k=1,2,....,∞) at ρ=0.8.



The reason for this behaviour is that in small batches, an increase in the variance of processing times results in significant fluctuations in the loading, i.e. there are periods where queues of jobs are built up in front of a certain machine and periods where no jobs are available for processing. This unutilised capacity cannot be balanced because jobs arrive singly or in small batches and no pool of unprocessed jobs exists. The greater the variance, the greater these fluctuations and the resulting delays, as can be illustrated better at the extreme case of a one-machine job-shop. The difference of the average waiting in the extreme cases of E_1 and E_∞ can be described in terms of the ratio $\bar{W}_M/\bar{W}_D$, taking the following values.

Ratio $\bar{W}_M / \bar{W}_D$

Q	FIFO	SI	SI*
1	2.9	1.4	1.6
10	2.6	1.2	1.4
35	1.4	0.8	0.9

The values of this ratio show clearly that there is a reversal of behaviour of SI and SI* for large batches, where the average waiting of highly diversified job-sets is lower than for homogeneous product lines. This is due to the fact that at high variance, SI and SI* are able to discriminate among jobs and exploit the increased availability of work load for processing, due to the larger batch size, by giving priority to shorter operations and therefore reducing $\bar{W}$,while at low variance, they behave more like FIFO. In fact for E_∞ (V = 0) FIFO and SI are identical. In that case SI* produces a lower $\bar{W}$ because of the truncation, whereby excessive delays of some jobs are avoided. In other words, with larger batches the fluctuation in processing times becomes an

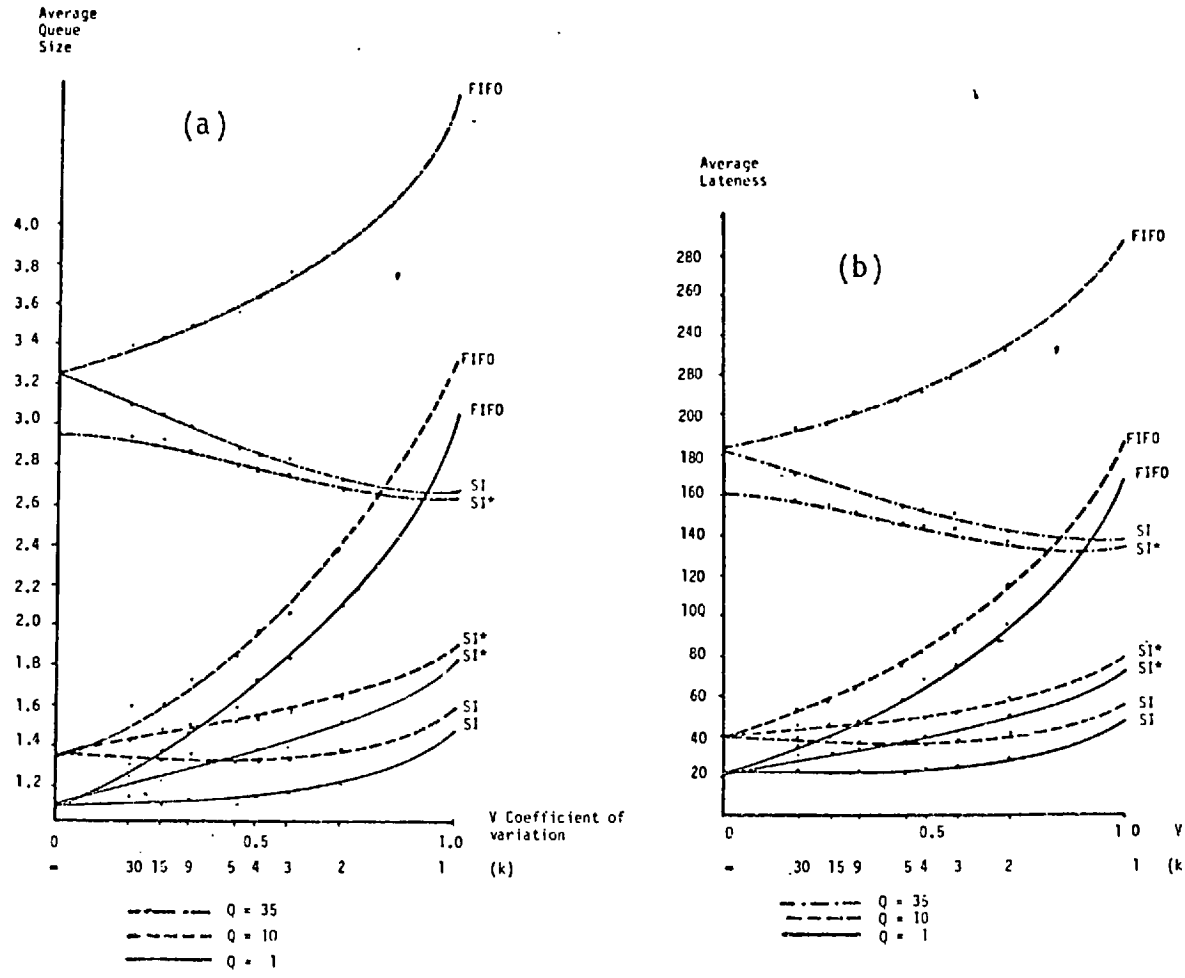
advantage when SI and SI* are used. The fact that SI* produces lower $\bar{W}$ than SI in larger batch sizes but not in small ones, can be explained again by a similar reasoning. By having this truncation (cut-off time) in small batches and operating it in very limited pools (queues)of jobs, long jobs are processed early enough to block the machines, impeding them processing the shorter operations of subsequent batch arrivals. Thus queues from these job arrivals are formed and the advantages of this truncation are lost. At larger queues there are usually enough jobs waiting to allow the SI part of the rule to bring about reduced $\bar{W}$. It is worth contrasting here the above results on single-pass non-delay, scheduling heuristics FIFO and SI to those obtained in Ch.4 on single-pass active scheduling heuristics ECT, FCFS and SPT. The active rule FCFS and the non-delay FIFO, as single pass heuristics, are the same. The use of the non-delay rule SI (choose the job with the shortest imminent operation) has the effect of processing first the job with the earliest completion time. Thus, it resembles, in principle, to the active rule ECT (earliest completion time). This resemblance is reflected in the good performance of ECT for the average waiting time (Section 4.3, Table 4.5), in agreement with the good performance of SI in the simulation experiment carried out in this section. The active rule SPT (shortest processing time) is not acting like SI, because the point t in time at which the scheduling (branching) decision is taken and t-dt, at which it is retrospectively implemented are not the same. In the non-delay case, the decision taken by SI is implemented at once. Thus one should not expect similar performance for SI and SPT.

Another point worth mentioning is that the total makespan values (C_{max}) obtained in this section by simulating 10,000 jobs, using

the dispatching rules FIFO, SI and SI* are not significantly different. This could be attributed to the large sample size, in agreement with the results of static and deterministic scheduling (Section 4.3) where for increasing number of jobs, the bracket for the optimal solution BOS becomes smaller, especially in problems with processing times of low-variance.

The average queue length results (Table E3) are, as expected, related to the average waiting (in the single server queue, expected queue length = mean arrival rate x expected waiting time). The average queue length has been plotted against the coefficient of variation V_s of the processing times for Q = 1, 10 and 35 in Figure 7.5 (a) below. As can be seen from comparison of Figures 7.4 (b) and 7.5 (a), the patterns are almost identical.

Figure 7.5 Mean queue size and mean lateness as a function of the coefficient of variation of service times.



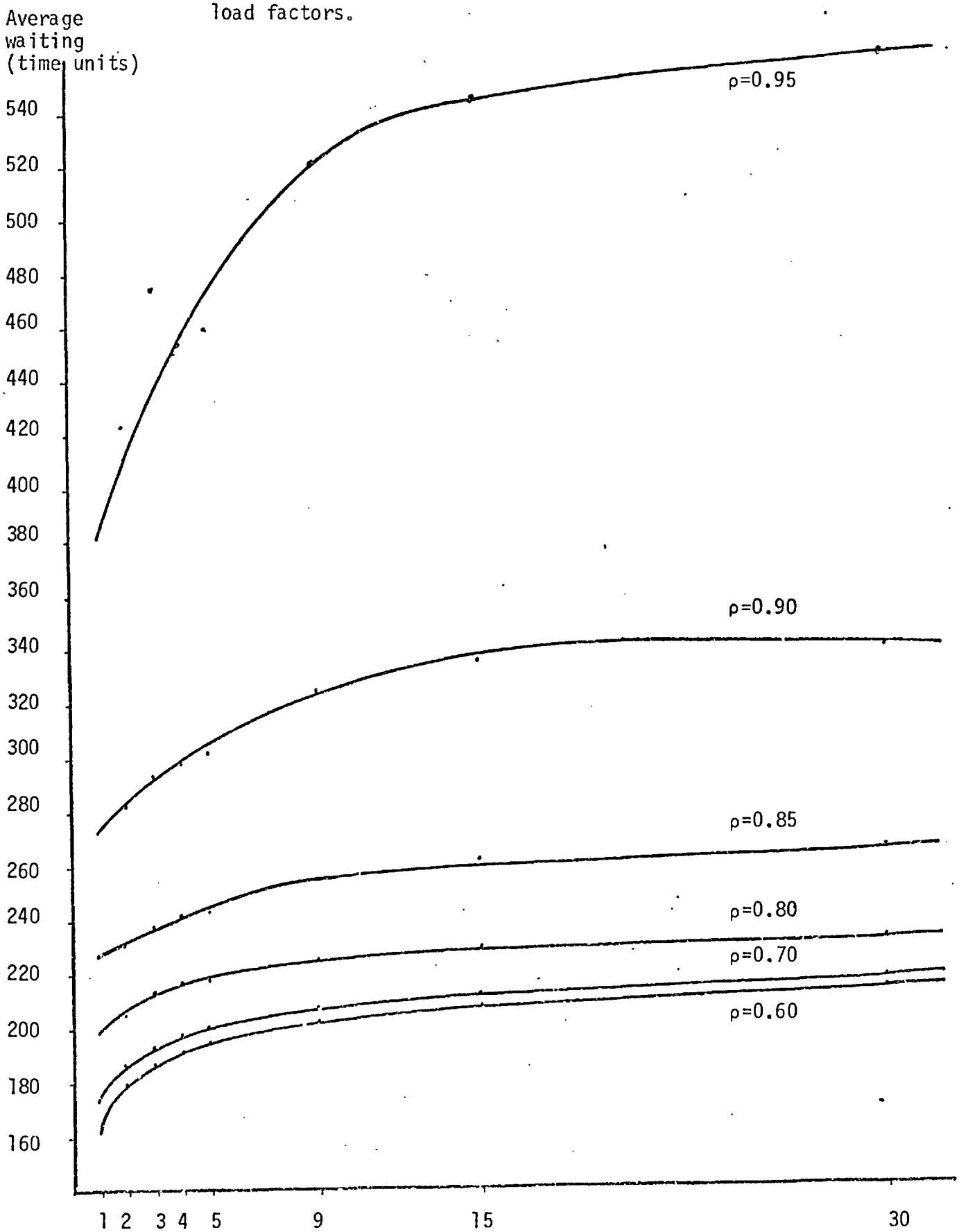
The values of average lateness have also been plotted against V_s, in Figure 7.5 (b), and again, the pattern is the same as that of average waiting times in Figure 7.4 (b). This result was predictable, because of the method used for determining due dates ($D_i = R_i + 2P_i$), since lateness is defined as $L_i = D_i - P_i - W_i = R_i + P_i - W_i$ and the criteria of performance $\bar{L}$ and $\bar{W}$ are equivalent (Section 1.4).

7.3.3. Effects of changes of processing times variance with variable load factor.

The effect of changes in the variance of processing times and in the job-shop load factor on the performance of the SI rule have been studied with an experiment where deterministic arrivals take place in batches of Q = 35 and Q = 10. The Erlang parameter k takes values from k = 1 to k = ∞ and the load factor takes values from $\rho = 0.6$ to $\rho = 0.95$. The average waiting time values are given in Table E2 in Appendix E and summarised in Figure 7.6 on the following page.

These results agree with the ones discussed in Section 7.3.2. For large batches (Q = 35) and given load factor ρ, the average waiting in the shop decreases with increasing variance of processing times. As expected, higher load ratios result in larger $\bar{W}$, created by the increasingly congested state of the job-shop. For load ratio approaching 1 ($\rho = 0.95$) results are unstable, varying widely for different seeds of random numbers. This high variance is probably due to autocorrelation which becomes increasingly important for $\rho \to 1$. The pattern of change for different values of the Erlang parameter k is the same for all load factors used. Results for smaller batch sizes (Q = 10) are again in agreement

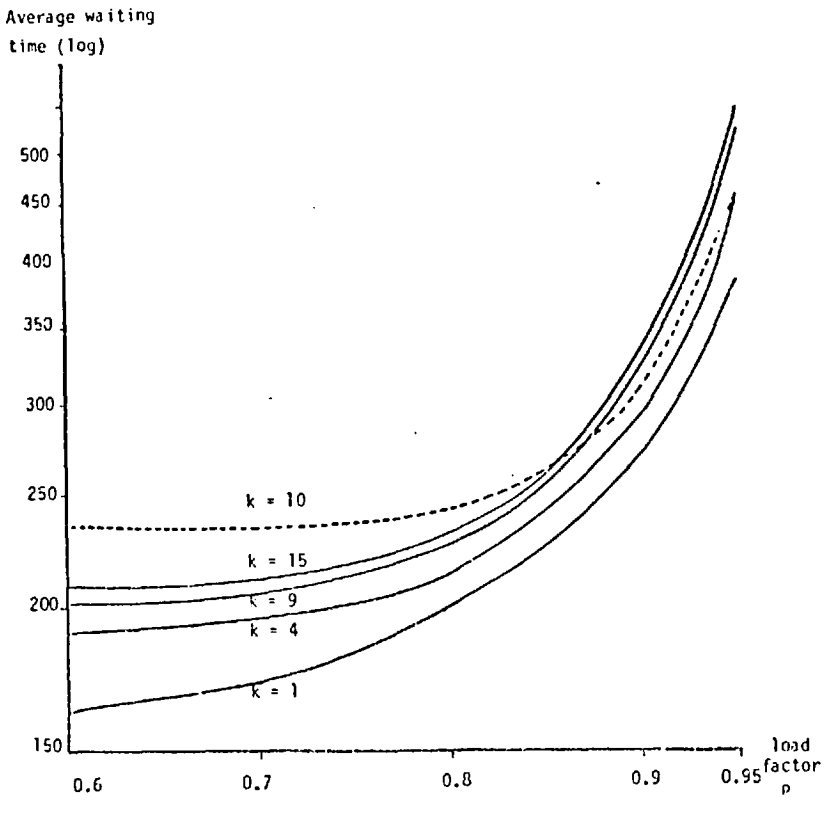
Figure 7.6 Average waiting time for Erlang processing times at various load factors.



with the preceding findings. For load factors up to $\rho = 0.85$ the average waiting increases with V, as explained in Section 7.3.2. At load ratios greater than $\rho = 0.9$, where the shop is nearly congested, the availability of more jobs in the queues allows the SI heuristic rule to discriminate sufficiently between jobs and reduce $\bar{W}$ for increasing V, in the same way as it does it for larger batches. The similarity lies in the near congestion conditions that are created either by the high load ratio or temporarily by the arrivals of large batches. The implication is that SI performs better with high variance data only under temporary or permanent near-congestion conditions. The performance of SI* under varying load factor conditions is expected to be similar to SI. The rule FIFO is also affected by varying ρ, but the pattern of change for varying V is the same as has been verified by the results in Table E2, in Appendix E.

For a given type of distribution E_k, the effect of increasing the load factor is illustrated below in Figure 7.7, where the average

Figure 7.7 Average waiting time for E_k processing times as a function of theload factor.



waiting times of SI (Q = 35) are plotted against the load factor, for fixed k. The irregularity that arises with the line $k = \infty$, is probably due to the higher simulation error for $\rho \to 1$, and to the method of generating Erlang variates for $k \neq \infty$.

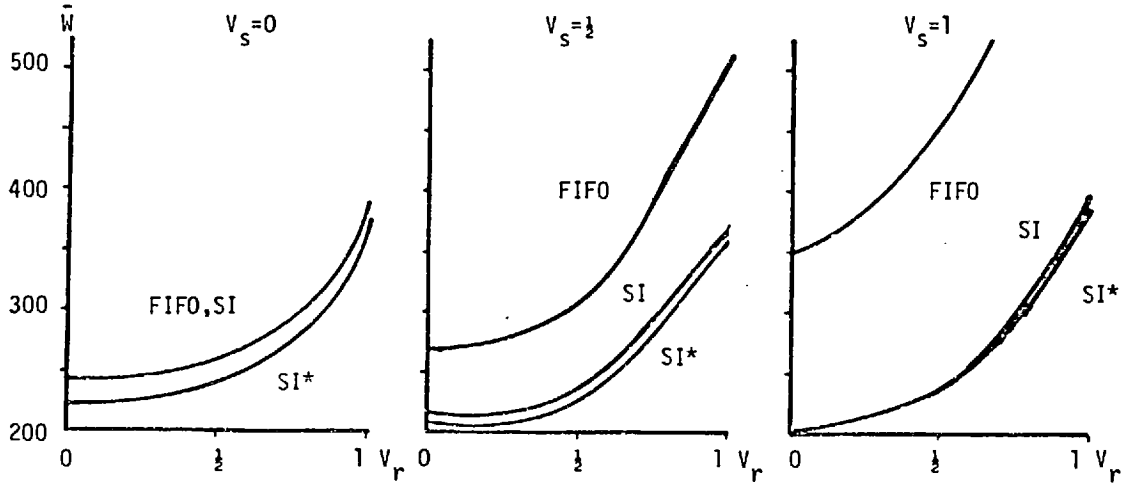
7.3.4 Effects of inaccuracy of processing times estimates

The usual assumption found in job-shop scheduling studies is that the processing times are known in advance. This is not the case in real life problems, where estimates are known but the actual times differ. The performance of the heuristics is not seriously affected, when minor differences occur. In this study, a full investigation of the performance of FIFO, SI, SI* has been carried out where the actual processing times have been taken to be distributed about the estimated, with coefficients of variation V_r ranging from 0 to 1 at Q = 35, ρ = 0.8. The results are given in Table E5 in Appendix E and summarised in Figure 7.8 below.

Figure 7.8 Effects of inaccuracy of processing times estimates.

V_s: coefficient of variation of estimates of processing times

V_r: coefficient of variation of real (actual) processing times about the estimates.



These figures show that the performance of the rules is affected by the variations from the estimated times, deteriorating almost exponentially with increasing variance of the actual times. This is due to the unpredictability of the real processing times, which renders rules based solely or partially on processing times information useless. As a consequence, the scheduling decisions produce large imbalances in the loading of the machines, and thus substantial increases in $\bar{W}$.

7.4 Evaluation of a new composite global scheduling rule

7.4.1 Description of the scheduling rule

The main idea in suggesting a composite priority rule is to relate desired results with decision variables, thus allowing some form of control of the job-shop state. A global dynamic rule has been designed in this study, where the priorities have been defined as functions of attributes related to some measures of performance. The objectives that have been thought to be appropriate for all industrial problems are: meeting due-dates, minimising work-in-progress, maximising utilisation of facilities. These criteria are from three different groups of measures of performance (Section 1.3) which are independent and usually conflicting. One can improve the utilisation (and increase the loading) of facilities at the expense of work-in-progress or due-dates. The numerical values that each of them can take are not directly comparable and if an overall evaluation of the performance of a system is required, one has to link them in the form of a composite criterion of performance (cost or utility function) with subjectively defined coefficients. Such a composite criterion of performance requires a composite decision rule, taking into account the individual measures, since simple rules like FIFO, SI etc. are unlikely to perform well for all of them.

The rule suggested here operates on each queue by calculating a priority index $p = \sum_{j=1}^{N} w_j p_j$, where w_j is a weighting coefficient ($\sum_{j=1}^{N} w_j = 100$), p_j is the priority attribute related to criterion j and N is the number of criteria taken into account.

A rule with three attributes is described below:

$$p = w_1 p_1 + w_2 p_2 + w_3 p_3$$

The priority attribute p_1 is related to due dates and slack.

$$p_1 = (1 - \delta)(D_i - T - P_{ri})^{-x} + \delta (D_i - T - P_{ri})^{x}$$

where D_i is the due date of job i

T is the current time

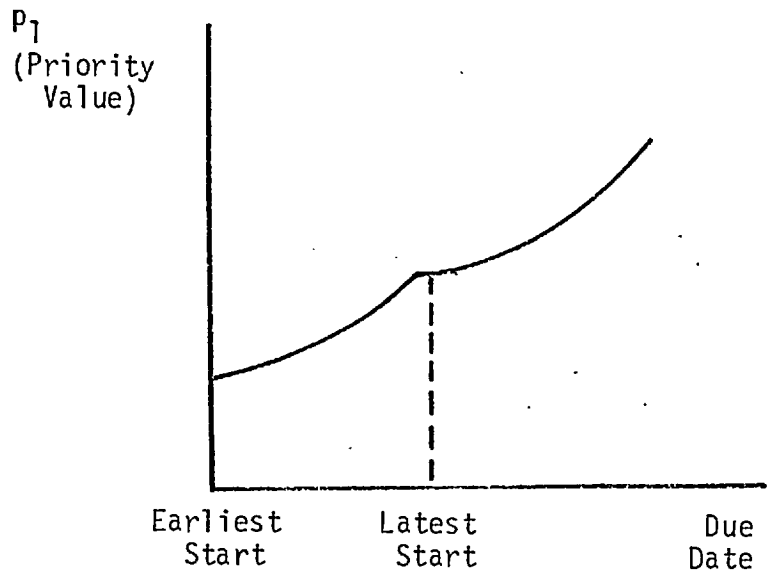
P_{ri} is the remaining processing of job i

$\delta = 1$ if the job is late, or else 0.

$x > 0$

When the quantity $(D_i - T - P_{ri})$, usually named slack, is a large positive number, the risk of the job becoming late is small, while it is high for small values of slack. The priority calculated in this way can be plotted as in Figure 7.9 below.

Figure 7.9 Priority values in a composite scheduling rule related to due-dates.



The second priority attribute p_2 is related to work-in-progress, or waiting in queues for processing.

$$p_2 = \{ (T - R_i) / (P_i - P_{ri}) \}^y$$

where R_i: arrival time of job i

P_i: total processing time of job i

$y > 0$

With p_2, the idea is to give high priority to jobs that have been in the shop longer than some target value, e.g. jobs whose flow time exceeds the total processing up to the current moment, multiplied by a factor, using y to adjust the values numerically.

The basic idea can be used also if more elaborate calculations are desired for the work-in-progress. A more exact method should take into account not only the time spent waiting in queues but also the value of the item waiting at every stage, based on the cost of materials and on the added value from processing at different facilities. The third priority attribute p_3 is based on a look-ahead (global) procedure, taking into account the size of the queue at the machine of the subsequent operation of each job. This can be extended to more than one subsequent operations if required. The amount of processing known to be waiting for the machine of the subsequent operation is L_m, and $p_3 = z\ L_m$, where $z > 0$ is a coefficient for adjusting the values of p_3.

7.4.2 Calibration and evaluation of the composite rule

This scheduling rule can take many forms, depending on the values of the parameters and weighting coefficients. The performance of the rule for a particular measure of performance, simple or composite, depends on these values. The objective of this experiment has been to demonstrate that it is possible to define suitable values of these coefficients, allowing the composite rule to perform better than the simple rules of the preceding section.

Ideally the work should have been carried out for a composite criterion of performance, but in order to avoid the complications of using more arbitrary parameters, the average waiting time has been used.

The $\bar{W}$ performance of the composite heuristic has been investigated for fixed values of x,y,z and for a range of values of the coefficients w_i. The parameters x,y,z have been defined so that the related priority attributes p_1, p_2, p_3 take values within an arbitrary range (between 0 and 10, with 0 and 10 corresponding to extreme cases and with a desired target value of 1). For deterministic batch arrivals of Q = 1, 10 and 35, load factor ρ = 0.6 and 0.8, the following sets of coefficients have been used:

w_1	w_2	w_3
0	0	100
50	0	50
100	0	0
0	50	50
0	100	0
30	30	40
30	50	20
50	30	20
33	1	66

In each of these experiments, the average waiting time obtained varied, depending on the combination of weighting coefficients used. Although an 'optimal' set of w_1, w_2, w_3 has always been located, it has not been possible to establish a pattern for varying batch size, load factor and variance of processing times. The consequence of this lack of overriding pattern is that for each special case of the problem (i.e. for each model, and specific values of Q, ρ and Erlang parameter k) a separate

'calibration' is required in order to find the best combination of coefficients for a given measure of performance.

Some tests have been carried out comparing the $\bar{W}$ performance of the global composite rule with SI, SI*, FIFO using fixed values of x,y,z with the results shown in Table 7.2 below (for $\rho = 0.8$).

Table 7.2 Comparison of the new composite rule with simple dispatching rules.

Q	E_k	w_1	w_2	Composite	$\bar{W}$ min (FIFO, SI, SI*)
1	9	33	1	89	86
1	∞	0	50	72	82
10	9	0	0	104	103
10	∞	0	0	83	101
35	9	33	1	208	219
35	∞	50	0	190	240

These results, as expected, indicate that for the problems investigated (and a given criterion of performance), there is a set of parameters (coefficients) with which the global rule gives better results than FIFO, SI, SI*. For Q = 35, $\rho = 0.8$ and Erlang estimated processing times the performance of the composite rule, relatively to FIFO, SI, SI* was found to be the same, even when the actual (real) processing times were assumed to be distributed about the estimated with V = 0.5 and 1.0.

It is to be expected that for any job-shop environment (model) and any criterion of performance, there is a set of x,y,z,w_1,w_2 and w_3 for which the global composite dispatching rule will perform better than simple ones, like FIFO, SI, SI*. A problem with this composite rule, which was predictable up to a certain extent, is the computational costs associated with the search for a good

combination of coefficient values. The cost of investigating fully such a rule in abstract terms, over a large range of job-shop environments (models) will be high. This cost though will be reduced when the job-shop and job-set (work load) characteristics are specified, and such a search may become acceptable in relation to a specific problem. The object of carrying out this work should be seen therefore as a demonstration of the potential of the composite rule not in an abstract theoretical case, but in a specific job-shop. The implementation of such a rule should be possible where an on-line data collection and production control system is available. Thus the work of this section should be seen as an attempt to adapt a scheduling system to a real life job-shop environment.

7.5 Job-shops with identical machines in parallel

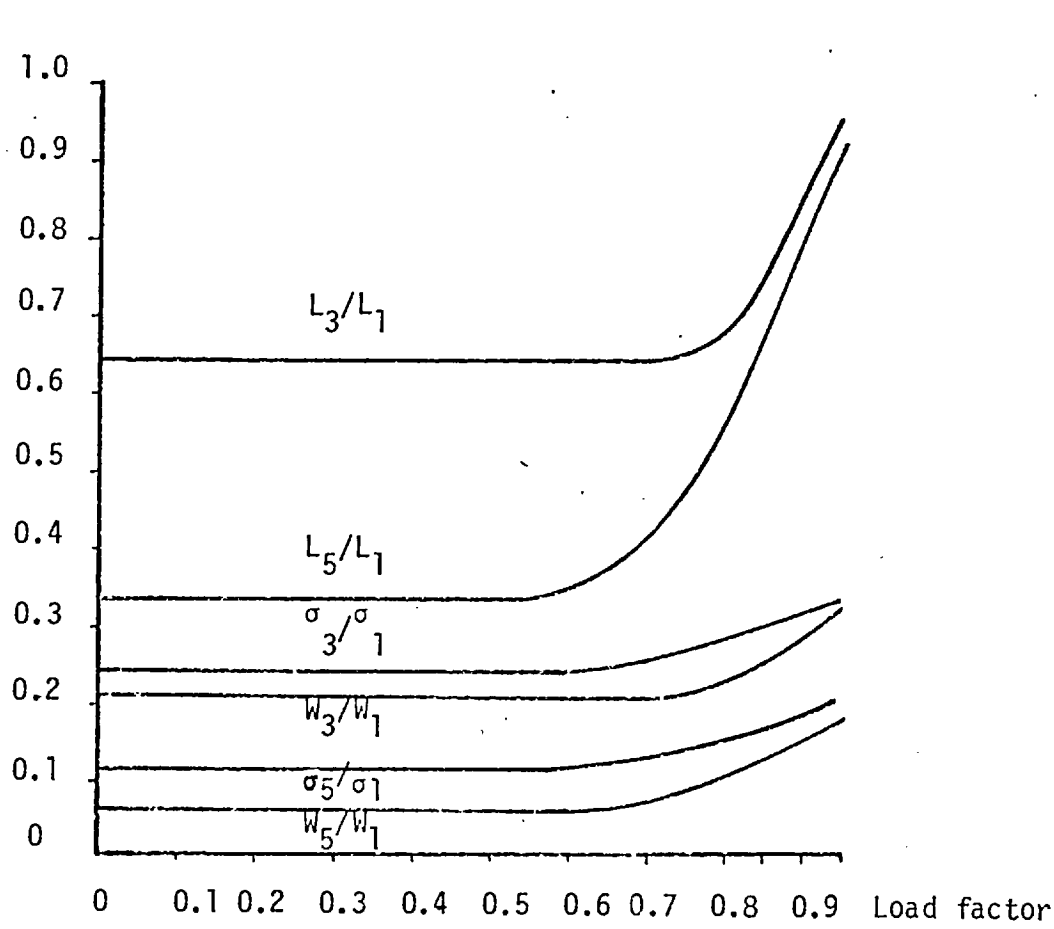
Analytical solutions to job-shop problems with m-machine centres and more than one machine in each work-centre are practically impossible. The obvious method available for dealing with this extra degree of complexity is simulation. A new approach, based on the principle of similarity, is suggested here as an alternative to full-scale simulation experiments. In this new method it is attempted to relate the results (e.g. average waiting or queue length) of single-machine and m-machine per work-centre, otherwise identical job-shops, by establishing some model predicting the performance of the one from the other. In the one-stage queueing systems with one or more identical servers in parallel, and in networks of queues like those used in Section 7.3, these quantities have been established to be functions of the load factor. Thus it is to be expected that ratios of waiting times for two different numbers of servers in parallel are also functions of the load factor.

A limited experiment has been carried out with deterministic arrivals of Q=35, E_∞ processing times and load factors ranging from ρ=0.1 to ρ=0.9, for the cases of 3 and 5 identical machines in parallel in each of the 5 work-centres of the model described in Section 7.2.

The ratios of the results for Average Queue Size, $\bar{W}$, and σ_W of the cases with 3 and 5 machines over the single-machine are given in Table E7 of Appendix E and plotted against the load factor in Figure 7.10 (next page).

These results show that the ratios of average waiting time, standard deviation of waiting time and average queue length are constant for load factor values ranging from ρ=0.1 to ρ=0.7. In this range of values, with these results it is possible to predict the effects of changing the number of machines in parallel on waiting times (mean and standard

Figure 7.10 Ratios of simulation results from models with r and r' machines in parallel.



deviation) and average queue length, without resorting to additional simulation experiments. It is also possible to cover the near congestion cases, by defining these ratios as functions of ρ .

The practical implication of this conclusion is that when studying problems with more than one machine in parallel it is possible to use the underlying similarities and avoid a lot of computational effort and cost in simulation. This idea has been the motivation for the work described in the following Section 7.6, where it has been attempted to relate the $\bar{W}$ result of the general job-shop problem to the $\bar{W}$ of some special cases, with queueing theory methodology.

7.6 Approximate formulae for networks of queues

Recent developments in queueing theory have changed the emphasis from calculating analytically the values of average waiting time or average queue size for complex single-stage problems, to deriving approximate expressions (e.g. Kingman 1965, Page 1972) or using published results derived by non-algebraic methods (Prabhu, 1965, and Page, 1972).
Page (1972) has suggested that linear interpolation in the values of these tables gives reasonable approximations to waiting times for cases not included. For the queue $E_k/E_\ell/1$, the average waiting time is approximately

$$\bar{W}(V_a^2,V_s^2)\simeq(1-V_a^2)\ (1-V_s^2)\bar{W}(0,0)+(1-V_a)^2V_s^2\bar{W}(0,1)+V_a^2(1-V_s^2)\bar{W}(1,0)+V_a^2V_s^2\bar{W}(1,1) \tag{1}$$

where $V_a^2=1/k$, $V_s^2=1/\ell$ and $\bar{W}(1,1)$ is the average waiting in the queue M/M/1, $\bar{W}(0,1)$ in D/M/1, $\bar{W}(1,0)$ in M/D/1 and $\bar{W}(0,0)$ in D/D/1.
This form of approximation is satisfactory for the single server queues and in this section a similar approach has been tried for another type of queueing system, the general job-shop described by a network of queues.

The basic idea is to use the network of queues as a single 'black-box' queue, obtain results from simulation experiments and test whether a formula similar to (1) can be verified with a satisfactorily low error. Assuming deterministic arrivals ($V_a^2=0$), the formula is reduced to

$$\bar{W}(0,V_s^2)\simeq(1-V_s^2)\bar{W}(0,0)+V_s^2\bar{W}(0,1) \quad \text{or}$$

$$\bar{W}(0,1/k)\simeq\frac{1}{k}\ \bar{W}(0,1)+(1-\frac{1}{k})\bar{W}(0,0) \tag{2}$$

where $\bar{W}(0,1)$ is the average waiting time for a system with identical negative exponential distributions of the processing times and $\bar{W}(0,0)$

for a completely deterministic system. The results summarised in Fig.7.6 of Section 7.3.3 suggest that such interpolation, based on the above formula, is reasonably accurate. To test this formula, with another method, results obtained as described in the previous sections have been used as the best estimates of average waiting time for the same model, for ρ=0.6 and ρ=0.8, Q=1, Q=10, Q=35 with FIFO, SI, SI*, COMP. The value of $\bar{W}$ from the approximate formula is calculated, using the simulation values for $\bar{W}(0,0)$ and $\bar{W}(0,1)$. The relative error ε% for using this approximation is given in Table E7 of Appendix E and a sample is presented in Table 7.3 below.

Table 7.3 Average waiting time from simulation and approximate formulae

Erlang Parameter k	Dispatching rule: FIFO			SI			SI*		
	W_{sim}	W_{appr}	ε%	W_{sim}	W_{appr}	ε%	W_{sim}	W_{appr}	ε%
1	348	-	-	200	-	-	197	-	-
2	293	295	-0.1	204	221	8.3	199	208	4.5
3	283	278	-1.8	213	228	7.0	207	212	2.4
4	272	269	-1.1	214	232	8.4	206	214	3.9
5	267	264	-1.1	215	234	8.8	208	215	3.4
9	262	255	-2.7	225	238	5.8	215	217	1.1
15	258	250	-3.0	229	240	4.8	219	219	0
30	255	247	-3.3	232	241	3.9	219	219	0
∞	243	-	-	243	-	-	220	-	-

These results suggest that the formula can give a reasonable approximation of the average waiting time, with processing times of any V_s^2.

It is worth noting that the error is consistent in most of the cases, i.e. it is either positive or negative. This suggests that there may

be a systematic reason for these deviations. A number of sources of this error could be considered; insufficiency of the approximate formula, inaccuracy in assuming $V_s^2=1/k$, simulation error in deriving $\bar{W}(0,0)$ or $\bar{W}(0,1)$.

In order to investigate this approach further, an independent method of deriving an approximate formula has been used, based on the single-stage approximate formula developed by Cosmetatos (1974):

$$\frac{\bar{W}(D,G)}{\bar{W}(M,G)} \simeq S\,\frac{\bar{W}(D,M)}{\bar{W}(M,M)} + (1-S)\,\frac{\bar{W}(D,D)}{\bar{W}(M,D)} \qquad (3)$$

For heavy traffic conditions, where $\rho \to 1$

$$\lambda_s \bar{W}\,(GI/G/r:\ GD/\infty/\infty) \to (V_a^2+\rho^2 V_s^2)/2\rho\,(1-\rho)r \qquad (4)$$

Substituting in (3) above for r=1, $(V_a^2)_D=0$, $(V_a^2)_M=1$

$$S \simeq \frac{\{1+\rho^2(V_s^2)_M\}\,\{(V_s^2)_G - (V_s^2)_D\}}{\{1+\rho^2(V_s^2)_G\}\,\{(V_s^2)_M - (V_s^2)_D\}} \qquad (5)$$

For the particular model under consideration with 5 machines, random routes (as described in the previous section), and identical E_k of processing times in all machines :

$$(V_s^2)_G = \frac{2 + 3/k}{9}$$

$$(V_s^2)_D = \frac{2}{9}$$

$$(V_s^2)_M = \frac{5}{9}$$

(See proof in Appendix E)

Substituting $$S = \frac{1}{k} \frac{9 + 5\rho^2}{9 + (2 + \frac{3}{k})\rho^2}$$

and for $\rho \to 1$ $S = 14 / (3+11k)$

Using this value of S and the heavy traffic formula (4) in the original approximate formula (3) of the preceding page:

$$\bar{W}(D,G) \simeq \frac{1}{k} \bar{W}(D,M) + (1-\frac{1}{k}) \bar{W}(D,D) \qquad (6)$$

This results in the same parameters as approximate formula (2) and gives an explanation for a phenomenon already established experimentally, with the simulation results.

The significance of these results lies in the fact that once some approximate formula is constructed for a particular type of problem, then a number of the results of intermediate cases that would be derived from simulation can instead be estimated with reasonable accuracy from the extreme cases of the same problem. The accuracy of this interpolation is thought to be satisfactory, given that the simulation results for the intermediate cases are anyway approximate, and that no exact or analytical method exists for predicting the performance of scheduling rules under different conditions.

CHAPTER 8

CONCLUSIONS AND FURTHER RESEARCH

8.1 Discussion of the context and summary of the thesis

8.2 Suggestions for further research

8.1 Discussion of the context and summary of the thesis

Job-shop scheduling in practice is not a 'free-standing' problem, but a sub-problem of the general one of production and operations management. The question of detailed scheduling arises after obtaining demand forecasts and after deciding on the long and medium term allocation of capacity(resources) over time. The complexity of aggregate planning and the uncertainty and instability inherent in the real life systems do not allow an overall optimisation of the long term capacity planning and of the short term detailed scheduling.

Thus, it is believed, that long-range planning covering the allocation of resources (capacity) to work load (tasks) is possible only with sub-optimal methods based on crude models, simplifications of the real problem, using the experience of past records and forecasts. Following this planning stage only, detailed job-shop scheduling becomes relevant.

Although the meaning of optimality in industrial practice is rather vague, due to the instability and variety of objectives, in theory it is not. Theoretically optimal job-shop scheduling is possible with simplifying assumptions and certainly desirable. However, the potential of exact methods, as has been demonstrated in this thesis, is limited by the complexity of the problem, to very small real life problems. The approximate techniques and methodology presented in this thesis can be used effectively for the sub-problems of the decomposed production scheduling problem. It is envisaged that they can be applied successully over short-term planning horizons, locally at shop-floor level rather than for the problem of overall system optimisation.

It is possible to link the two levels of planning (aggregate and detailed scheduling) in an iterative procedure, where an initial allocation of

capacity is followed by detailed scheduling. The degree to which the overall objectives are satisfied can be used to decide whether a revision of the long-term capacity allocation is required or not. In this way, a feedback mechanism is used to control the aggregate loading (allocation of capacity).

Within the context defined above, a study of approximate methods has been carried out. A review of the exact (optimal) methods and of the recent developments in the theory of computational complexity has demonstrated the futility of research focused on finding 'good' algorithms for even the simplest form of the general n jobs m machines scheduling problem. It has also shown the limitations of the size (complexity) of problems that can be solved optimally with enumerative methods. The usefulness of optimal solutions has also been disputed, given the inevitable human errors in forecasting demand (numbers and types of jobs), processing times etc. Thus the problem has been reformulated into 'finding good approximate methods with predictable performance' and 'finding the most appropriate approximate algorithm for a scheduling problem with given characteristics'. Under this formulation, a range of problems has been investigated with various approximate methods.

For flow-shops, the simplest form of the scheduling problem, with and without in-process waiting, new non-enumerative heuristics have been constructed, tested against the most powerful published ones and found to be as good and usually better (Chapter 3).

For the general job-shop scheduling problem, the probabilistic and worst case performance of simple single-pass heuristic rules have been established for the first time, with a method that could be the model for a similar analysis of all sorts of heuristics. A model for describing the probabilistic behaviour of heuristics has been tested and found to be a satisfactory predictor for other problems of different complexity (Chapter 4).

It has been felt that approximate methods based on local neighbourhood search (LNS) could become a feasible proposition for many problems, where more accuracy than that of a single-pass heuristic is required, without bearing the computational cost of a complete tree search. With this objective in mind, a number of issues have been studied. Strong lower bounds have been calculated based on the relaxation of capacity constraints in all but two machines. The complexity of job-shop scheduling problems has been analysed and some characteristics of its tree representation (depth) have been established, allowing an a-priori estimate of the tree size. No correlation has been found between the variance of processing times or the scheduling rule used for the tree generation and the depth or the estimated size of the tree. A model describing the proportion of the tree searched as a function of CPU time or number of iterations has been proposed. An analysis has been carried out for simple heuristics in this partially enumerative method, establishing for the first time, bounds of heuristic performance, probabilistic descriptors of expected behaviour and a model for predicting the potential outcome of incomplete search procedures. (Chapter 5)

Applications of statistical sampling methods in local neighbourhood search (LNS) have been proposed, for stopping rules and bound calculations. The Weibull distribution has been used as the limiting form of the frequency distribution of the smallest members of samples of feasible job-shop schedules, for the first time. Estimates of the optimal solution have been obtained by calculating the most likely value of the Weibull location parameter, allowing a tight bracket for the optimal solution to be established. Estimates of the optimal solution of sub-problems, with fixed partial sequence defined by the LNS partitioning method, have been proposed as approximate lower bounds for a 'best-bound' search strategy (Chapter 6).

For the stochastic-dynamic problem, where no exact analytical method can be applied, simulation has been used to study special aspects of the general problem. The sensitivity of simple and new composite heuristics to changes in loading and in the data structures (distribution, variance and error of estimates of processing times) has been studied. Simple approximate models based on similarities of job-shops to simpler queueing models, have been established for predicting the performance of these heuristics (dispatching rules), as well as the effects of changes in the number of machines in parallel (Chapter 7).

8.2 Suggestions for further research

It is believed that no P-class algorithm can be constructed for the general job-shop scheduling problem and that the practical benefits from finding a 'good' algorithm for some special case of a NP-complete problem with one, two or three machines at best, are insignificant. Thus, it is argued that future research should concentrate in the area of sub-optimal or approximate methods, trying to establish guarantees of performance, and in their implementation. The latter point is of great importance, given the existing gap between the theory of scheduling and the scheduling practices in industry (King, 1976, Dudek et al, 1974). Although the general job-shop scheduling problem cannot be solved with exact methods, the theory of scheduling, compared with the practice, is relatively advanced. Work needs to be done towards bridging this gap, taking into account the potential of the theory (single-pass heuristics, LNS, simulation) and the current computer technology.

Some ideas which occured to the author during this project are listed below. They were not however pursued for lack of time and for computational expense.

(i) Flow-shop heuristics (Chapter 3)

A number of weighting coefficients could be tried in the 'slack-based' and 'savings' heuristics and the ones with the best performance for each type of problem established.

(ii) Local Neighbourhood Search (Chapters 5 and 6)

Instead of using the same heuristic scheduling rule during a LNS, it is worth trying different ones at various instances of the search. In this way, simple rules could be used at the beginning of the tree and some global rule, with a look ahead mechanism, near the bottom of

the tree. Alternatively, an initial solution and the related local neighbourhood would be defined with some method. The search could then be carried out using a simple heuristic, with the objective of studying its performance as a function of the initial neighbourhood.

Another idea worth exploring is concerned with a different backtracking mechanism. This is a heuristic method whereby backtracking to the immediately previous level is not always necessary. In this procedure, nodes that do not appear to be promising would be ignored by 'jumptracking'. The criterion for ignoring nodes could be based on comparison of the lower bounds; if the lower bounds at two different level nodes are equal, further backtracking takes place. It could be based also on a statistical estimate of the optimal of the particular sub-problem defined by the node in question. The idea of jumptracking seems to be particularly useful in an interactive mode of operation of LNS.

(iii) Simulation and approximate network formulae.

The dynamic job-shop scheduling problem can be studied as a sequence of static instances (Nelson et al 1977), with some form of local neighbourhood search, where the jobset of executable operations is updated at regular review times, with or without preemption. A similar approach can be used when the set-up times are sequence dependent. Every time a machine becomes idle, a heuristic rule can be applied to select the next operation (eg. minimum change-over cost).

The relative performance of single-pass non-delay (dispatching) rules can be studied with simulation under conditions of limited storage capacity between different work-centres (limited in-process inventory), or under the assumption of machines breaking down

according to a Weibull process. When the machines are assumed to have breakdowns, the operational flexibility becomes important. The effect of allowing such flexibility on the average waiting time can be studied again with simulation.

The simulation results obtained (Chapter 7) suggest that approximate formulae, based on interpolation, might be developed also for a given processing times distribution, with the load factor as the independent variable.

(iv) Interaction between detailed scheduling and aggregate capacity planning.

Attempts to link these two levels of planning have been restricted to simplified job-shop models (Schwimer, 1972, Gelders and Kleindorfer, 1974 and 1975). It is believed that there is scope for further research in this direction, trying to link capacity planning of real life production planning problems (eg Spachis, 1975, Adam and Surkis, 1977) with detailed scheduling, probably in the form of LNS, as described in this thesis.

(v) Interaction between automatic LNS and human scheduler.

It is envisaged that the schedules would be presented in a Gantt-chart form with the help of a video display unit (VDU), where the scheduler would be able to resolve conflicts himself or instruct the machine to do it according to prespecified routines.
A visual simulation would display the progress of scheduling.
The interactive software would then ask the scheduler whether to continue or jumptrack, and whether to estimate the optimal solution with sampling.

(vi) Implementation of multiple objective criteria.

The criteria of performance used in industry are more complex than 'minimise makespan, machine idle time, work in progress' etc. They can be 'minimise cost', 'maximise return' and more generally, 'maximise profit'. The degree to which these generalised criteria are satisfied can be measured with an objective function which is set up as the weighted sum of all the deviations from given targets. The difficulty is that the weight coefficients are necessarily subjective and thus, in abstract form, meaningless. One can assign values to them relevant to a specific context only. At this point, the necessity of research in scheduling practices in industry becomes apparent.

(vii) Routines for computerised scheduling.

A review of the Scheduling Handbook (O'Brien, 1969) shows that the number of potential applications of scheduling techniques which might influence the decision processes of managers, is very large. Although there is some scope for manual application of scheduling methods (New, 1975), the majority of real life scheduling problems is so complex, that in some way or another, the implementation has to be computerised. The interest in implementation lies not in one-off applications but in complete and permanent adoption of the approach, in the integration and institutionalization of scheduling techniques. There is evidence that this is rather limited (King, 1972 and 1975, Chazapis, 1977), confined to simple forms of data processing and record keeping (Nicholson and Pullen, 1974, Holden, 1976) rather than elaborate decision making. This becomes apparent, when reviewing the available packages on

production control.

The commercial packages of computer manufacturers (PCS II of Burroughs, FACTOR of Honeywell, CAPOSS of IBM, PROMPT of ICL etc) provide a wide coverage of data processing and control functions. Their main features include order processing (eg parts explosion, requirements planning), some form of stock control, work-in-progress control, and usually simple dispatching rules for scheduling.

It is felt that more sophisticated scheduling routines, like a depth-first LNS, can be of value in improving scheduling procedures. These routines with relatively small core requirements could be used locally in mini-or even micro-processors and the resulting schedules could be implemented at shop floor level, requiring some form of interfacing with the main processors where the packages are run. The development and implementation of these types of routines(procedures)either as free-standing programs or in conjuction with some computerised data processing and control system,is a feasible task even for small or medium size companies.

REFERENCES

ADAM N. and SURKIS J. (1977) A comparison of capacity planning techniques in a job shop control system, Management Sci. 23, 1011-1015.

AGGARWAL S.C.,WYMAN P.F. and McCARL B.A. (1973) An investigation of a cost-based rule for job-shop scheduling, Int. J. Production Research 11, 247-261.

ASHOUR S. (1970) A branch and bound algorithm for flow-shop scheduling problems, AIIE Trans. 2, 172-176.

ASHOUR S. and HIREMATH S.R. (1973) A branch and bound approach to the job shop sceduling problem, Int.J.Production Research 11, 47-58.

ASHOUR S.,MOORE T.E. and CHIU K.Y. (1974) An implicit enumeration algorithm for the nonpreemptive shop scheduling problem, AIIE Trans. 6, 62-72.

ASHOUR S. and PARKER R.G. (1971) A precedence graph algorithm for the shop scheduling problem, Operational Res. Quart. 22, 165-175.

ASHOUR S. and QURAISHI M.N. (1969) Investigation of various bounding procedures for production scheduling problems, Int. J. Production Research 8, 249-252.

ASHOUR S. and VASWANI S.D. (1972) A GASP simulation study of job-shop scheduling, Simulation 18, 1-10.

BAKER C.T. and DZIELINSKI B.P. (1960) Simulation of a simplified job shop, Management Sci. 6, 311-323.

BAKER K.R. (1974) Introduction to sequencing and scheduling, Wiley and Sons, New York.

BAKER K.R. (1975b)An elimination method for the flow-shop problem, Operations Res. 23, 159-162.

BAKER K.R. (1975a)A comparative study of flow-shop algorithms, Operations Res. 23, 62-73.

BAKER K.R. and MARTIN J.B. (1974) An experimental comparison of solution algorithms for the single-machine tardiness problem, Naval Res. Logistics Quart. 21, 187-206.

BAKER K.R. and SCHRAGE L.E. (1978) Finding an optimal sequence by dynamic programming: an extension to precedence-related tasks, Operations Res. 26, 111-120.

BAKER K.R. and SU Z.-S. (1974) Sequencing with due-dates and early start times to minimize maximum tardiness, Naval Res. Logistics Quart. 21, 171-176.

BALAS E. (1969) Machine sequencing via disjunctive graphs: an implicit enumeration algorithm, Operations Res. 17, 941-957.

BANSAL S.P. (1977) Minimising the sum of completion times of n jobs over m machines in a flowshop: a branch-and-bound approach, AIIE Trans. 9, 306-311.

BAZARAA M.S. and ELSHAFEI A.N. (1977) On the use of fictitious bounds in tree search algorithms, Management Sci. 23, 904-907.

BESTWICK P.F. and HASTINGS N.A.J. (1976) A new bound for machine scheduling, Operational Res. Quart. 27, 479-487.

BONNEY M.C. and GUNDRY S.W. (1976) Solutions to the constrained flowshop sequencing problem, Operational Res. Quart. 27, 869-883.

BOWMAN E.H. (1959) The schedule-sequencing problem, Operations Res. 7, 621-624.

BRATLEY P., FLORIAN M. and ROBILLARD P. (1971) Scheduling with earliest start and due-date constraints, Naval Res. Logistics Quart. 18, 511-519.

BRATLEY P., FLORIAN M. and ROBILLARD P. (1973) On sequencing with earliest start and due dates with application to computing bounds for the $n/m/G/F_{max}$ problem. Naval Res. Logistics Quart. 20, 57-67.

BRATLEY P., FLORIAN M. and ROBILLARD P. (1975) Scheduling with earliest start and due date constraints on multiple machines, Naval Res. Logistics Quart. 22, 165-174.

BROOKS G.H. and WHITE C.R. (1965) An algorithm for finding optimal or near optimal solutions to the production scheduling problem, J. Industrial Engineering 16, 34-40.

BRUNO J.L. (1976) Sequencing jobs with stochastic task structures on a single machine, J. Assoc. Comp. Mach. 23, 655-664.

BUFFA E.S. and TAUBERT W.H. (1972) Production-inventory systems; Planning and Control, Irwin, Homewood, Illinois.

BURNS R.N. (1976) Scheduling to minimise the weighted sum of completion times with secondary criteria, Naval Res. Logistics Quart. 23, 125-129.

BURNS F. and ROOKER J. (1975) A special case of the 3xn flowshop problem, Naval Res. Logistics Quart. 22, 811-817.

BURNS F. and ROOKER J. (1976) Johnson's three-machine flow-shop conjecture, Operations Res. 24, 578-580.

BURNS F. and ROOKER J. (1978) Three-stage flow-shops with recessive second stage, Operations Res. 26, 207-208.

CAMPBELL H.G., DUDEK R.A. and SMITH M.L. (1970) A heuristic algorithm for the n-job, m-machine sequencing problem, Management Sci. 10, B630- 637.

CHARLTON J.M. and DEATH C.C. (1970a)A generalised machine-scheduling algorithm, Operational Res. Quart. 21, 127-134.

CHARLTON J.M. and DEATH C.C. (1970b)A method of solution for general machine-scheduling problems, Operations Res. 18, 689-707.

CHAZAPIS L. (1977) Minicomputers for production planning and control, M Sc Dissert, Dept. of Management Sci., Imperial College of Science and Technology, University of London.

CHOWDHURY I.G. (1976) Job scheduling with single and multiple operations, Ph D Thesis, Dept. of Management Sci., Imperial College of Science and Technology, University of London.

CHOWDHURY M.A. (1974) Operating procedures for multi-resource constrained systems, Ph D Thesis, Faculty of Science and Engineering, University of Birmingham.

CHRISTOFIDES N. (1975) Graph Theory: An algorithmic approach, Academic Press, London.

CHRISTOFIDES N. (1976) Worst-case analysis of a new heuristic for the travelling salesman problem, Research Report 388, Grad. School of Industrial Admin., Carnegie-Mellon University, Pittsburgh, Pensylvania.

COFFMAN E.G. (ed) (1976) Computer and job-shop scheduling theory, Wiley and Sons, New York.

COFFMAN E.G. and GRAHAM R.L. (1972) Optimal scheduling for two-processor systems, Acta Informatica 1, 200-213.

CONWAY R.W. (1965) Priority dispatching and work-in-process inventory in a job shop, J. Industrial Engineering 16, 123-130.

CONWAY R.W. and MAXWELL W.L. (1962) Network dispatching by the shortest operation discipline, Operations Res 1, 51-73.

CONWAY R.W., MAXWELL W.L. and MILLER L.W. (1967) Theory of scheduling, Addison-Wesley, Reading, Massachusetts.

COOK S.A. (1971) The complexity of theorem-proving procedures, Proceedings of the 3rd Annual ACM Symposium on the theory of computing, 151-158.

CORWIN B.D. and ESOGBUE A.O. (1974) Two machine flowshop scheduling problems with sequence dependent setup times: a dynamic programming approach, Naval Res. Logistics Quart. 21, 515-524.

COSMETATOS G.P. (1974) Approximate equilibrium results for the multi-server queue (GI/M/r), Operational Res. Quart. 25, 625-634.

COSMETATOS G.P. (1975)Approximate equilibrium results for the average queueing time in the processes (M/D/r) and (D/M/r), INFOR - Canadian J. Operational Res. 13, 328-331.

COSMETATOS G.P. (1976) Some approximate equilibrium results for the multi-server queue (M/G/r), Operational Res. Quart. 27, 615-620.

COSMETATOS G.P. (1977) Cobham's model on preemptive multi-server queueing systems, European J. Operat. Res. 1, 262-264.

COX D.R. and MILLER H.D. (1965) The theory of stochastic processes, Methuen, London.

COX D.R. and SMITH W.L. (1961) Queues, Methuen, London.

CUNNINGHAM A. and TURNER I.B. (1973) Decision analysis for job-shop scheduling, Omega 1, 733-746.

DANNENBRING D.G. (1977) An evaluation of flow shop sequencing heuristics, Management Sci. 23, 1174-1182.

DAY J.E. and HOTTENSTEIN M.P. (1970) Review of sequencing research, Naval Res. Logistics Quart. 17. 11-32.

DAY J.E. and HOTTENSTEIN M.P. (1975) The impact of advancing due-dates in a pure job shop, Int. J. Production Research 13, 603-613.

DESSOUKY M.I. and MARGENTHALER C.R. (1972) The one-machine sequencing problem with early starts and due dates, AIIE Trans. 4, 212-222.

DUDEK R.A., SMITH M.L. and PANWALKAR S.S. (1974) Use of a case study in sequencing-scheduling research, Omega 2, 253-261.

EDMONDS J. (1965) Paths, trees and flowers, Canadian J. Mathematics 17, 449-467.

EILON S. The job shop simulation programme, Int. J. Production Research 11, 299-300.

EILON S. and CHOWDHURY I. (1975) Studies in a simulated job-shop, Proc. I. Mech. E. 189, 417-425.

EILON S. and CHOWDHURY I.G. (1977) Minimising waiting time variance in the single machine problem, Management Sci. 23, 567-575.

EILON S., CHOWDHURY I. and SERGHIOU S. (1975) Experiments with the SI* rule in job-shop scheduling, Simulation, 24, 45-48.

EILON S. and COTTERILL D.J. (1968) A modified SI rule in job shop scheduling, Int. J. Production Research 7, 135-145.

EILON S. and HODGSON R.M. (1967) Job-shop scheduling with due dates, Int. J. Production Research 6, 1-13.

EILON S. and KING J.R. (1967) Industrial scheduling abstracts (1950-1966), Oliver and Boyd, London.

EILON S., WATSON-GANDY C.D.T. and CHRISTOFIDES N. (1971) Distribution management: mathematical modelling and practical analysis, Griffin, London.

ELMAGHRABY S. (1968) The machine sequencing problem - review and extensions, Naval Res. Logistics Quart. 15, 205-232.

ELMAGHRABY S.E. (ed) (1973) Symposium on the theory of scheduling and its applications, Lecture notes in economics and mathematical systems, Springer-Verlag, Berlin.

ELVERS D.A. (1974) The sensitivity of the relative effectiveness of job-shop dispatching rules with respect to various arrival distributions, AIIE Trans. 6, 41-49.

ERDOS P. and SPENCER J. (1974) Probabilistic methods in combinatorics, Academic Press, New York.

ESKEW J.D. and PARKER R.G. (1975) Computational experience with a cost-based algorithm for the shop scheduling problem, Operational Res. Quart. 26, 211-215.

FISHER M.L. (1973) Optimal solution of scheduling problems using Lagrange multipliers: Part I, Operations Res. 21, 1114-1127.

FISHER M.L. (1976) A dual algorithm for the one-machine scheduling problem, Math. Programming 11, 229-251.

FISHER R.A. and TIPPETT L.H.C. (1927) Limiting forms of the frequency distribution of the largest or smallest member of a sample, Proceedings of the Cambridge Philosophical Society 24, 180-190.

FISHMAN G. (1973) Discrete event digital simulation: concepts and methods, Wiley and Sons, New York.

FLORIAN M., TILQUIN C. and TILQUIN G. (1975) An implicit enumeration algorithm for complex scheduling problems, Int. J. Production Research 13, 25-40.

FLORIAN M., TREPANT P. and McMAHON G. (1971) An implicit enumeration algorithm for the machine sequencing problem, Management Sci. 17. B782-792.

GAREY M.R., GRAHAM R.L. and JOHNSON D.S. (1978) Performance guarantees for scheduling algorithms, Operations Res. 26, 3-21.

GAREY M.R., JOHNSON D.S. and SETHI R. (1976) The complexity of flow-shop and job-shop scheduling, Math. Operations Res. 1, 117-129.

GELDERS L. and KLEINDORFER P.R. (1974) Coordinating aggregate and detailed scheduling decisions in the one-machine job-shop. I Theory, Operations Res. 22, 46-60.

GELDERS L. and KLEINDORFER P.R. (1975) Coordinating aggregate and detailed scheduling decisions in the one-machine job-shop. II Computation and structure, Operations Res. 23, 312-324.

GELDERS L.F. and SAMBANDAM N. (1978) Four simple heuristics for scheduling a flow-shop, Int. J. Production Research 16, 221-231.

GEOFFRION A.M. (1971) Duality in nonlinear programming, SIAM Review 13, 1-37.

GEOFFRION A.M. (1974) Lagrangean relaxation and its uses in integer programming, Math. Programming Study 2, 82-114.

GERE W. (1966) Heuristics in job-shop scheduling, Management Sci. 13, 164-180.

GIFFLER B. and THOMPSON G. (1960) Algorithms for solving production-scheduling

problems, Operations Res. 8. 487-503.

GILMORE P.C. and GOMORY R.E. (1964) Sequencing a one state-variable machine: a solvable case of the traveling salesman problem, Operations Res. 12, 655-679.

GOLDEN B.L. (1977) A statistical approach to the TSP, Networks 7, 209-225.

GONZALEZ T. and SAHNI S. (1978) Preemptive scheduling of uniform processor systems, J. Assoc. Comp. Mach. 25, 92-101.

GONZALEZ T. and SAHNI S. (1978) Flowshop and jobshop schedules: complexity and approximation, Operations Res. 26, 36-52.

GRAHAM R.L. (1966) Bounds for certain multiprocessing anomalies, Bell System Tech. J. 45, 1563-1581.

GRAHAM R.L. (1969) Bounds on multiprocessing timing anomalies, SIAM J. Applied Math. 17, 416-429.

GRAHAM R.L. (1978) The combinational mathematics of scheduling, Scientific American 238, 124-132.

GREENBERG H.H. (1968) A branch-bound solution to the general scheduling problem, Operations Res. 16, 353-361.

GUPTA J.N.D. (1971) The generalized n-job m-machine scheduling problem, Opsearch 8, 173-185.

GUPTA J.N.D. (1972) Heuristic algorithms for multistage flowshop scheduling problems, AIIE Trans. 4, 11-18.

GUPTA J.N.D. (1975) Optimal schedules for special structure flowshops, Naval Res. Logistics Quart. 22, 255-269.

GUPTA J.N.D. (1976) Optimal flowshop schedules with no intermediate storage space, Naval Res. Logistics Quart. 23, 235-243.

GUPTA J.N.D. (1973) Flowshop scheduling by heuristic decomposition, Int. J. Production Research 11, 105-111.

GUPTA J.N.D. (1978) Improved dominance conditions for the three-machine flowshop scheduling problem, Operations Res. 26, 200-203.

HARDGRAVE W.W. and NEMHAUSER G.L. (1963) A geometric method and a graphical algorithm for a sequencing problem, Operations Res. 12, 655-679.

HASTINGS N.A.J. and PEACOCK J.B. (1974) Statistical Distributions, Butterworths, London.

HELLER J. (1960) Some numerical experiments for an M x J flow shop and its decision-theoretical aspects, Operations Res. 8, 178-184.

HELD M. and KARP R.M. (1962) A dynamic programming approach to sequencing problems, J. SIAM 10, 196-210.

HELD M. and KARP R.M. (1970) The travelling salesman problem and

minimum spanning trees, Operations Res. 18, 1138-1162.

HELD M. and KARP R.M. (1971) The travelling salesman problem and minimum spanning trees: Part II, Math. Programming 1, 6-25.

HERSHAUER J.C. and EBERT R.J. (1975) Search and simulation selection of a job-shop sequencing rule, Management Sci. 21, 833-843.

HOEL P.G. (1971) Introduction to mathematical statistics, 4th ed., Wiley and Sons, New York.

HOLDEN G.K. (1976) Production control: packages and services, 2nd ed., National Computing Centre, Factfinder 13.

HOLLIER R.H. (1968) A simulation study of sequencing in batch production, Operational Res. Quart. 19, 389-407.

HOLLOWAY C.A. and NELSON R.T. (1973) Alternative formulation of the job-shop problem with due dates, Management Sci. 20, 65-75.

HOLLOWAY C.A. and NELSON R.T. (1974) Job-shop scheduling with due-dates and variable processing times, Management Sci. 20, 1264-1275.

HOLLOWAY C.A. and NELSON R.T. (1975) Job-shop scheduling with due dates and operation overlap feasibility, AIIE Trans. 7, 16-20.

HORN W.A. (1972) Single-machine job sequencing with treelike precedence ordering and linear delay penalties, SIAM J. Applied Math. 23, 189-202.

HOTTENSTEIN M.P. (1970) Expediting in job-order-control systems: a simulation study, AIIE Trans. 2, 46-54.

HU T.C. (1961) Parallel sequencing and assembly line problems, Operations Res. 9, 841-848.

IBARAKI T. (1976a)Computational efficiency of approximate branch and bound algorithms, Math. Operations Res. 1,287-298.

IBARAKI T. (1976b)Theoretical comparisons of search strategies in branch-and-bound algorithms, Int. J. Computer and Information Sci. 5, 315-344.

IBARAKI T. (1977) On the computational efficiency of branch-and-bound algorithms, J. Operations Res. Soc. Japan 20, 16-35.

IGNALL E. and SCHRAGE L. (1965) Application of the branch-and-bound technique to some flow-shop scheduling problems, Operations Res. 13, 400-412.

JACKSON J.R. (1956) An extension of Johnson's results on job lot scheduling, Naval Res. Logistics Quart. 3, 201-203.

JACKSON J.R. (1957) Simulation research on job-shop production, Naval Res. Logistics Quart. 4, 287-295.

JACKSON J.R. (1963) Jobshop-like queueing systems, Management Sci. 10, 131-142.

JOHNSON S.M. (1954) Optimal two- and three-stage production schedules with set up times included, Naval Res. Logistics Quart. 1, 61-68.

JOHNSON D.S., DEMERS A., ULLMAN J.D., GAREY M.R. and GRAHAM R.L. (1974) Worst-case performance bounds for simple one-dimensional packing algorithms, SIAM J. Computing 3, 299-326.

JONES C.H. (1973) An economic evaluation of job shop dispatching rules, Management Sci. 20, 293-307.

JSS (1972) Job shop simulation programme manual, Report, Dept. of Management Science, Imperial College of Science and Technology, University of London.

KARP R.M. (1972) Reducibility among combinational problems in MILLER R.E. and THATCHER J.W. (eds) 'Complexity of computer computations', Plennum Press, New York.

KARP R.M. (1975) On the computational complexity of conbinational problems, Networks 5, 45-68.

KARP R.M. (1977) Probabilistic analysis of partitioning algorithms for the travelling-salesman problem in the plane, Math. Operations Res. 2, 209-224.

KING J.R. (1972) Production planning and control by computer - A survey, The Production Engineer 51, 333-336.

KING J.R. (1975) Production planning and control; an introduction to quantitative methods, Pergamon Press, Oxford.

KING J.R. (1976) The theory-practice gap in job-shop scheduling, The Production Engineer 55, 137-143.

KINGMAN J.F.C. (1970) Inequalities in the theory of queues, Journal of the Royal Statistical Society, B 32, 102-107.

KLEINROCK L. (1975) Queueing systems: Vol 1 Theory, Wiley and Sons, New York.

KLEINROCK L. (1976) Queueing systems: Vol 2 Comput. applications, Wiley and Sons, New York.

KISE H., IBARAKI T. and MINE H. (1978) A solvable case of the one-machine scheduling problem with ready and due times, Operations Res. 26, 121-126.

KLEE V. and MINTY G.I. (1972) How good is the simplex algorithm ?, 159-175, In: SHISHA O. (ed.) 'Inequalities III', Academic Press, New York.

KOHLER W.H. and STEIGLITZ K. (1975) Exact, approximate and guaranteeded accuracy algorithms for the flow-shop problem $n/2/F/\overline{F}$ J. Assoc. Comput. Mach. 22, 106-114.

KRONE M.J. and STEIGLITZ K. (1974) Heuristic programming solution of a flow-shop scheduling problem, Operations Res. 22, 629-638.

LAGEWEG B.J., LENSTRA J.K and RINNOY KAN A.H.G. (1978) A general bounding scheme for the permutation flow-shop problem, Operations Res. 26, 53-67.

LAWLER E.L. (1964) On scheduling problems with deferral costs, Management Sci. 11, 280-288.

LAWLER E.L. (1973) Optimal sequencing of a single machine subject to precedence constraints, Management Sci. 19, 544-546.

LAWLER E.L. A "pseudopolynomial" algorithm for sequencing jobs to minimize total tardiness, Annals of Discrete Mathematics 1, 331-342.

LAWLER E.L. and MOORE J.M. (1969) A functional equation and its application to resource allocation and sequencing problems, Management Sci. 16, 77-84.

LAWLER E.L. and WOOD D.E. (1966) Branch and bound methods: a survey, Operations Res. 14, 699-719.

LEE A.M. (1966) Applied queueing theory, Macmillan.

LENSTRA J.K. (1977) Sequencing by enumerative methods, Mathematical Centre Tracts 69, Amsterdam.

LENSTRA J.K., RINNOOY KAN A.H.G. and BRUCKER P. (1977) Complexity of machine scheduling problems, Annals of Discrete Mathematics 1, 343-362.

LIN S. (1965) Computer solutions of the travelling salesman problem, Bell System Tech. J. 44, 2245-2269.

LITTLE J.D.G., MURTY K.G., SWEENEY D.W. and KAREL G. (1963) An algorithm for the travelling salesman problem, Operations Res. 11, 972-989.

LOMNICKI Z.A. (1965) A branch-and-bound algorithm for the exact solution of the three-machine scheduling problem, Operations Res. Quart. 16, 89-100.

MCMAHON G.B. and BURTON P.G. (1967) Flow-shop scheduling with the branch and bound method, Operations Res. 15, 473-481.

MCMAHON G.B. and FLORIAN M. (1975) On scheduling with ready times and due dates to minimise maximum lateness, Operations Res. 23, 475-482.

MANNE A.S. (1960) On the job shop scheduling problem, Operations Res. 8, 219-223.

MARCHAL W.G. (1976) An approximate formula for waiting time in single server queues, AIIE Trans. 8, 473-474.

MELLOR P. (1966) A review of job-shop scheduling, Operational Res. Quart. 17, 161-171.

MERTEN A.G. and MULLER M.E. (1972) Variance minimization in single machine sequencing problems, Management Sci. 18, 518-528.

MITTEN L.G. (1958) Sequencing n jobs on two machines with arbitrary time lags, Management Sci. 5, 293-298.

MITTEN L.G. (1970) Branch and bound methods: general formulation and properties, Operations Res. 18, 24-34.

MOOD A.M. and GRAYBILL F.A. (1963) Introduction to the theory of statistics, 2nd ed., McGraw Hill, New York.

MOODIE C.L. and ROBERTS S.D. (1968) Experiments with priority dispatching rules in a parallel processor shop, Int. J. Production Research 6, 303-312.

MOORE J.M. (1968) An n job, one machine sequencing algorithm for minimizing the number of late jobs, Management Sci. 14, 102-109.

MUTH J.F. and THOMPSON G.L. (eds.) (1963) Industrial scheduling, Prentice-Hall, Englewood Cliffs, New Jersey.

NELSON R.T. (1967) Labor and machine limited production systems, Management Sci. 13, 648-671.

NELSON R.T.,HOLLOWAY C.A. and WONG R.M.-L. (1977) Centralised scheduling and priority implementation heuristics for a dynamic job-shop model, AIIE Trans. 9, 95-102.

NEW C.C. (1975) Job-shop scheduling: Is manual application of dispatching rules feasible ?, Operations Res. Quart. 26, 35-43.

NEWELL G.F. (1973) Approximate stochastic behaviour of n-server service systems with large n, Springer-Verlag, Berlin.

NICHOLSON T.A.J. and PULLEN R.D. (1974) Computers in production management decisions, Pitman, London,

O'BRIEN J.J. (1969) Scheduling handbook, McGraw Hill ,

ORAL M. and MALOUIN J.L. (1973) Evaluation of the shortest processing time scheduling rule with truncation process, AIIE Trans. 5, 357-365.

PACE A.J. (1969) Priority loading rules in job-shop scheduling, MPhil. thesis, Management Engineering Section, Imperial College of Science and Technology, University of London.

PAGE E.S. (1961) An approach to the scheduling of jobs on machines, Journal of Royal Statistical Society, B 23, 484-492.

PAGE E. (1973) Queueing theory in O.R., Butterworth, London.

PALMER D.S. (1965) Sequencing jobs through a multi-stage process in the minimum total time - a quick method of obtaining a near optimum, Operational Res. Quart. 16, 101-107.

PANWALKAR S.S. and ISKANDER W. (1977) A survey of scheduling rules, Operations Res. 25, 45-61.

PANWALKAR S.S. and KHAN A.W. (1976) An ordered flow-shop sequencing problem with mean completion time criterion, Int. J. Production Research 14, 631-635.

PANWALKAR S.S. and KHAN A.W. (1977) A convex property of an ordered flow-shop sequencing problem, Naval Res. Logistics Quart. 24, 159-162.

PRABHU N.U. (1965) Queues and inventories, Wiley and Sons, New York.

PRITSKER A.A.B., WATTERS L.J. and WOLFE P.M. (1969) Multi-project scheduling with limited resources: a zero-one programming approach, Management Sci. 16, 93-108.

RANDOLPH P.H., SWINSON G. and ELLINGSEN C. (1973) Stopping rules for sequencing problems, Operations Res. 21, 1309-1315.

RAO N.P. (1976) A viable alternative to the 'method of stages' solution of series production systems with Erlang service times, Int. J. Production Research 14, 699-702.

RAU J.G. (1970) Minimising a function of permutations of n integers, Operations Res. 18, 237-240.

REDDI S.S. and RAMAMOORTHY C.V. (1972) On the flow-shop sequencing problem with no wait in process, Operational Res. Quart. 23, 323-331.

REINGOLD E.M., NIEVERGELT J. and DEO J. (1977) Combinatorial algorithms: theory and practice, Prentice-Hall, Eaglewood Cliffs, New Jersey.

RINNOYKAN A.H.G. (1976) Machine scheduling problems: classification, complexity and computations, Nijhoff, The Hague,

RINNOY KAN A.H.G., LAGEWEG B.J. and LENSTRA J.K. (1975) Minimizing total costs in one-machine scheduling, Operations Res. 23, 908-927.

ROCHETTE R. and SADOWSKI R. (1976) Statistical comparison of the performance of simple dispatching rules for a particular set of job-shops, Int. J. Production Research 14, 63-75.

ROSENSHINE M. and CHANDRA M.J. (1975) Approximate solutions for some two-stage tandem queues, Part 1: Individual arrivals at the second stage, Operations Res. 23, 1155-1166.

ROWE A.J. and JACKSON J.R. (1956) Research problems in production routing and scheduling, J. Industrial Engineering 7, 116-121.

SAATY T.L. (1961) Elements of queueing theory, McGraw Hill, New York.

SAHNI S.K. (1976) Algorithms for scheduling independent tasks, J. Assoc. Comp. Mach. 23, 116-127.

SAHNI S. and GONZALEZ T. (1976) P - complete approximation problems, J. Assoc. Comp. Mach. 23, 555-565.

SALKIN H.M. and KLUYVER C.A. (1975) The knapsack problem: a survey, Naval Res. Logistics Quart. 22, 127-144.

SCHRAGE L. (1970) Solving resource-constrained network problems by implicit enumeration - non-preemptive case, Operations Res. 18, 263-278.

SCHRAGE L. (1975) Minimizing the time-in-system variance for a finite jobset, Management Sci. 21, 540-543.

SCHMIDT J.W. and TAYLOR R.E. (1970) Simulation and analysis of industrial systems, Irwin, Illinois.

SHAPIRO J. (1977) A survey of Lagrangean techniques for discrete optimisation, Techn. Rep. 133, OR Centre, MIT, Cambridge, Massachusetts.

SHWIMER J. (1972) Interaction between aggregate and detailed scheduling in a job-shop, Ph D Dissertation, Sloan School of Management, MIT, Cambridge, Massachusetts.

SHWIMER J. (1972) On the n-job one-machine sequence independent scheduling problem with tardiness and penalties: a branch and bound solution, Management Sci. 18, 301-313.

SETHI R. (1977) On the complexity of mean flow time scheduling, Math. Operations Res. 2, 320-330.

SIDNEY J.B. (1975) Decomposition algorithms for single-machine sequencing with precedence relations and deferral costs, Operations Res. 23, 283-298.

SIDNEY J.B. (1977) Optimal single-machine scheduling with earliness and tardiness penalties, Operations Res. 25, 62-69.

SMITH W.E. (1956) Various optimisers for single-stage production, Naval Res. Logistics Quart. 3, 59-66.

SMITH M.L., PANWALKAR S.S. and DUDEK R.A. (1975) Flowshop sequencing problem with ordered processing time matrices, Management Sci. 21, 544-549.

SMITH M.L., PANWALKAR S.S. and DUDEK R.A. (1976) Flowshop sequencing problem with ordered processing time matrices: a general case, Naval Res. Logistics Quart. 23, 481-486.

SPACHIS A. (1975) Production scheduling in a canning company, M Sc Dissertation, Dept. of Management Science, Imperial College of Science and Technology, University of London.

SPACHIS A. (1978a) Routines for heuristic flow-shop scheduling with no-job-passing and/or no-waiting, Report, Dept. of Management Science, Imperial College of Science and Technology, University of London.

SPACHIS A. (1978b) Routines for local neighbourhood search and statistical methods in job-shop scheduling, Report, Dept. of Management Science, Imperial College of Science and Technology, University of London.

SZWARC W. (1960) Solution of the Akers-Friedman scheduling problem, Operations Res. 8, 782-788.

SZWARC W. (1971) Elimination methods in the m x n sequencing problems, Naval Res. Logistics Quart. 18, 295-305.

SZWARC W. (1974) Mathematical aspects of the 3 x n job-shop sequencing problem, Naval Res. Logistics Quart. 21, 145-154.

SZWARC W. (1977) Optimal two-machine orderings in the 3 x n flow-shop problem, Operations Res. 25, 70-77.

SZWARC W. (1978) Dominance conditions for the three-machine flow-shop problem. Operations Res. 26, 203-206.

SZWARC W. and HUTCHINSON G.K. (1977) Johnson's approximate method for the 3 x n job-shop problem, Naval Res. Logistics Quart. 24, 153-157.

TAHA H.A. (1976) Operations Research - an introduction, 2nd ed., Macmillan, New York.

TOWNSEND W. (1977a) Sequencing n jobs on m machines to minimise maximum tardiness: a branch-and-bound solution, Management Sci. 23, 1016-1019.

TOWNSEND W. (1977b) Minimising the maximum penalty in the m-machine sequencing problem, Operational Res. Quart. 28, 727-734.

TOWNSEND W. (1977c) Minimising the maximum penalty in the two-machine flow-shop, Management Sci. 24, 230-234.

ULLMAN J.D. (1976) Complexity of sequencing problems,139-164,in:COFFMAN E.G. (ed.) 'Computer and job-shop scheduling theory',Wiley and Sons, N.Y.

USKUP E. and SMITH S.B. (1975) A branch-and-bound algorithm for two-stage production-sequencing problems, Operations Res. 23, 118-136.

WAGNER H.M. (1972) Principles of operations research, Prentice Hall Int., London.

WAGNER H.M. (1959) An integer programming model for machine scheduling, Naval Res. Logistics Quart. 6, 131-140.

WEIBULL W. (1951) A statistical distribution function of wide applicability, Journal of Applied Mechanics 18, 293-297.

WISMER D.A. (1972) Solution of the flow-shop scheduling problem with no intermediate queues, Operations Res. 20, 689-687.

APPENDICES

APPENDIX A

Erlang process generator

The Erlang distribution is a statistical distribution which takes different forms for different values of a shape parameter k. In fact, the Erlang family of distributions is a special case of the Gamma, where the shape parameter is integer.

The Erlang probability density function is

$$f(x) = (x/b)^{k-1} \exp(-x/b)/\{b(k-1)!\}$$

where b is the scale parameter. The cumulative distribution function is

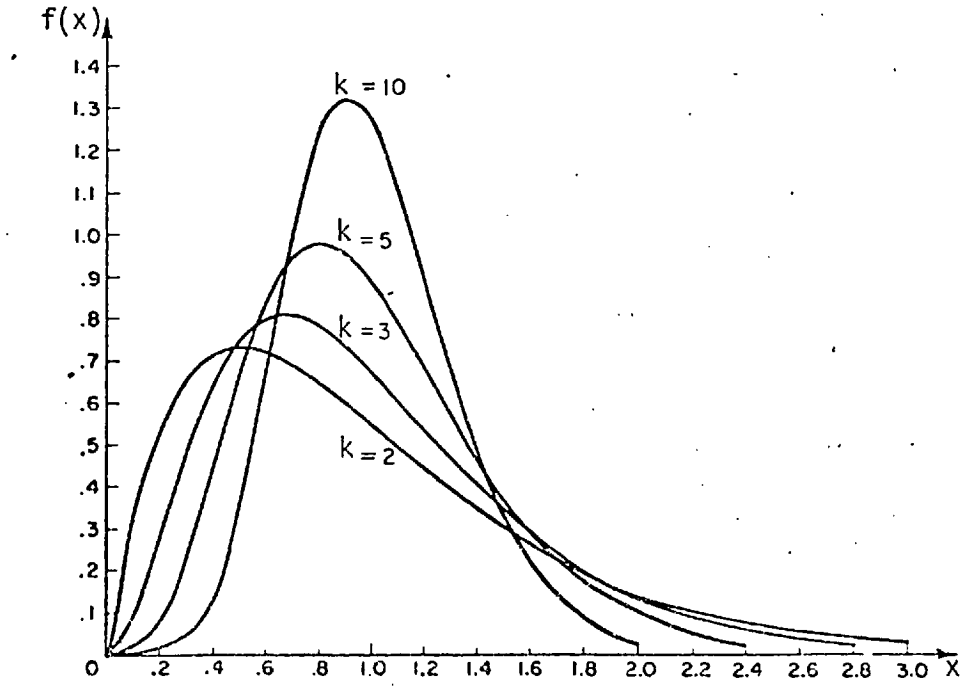
$$F(x) = 1-\{\exp(-x/b)\}\left(\sum_{i=0}^{k-1}(x/b)^i/i!\right)$$

where the mean is $\mu = bk$, the variance is $\sigma^2 = b^2k$ and hence, the coefficient of variation is

$$V = \sigma/\mu = 1/\sqrt{k}$$

The distribution density function is plotted below for a number of values of the shape parameter k.

Figure A1 Erlang distribution with scale parameter b=1



For k=1 the Erlang distribution reduces to the negative exponential. In fact the variate x of E_k is the sum of k independent variates x_i, i=1,k from a negative exponential with mean μ/k and variance μ^2/k^2

$$x = \sum_{i=1}^{k} x_i$$

with mean: $k(\mu/k) = \mu$ and variance: $k(\mu/k)^2 = \mu^2/k$

The implication is that if an operation is composed of k sub-operations coming from the same negative exponential then its aggregate distribution can be described by E_k.

Random numbers of the variate x of the negative exponential distribution with mean $1/\lambda = \mu/k = b$ can be computed from random numbers r of the uniform rectangular distribution since

$r = F(x)$, $r = 1-\exp(-\lambda x)$ and $x = (-1/\lambda)\ln(1-r)$

Suffices to use r instead of 1-r for x

$$x = (-1/\lambda)\ln r$$

The Erlang k random variate x then is:

$$x = (-\mu/k)\ln\left(\prod_{i=1}^{K} r_i\right)$$

Table A1 Heuristics for flow-shop scheduling with no-job-passing

No. of Jobs	No. of Mach.	Problem	Lower bound	Random	CDS	h1	h2	h3	h4	h5	Problem	Lower bound	Random	CDS	h1	h2	h3	h4	h5
			E_1 processing times									E_9 processing times							
10	4	C1	61	77	75	70	70	74	77	81	C6	58	89	73	74	73	74	80	77
		C2	68	109	86	92	84	90	94	100	C7	72	95	85	84	84	84	89	87
		C3	58	88	70	71	73	71	80	91	C8	63	85	78	78	78	78	87	83
		C4	61	87	81	87	78	87	101	98	C9	64	76	77	75	77	77	78	77
		C5	83	101	94	92	92	92	95	95	C10	70	87	79	78	78	87	78	80
20	5	D1	132	172	139	140	144	144	158	164	D6	119	147	146	146	143	145	152	146
		D2	152	208	205	215	202	214	220	212	D7	122	153	143	140	140	143	154	148
		D3	131	185	173	160	153	160	173	186	D8	123	148	142	143	142	143	155	152
		D4	144	191	153	158	161	164	187	182	D9	122	151	142	145	145	145	152	148
		D5	132	161	152	148	148	154	197	189	D10	120	151	144	145	142	144	157	150
35	5	E1		289	269	252	241	266	315	323	E6		229	218	225	223	228	234	229
		E2		263	241	257	253	262	320	314	E7		243	228	231	229	231	241	242
		E3		318	243	255	253	270	290	286	E8		244	231	234	228	229	250	244
		E4		295	237	223	223	237	307	314	E9		230	221	220	221	222	230	231
		E5		318	249	262	265	278	290	285	E10		240	227	229	230	232	235	233

Table A2 Heuristics for flow-shop scheduling with no-waiting

No. of Jobs	No. of Machines	E_1 processing times							E_9 processing times							
		Problem	Optimal Solution	Random	H1	H2	H3	H4	H5	Problem	Optimal Solution	Random	H1	H2	H3	H4	H5
10	4	B1	97	120	109	109	116	101	98	B6	79	98	80	85	86	81	80
		B2	89	129	93	109	110	94	90	B7	84	98	85	92	92	89	84
		B3	106	143	106	109	109	112	112	B8	79	84	81	85	81	79	81
		B4	82	112	86	88	89	89	86	B9	80	92	83	86	89	84	80
		B5	82	112	84	86	89	89	85	B10	74	86	74	78	77	77	77
20	5	D1	160	234	167	174	175	175	166	D6	152	183	156	156	160	156	155
		D2	227	342	238	251	266	238	236	D7	151	182	152	162	164	155	156
		D3	188	285	199	219	228	205	201	D8	149	172	155	153	153	154	155
		D4	175	266	189	185	185	189	178	D9	150	172	154	152	158	156	154
		D5	175	247	188	193	183	198	183	D10	151	177	155	156	160	157	153
35	5	E1	298	498	313	330	343	305	322	E6	232	285	234	244	243	236	244
		E2	295	452	297	297	297	295	313	E7	241	300	245	246	251	249	248
		E3	296	457	302	309	308	308	325	E8	249	309	256	252	255	249	253
		E4	264	417	285	290	297	286	296	E9	232	298	238	240	243	244	236
		E5	291	460	308	299	327	318	308	E10	239	244	249	250	249	245	251

APPENDIX B

Table B1

Bracket for optimal solution (BOS) with single pass active job-shop scheduling heuristic ECT

	E_9 processing times			E_{36} processing times		
Problem	n=10,m=4	n=20,m=5	n=35,m=5	n=10,m=4	n=20,m=5	n=35,m=5
1	.08	.03	.08	.10	.03	.07
2	.09	.06	.08	.14	.08	.09
3	.09	.07	.09	.14	.09	.10
4	.11	.08	.10	.14	.09	.10
5	.12	.08	.11	.14	.09	.15
6	.13	.08		.15	.10	
7	.13	.09		.15	.10	
8	.14	.09		.15	.12	
9	.14	.09		.18	.12	
10	.16	.10		.18	.12	
11	.17	.12		.18	.13	
12	.19	.13		.18	.13	
13	.19	.13		.19	.14	
14	.22	.14		.19	.14	
15	.23	.16		.19	.15	
16	.24	.18		.20	.15	
17	.25	.19		.20	.16	
18	.25	.20		.22	.16	
19	.29	.22		.22	.18	
20	.31	.22		.25	.18	

APPENDIX C

Extension to Jackson's method for $n/2/G/C_{max}$ problem

Algorithm (Jackson, 1956)

Partition set of jobs in 4 sub-sets, AB, A, BA, B according to the order of their operations (A and B stand for the two machines) where sets AB and BA are ordered with Johnson's rule. Construct optimal sequence by ordering the jobs as follows:

machine A AB,A,BA

machine B BA,B,AB

Jackson's algorithm gives the optimal sequence also when the two machines are not available simultaneously, as can be seen below. In machine A, there are no delays for operations of the subsets AB and A, and in machine B of the subsets BA and B. In the optimal schedule, delays are possible in one of the two machines only, and not in both simultaneously; assuming there are idle times in machine B (sets AB or A), there can be no delays for the operations of sets BA, B in machine A, because the operations of BA in machine B are completed before the AB,A operations in machine A, and thus any initial delay in machine A will not affect the optimal sequence. If, on the other hand, machine B is not available from the start of the schedule, delays that would arise otherwise will be reduced or eliminated, but again no other sequence can produce a schedule with smaller makespan.

Estimates of the size of the set of active solutions

The complete space of active solutions of a job-shop scheduling problem can be represented by a tree. Each traversal of this tree, starting from the first level of nodes, passing from one node at every other level and ending in the bottom of the tree, describes fully one solution. Although the size of the tree for every problem is bounded by $(n!)^m$, the actual size is substantially reduced by the 'characteristic function' which eliminates the non-feasible sequences, allowing only those described by the transfer function to be constructed.

Counting the members of the set A of the active solutions (ie the size of the tree) is not a practical proposition. Instead, it is possible to estimate its size by the value of the function T_d using a statistical method for branching processes (Cox and Miller, 1965). Each node i at some level d has a number of offsprings or branches, (between 2 and n) and the probabilities of these values occurring are respectively g_0, g_1, g_2, g_3,....g_n with a distibution $\{g_i\}$ ($g_0 = g_1 = 0$). Let $Y_d = j$ denote the number of nodes in the d - th generation (or level). The node i for i = 1,....j has z_i branches, where s_i has a distribution $\{g_i\}$. The number of nodes in the next generation is

$$Y_{d+1}=z_1+z_2+.....+z_j = \sum_{i=1}^{j} z_i$$

and $$\text{Prob}(Y_{d+1}=k/Y_d=j) = \text{Prob}(z_1+z_2+....+z_j=k)$$

The probability generating function P(t) defined for discrete variates is a function of an auxiliary variable t such that the coefficient of t^i is the probability density function g_i.

$$P(t) = \sum_{i=0}^{\infty} t^x g_i \quad , \quad x>o$$

and $$g_i = (1/x!)\ (\partial^i p(t)/\partial t^i)_{t=o}$$

Then $$\text{Prob}(z_1 + z_2 + \ldots + z_j = k) = \text{coefficient of } t^k \text{ in } \{P(t)\}^j$$

It is practically impossible to derive an explicit expression describing the distribution of k at level d. It has been possible though to calculate the mean and variance of its distribution recursively from

$$\mu_d = \mu\ \mu_{d-1}$$

$$\nu_d = \sigma^2 \mu^{2d-2} + \mu\nu_{d-1}$$

where μ, σ^2 are the mean and variance of number of branches per individual node.

Assuming that μ, σ^2 are the same for all nodes, then

$$\mu_d = \mu^d$$

$$\nu_d = \sigma^2 \mu^{d-1} (\mu^d - 1)/(\mu - 1$$

Under this assumption μ and σ^2 are easily calculated from the data generated during the tree search.

$$\mu = \frac{\text{number of branches generated}}{\text{number of nodes (conflicts) generated}} = N_b/N_c$$

$$\sigma^2 = (1/N_c)\sum_{i=2}^{n}(z_i-\mu)^2 f_i \qquad f_i \text{ the observed frequency of nodes with } z_i \text{ branches}$$

Every time a solution improvement is obtained, the proportion of the tree that has been searched is:

$$P_r = 1 - (\text{Remaining tree})/(\text{Total tree}) = 1 - R_d/T_d$$

where

$$R_d = \sum_{h=1}^{d} u_h \cdot \mu^{n-h}$$

h is the level number of an active node

u_h is the number of unexplored branches at the active node of level h

and thus

$$P_r = 1 - \left(\sum_{h=1}^{d} u_h \cdot \mu^{n-h}\right)/\mu^d$$

The value μ_d is only a statistical estimate, and the confidence for its value depends on the variance ν_d. Under the assumption of μ, σ^2 constant, ν_d depends mainly on the size of the tree described by μ^d. It is certain that $n > \mu \geq 2$ and therefore the terms μ^{d-1} and $(\mu^d-1)/(\mu-1)$ increase exponentially. The drawback of this method of estimating the size of the tree is that for large trees, even for small values of σ^2, ν_d is large. Another drawback of this method, which is particularly salient in large trees is the assumption of μ being constant. The initial stages of the tree search are likely to have more branches per node than those near the bottom of the tree. This is due to the fact that near the end, large partial schedules are already fixed and there are fewer unscheduled operations left as candidates for conflicts. This agrees with the computational

experience on this method with large problems where most of the solution time is spent searching the lower parts of the tree which results in an overall μ (average branches per node) decreasing with time.

Another method which appears to be more reliable has been used and is described below. This method is based on the assumption that the z_i sub-trees originating from the z_i branches of a node i at level h, have the same number of branches at the bottom of the tree (the same number of active solutions).

The important feature of this assumption is that the error although unknown is likely to be small and self-balancing. This method avoids completely any assumption about the values of μ and σ^2 and is more realistic, since it avoids using a constant depth d for the tree.

A recursive formula is constructed for calculating the total T_i and remaining tree R_i at level i, starting from the bottom of the tree (i=0 at bottom of tree, i=d at the top). At every level i, in the depth-first search method there is one active node only, with t_i branches, r_i of which are still unexplored (have not yet become active nodes).

For every complete active solution

$$T_0 = 1$$

$$R_0 = 0 \qquad \text{i.e.} \quad T_0 > R_0$$

At every level i

$$t_i \geq r_i + 1 \quad \rightarrow \quad t_i > r_i$$

It is

$$R_{i+1} = R_i + (T_i - r_i)$$

$$T_{i+1} = T_i \, t_i$$

If $T_i > R_i$, then $T_{i+1} > R_{i+1}$

$$T_{i+1} = T_i t_i \geq T_i r_i + T_i > T_i r_i + R_i = R_{i+1}$$

The proportion searched is computed from

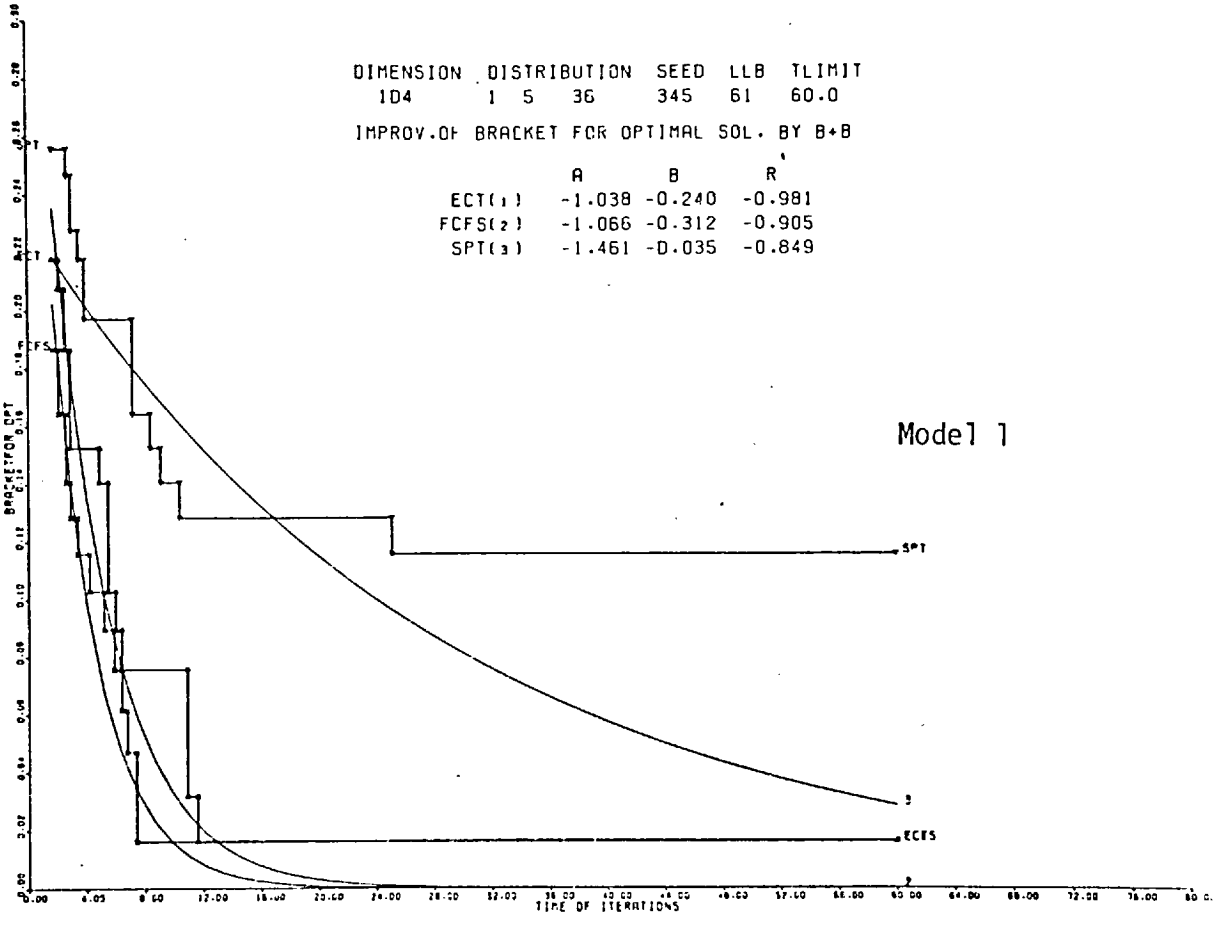
$$P_r = 1 - R_d/T_d$$

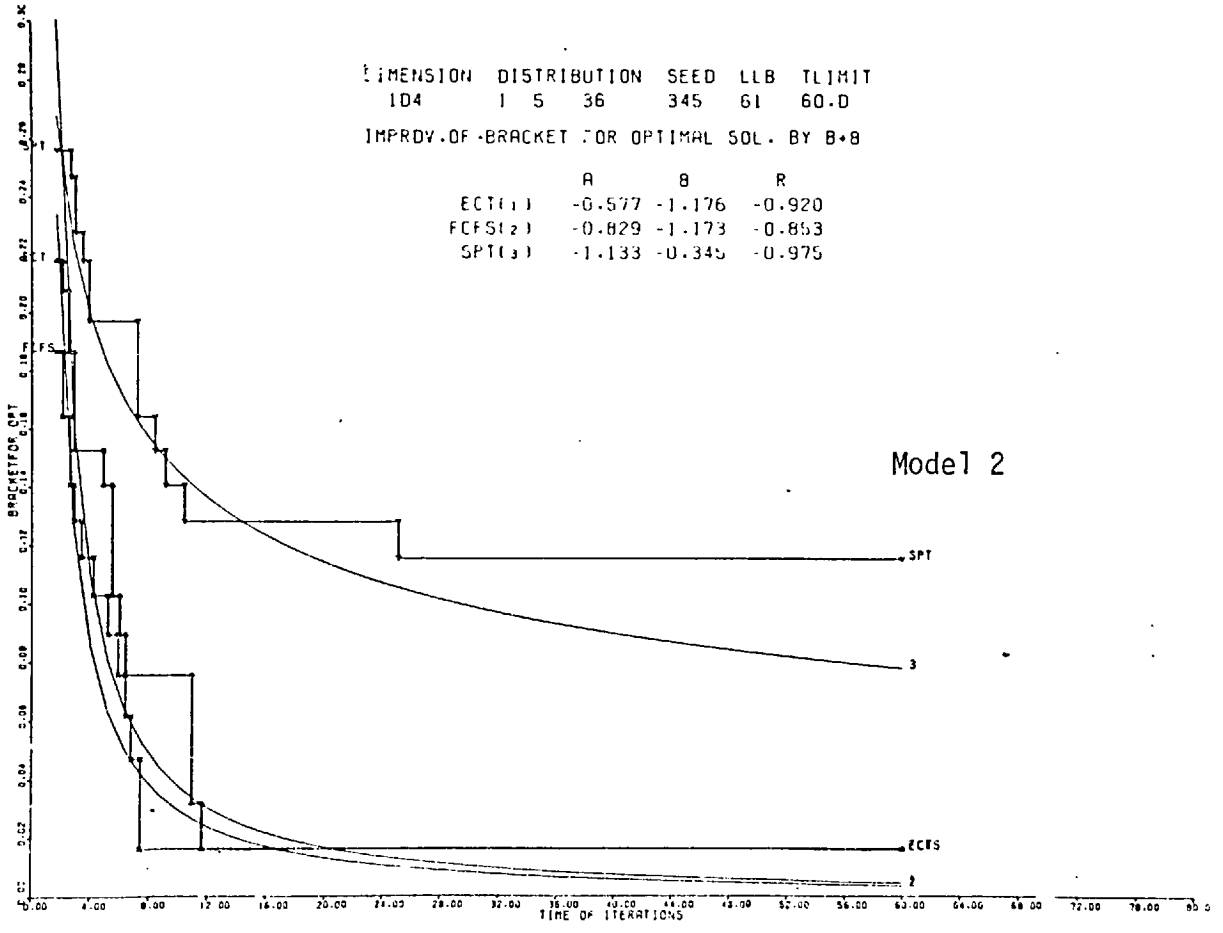
At large problems, T_i and R_i become very large numbers and special care must be taken to avoid overflow of digits in the computer storage of integers, which might result in T_i being treated as equal to R_i.

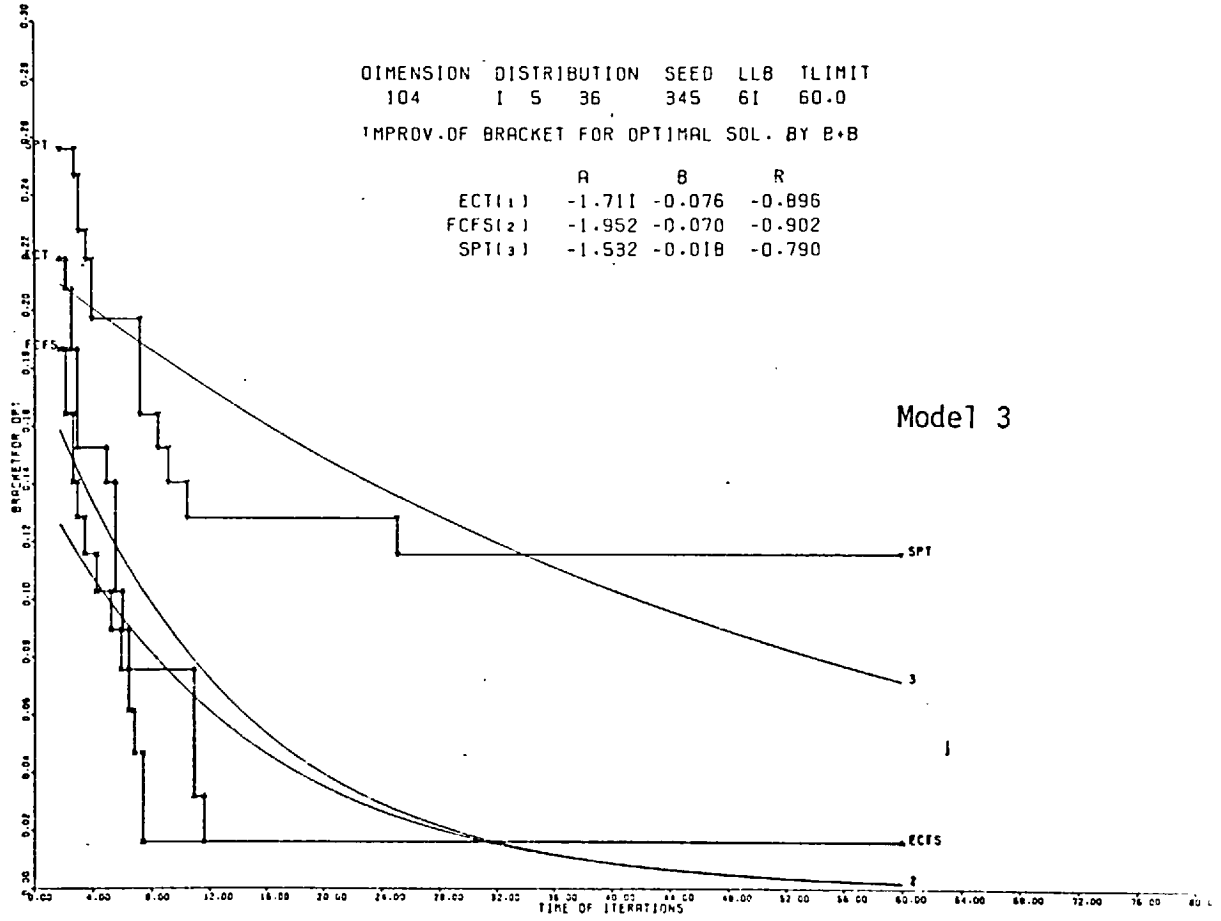
Table C1 Average and standard deviation of regression coefficients from the solution improvements model.

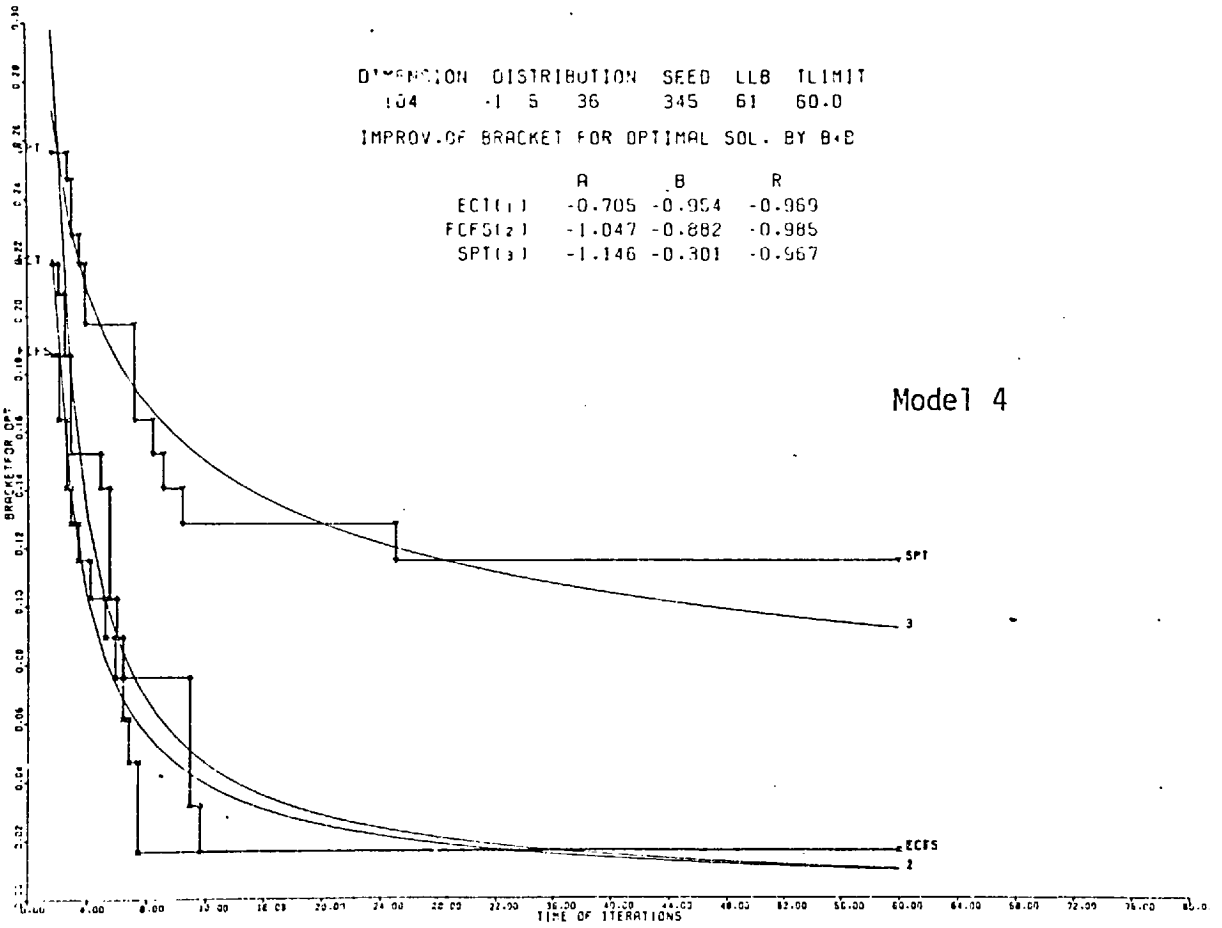
		TL=5 n=10 m=4			TL=10 n=10 m=4			TL=60 n=35 m=5		
		ECT	FCFS	SPT	ECT	FCFS	SPT	ECT	FCFS	SPT
E_1	mean	.435	.341	.448	.295	.148	.229	.215	.216	.199
	SD	.244	.235	.224	.191	.133	.118	.040	.106	.088
UR	mean	.403	.317	.289	.252	.343	.335	.235	.133	.102
	SD	.183	.188	.243	.180	.159	.115	.213	.033	.056
E_4	mean	.414	.271	.301	.296	.230	.308	.162	.364	.130
	SD	.189	.183	.161	.150	.062	.168	.034	.099	.075
E_9	mean	.181	.225	.257	.247	.251	.190	.342	.256	.109
	SD	.127	.123	.078	.192	.091	.079	.329	.075	.045
E_{36}	mean	.350	.415	.272	.120	.267	.200	.347	.245	.077
	SD	.382	.283	.062	.063	.171	.136	.208	.230	.008

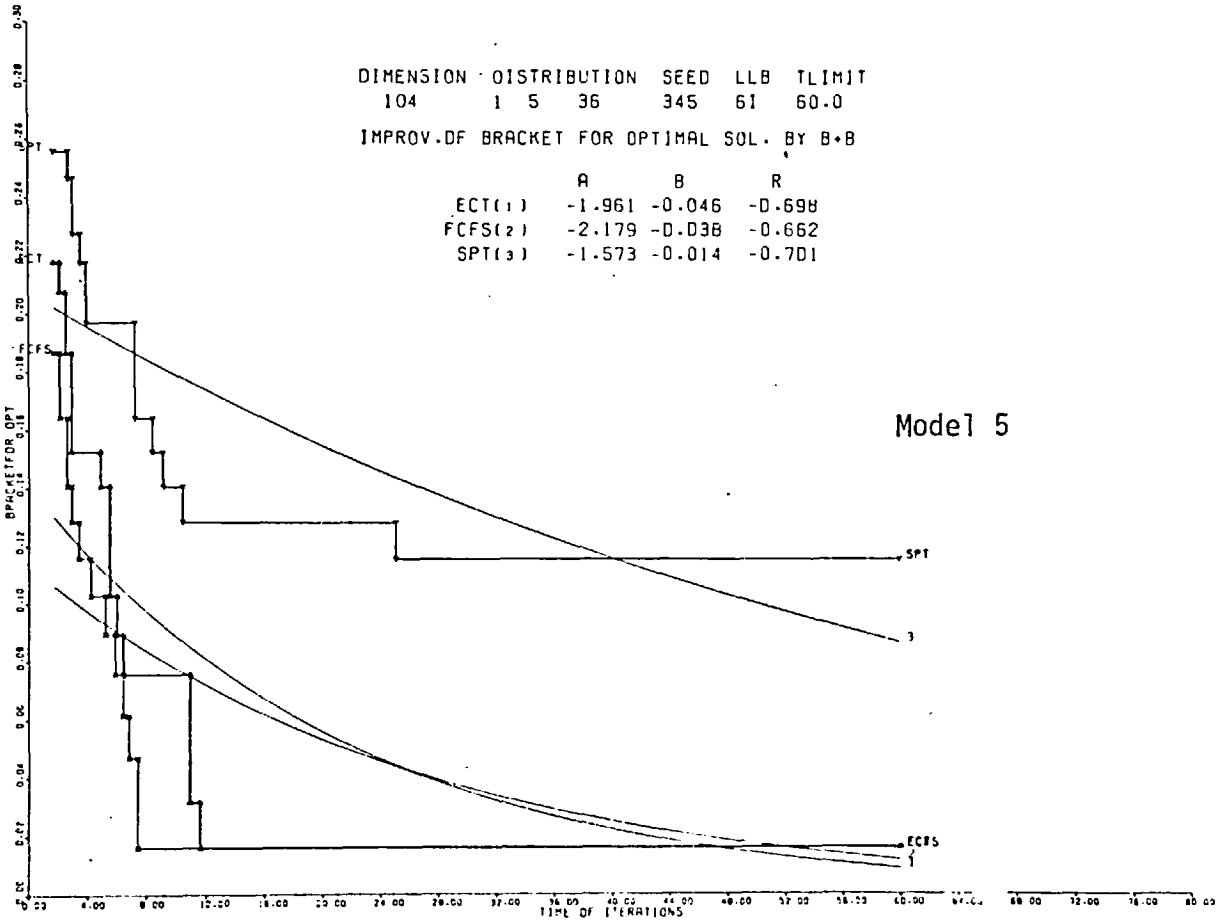
Figure C1 Models for continuous approximation of the step function of the improvements for BOS.

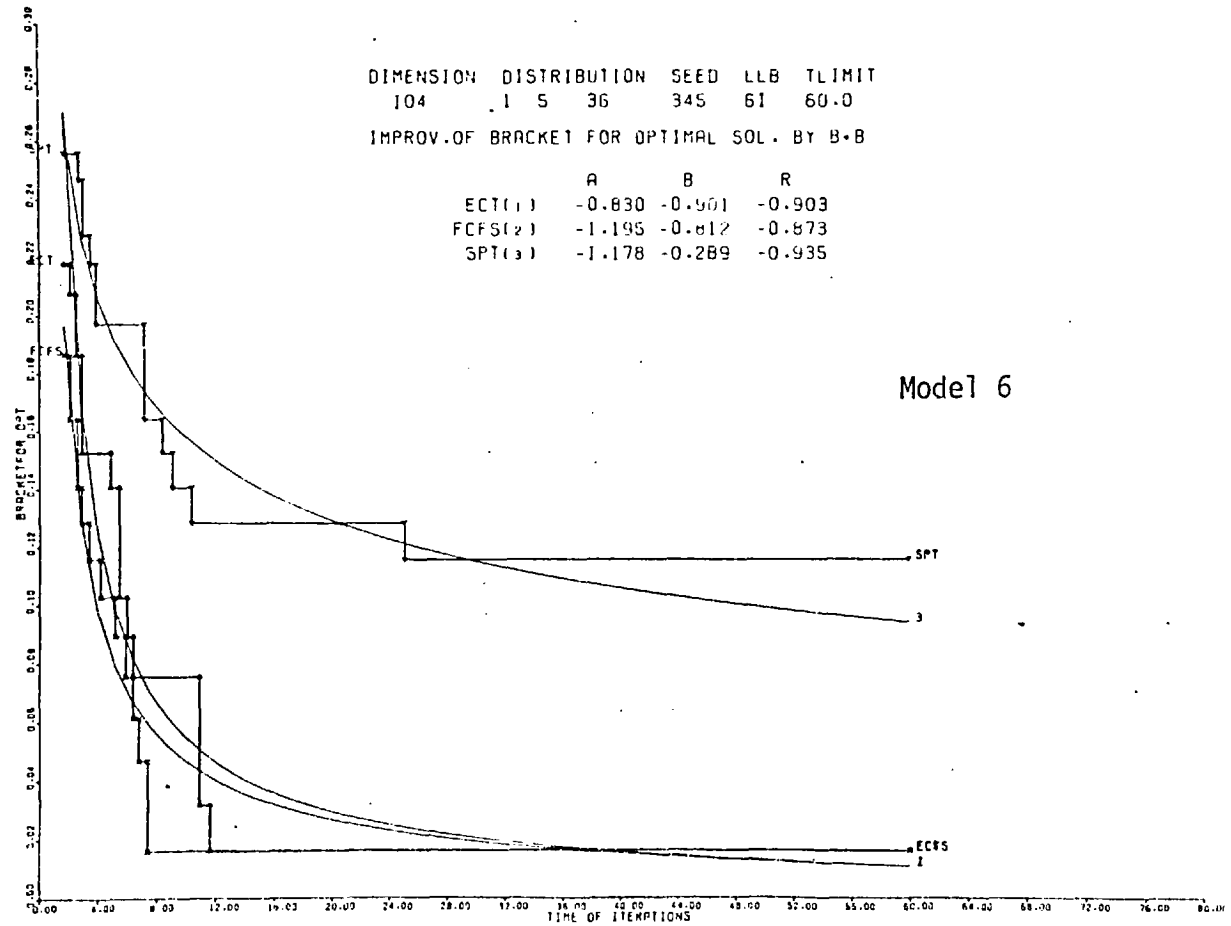





DIMENSION DISTRIBUTION SEED LLB TLIMIT
104 I 5 36 345 61 60.0
IMPROV.OF BRACKET FOR OPTIMAL SOL. BY B+B
A B R
ECT(1) -1.711 -0.076 -0.896
FCFS(2) -1.952 -0.070 -0.902
SPT(3) -1.532 -0.018 -0.790
Model 3
BRACKETFOR OPT
TIME OF ITERATIONS
SPT
ECT
FCFS
3
2


DIMENSION DISTRIBUTION SEED LLB TLIMIT
104 I 5 36 345 61 60.0
IMPROV.OF BRACKET FOR OPTIMAL SOL. BY B+B
A B R
ECT(1) -0.705 -0.954 -0.969
FCFS(2) -1.047 -0.882 -0.985
SPT(3) -1.146 -0.301 -0.967
Model 4
BRACKETFOR OPT
TIME OF ITERATIONS
SPT
ECT
FCFS
3
2


DIMENSION DISTRIBUTION SEED LLB TLIMIT
104 1 5 36 345 61 60.0
IMPROV.OF BRACKET FOR OPTIMAL SOL. BY B*B
A B R
ECT(1) -1.961 -0.046 -0.698
FCFS(2) -2.179 -0.038 -0.662
SPT(3) -1.573 -0.014 -0.701
Model 5
SPT
3
ECFS
1
TIME OF ITERATIONS


DIMENSION DISTRIBUTION SEED LLB TLIMIT
104 1 5 36 345 61 60.0
IMPROV.OF BRACKET FOR OPTIMAL SOL. BY B*B
A B R
ECT(1) -0.830 -0.901 -0.903
FCFS(2) -1.195 -0.812 -0.873
SPT(3) -1.178 -0.289 -0.935
Model 6
SPT
3
ECFS
1
TIME OF ITERATIONS

APPENDIX D

Table D1

Table of critical values of D in the Kolmogorov-Smirnov one-sample test

Sample size N	Level of significance for D = maximum $\lvert F_o(X) - S_N(X) \rvert$				
	.20	.15	.10	.05	.01
1	.900	.925	.950	.975	.995
2	.681	.720	.776	.842	.929
3	.565	.597	.642	.703	.828
4	.494	.525	.564	.624	.733
5	.446	.474	.510	.565	.669
6	.410	.486	.470	.521	.618
7	.381	.405	.433	.486	.577
8	.358	.381	.441	.457	.543
9	.330	.360	.388	.432	.514
10	.322	.343	.368	.410	.490
11	.307	.326	.352	.391	.468
12	.295	.313	.338	.375	.450
13	.284	.302	.325	.361	.433
14	.274	.292	.314	.349	.418
15	.266	.283	.304	.338	.404
16	.258	.274	.295	.328	.392
17	.250	.266	.286	.318	.381
18	.244	.259	.278	.309	.371
19	.237	.252	.272	.301	.363
20	.231	.246	.264	.294	.356
25	.21	.22	.24	.27	.32
30	.19	.20	.22	.24	.29
35	.18	.19	.21	.23	.27
Over 35	$\frac{1.07}{\sqrt{N}}$	$\frac{1.14}{\sqrt{N}}$	$\frac{1.22}{\sqrt{N}}$	$\frac{1.36}{\sqrt{N}}$	$\frac{1.63}{\sqrt{N}}$

APPENDIX E

Approximate formulae for complex queueing systems with general queueing discipline.

$$\mu W(GI/G/r) \to (V_a^2+\rho^2 V_s^2)/2\rho(1-\rho)r \text{ for } \rho\to 1$$

(Kingman, 1970)

$$W(E_k/E_\ell/r) \simeq (1-V_a^2)\,(1-V_s^2)\,W(D/D/r) + (1-V_a^2)\,V_s^2 W(D/M/r) + (1-V_s^2)\,V_a^2\,W(M/D/r) + V_a^2 V_s^2 W(M/M/r)$$

(Page, 1972)

$$W(GI/M/r) \simeq \left\{\frac{2V_a^2}{1+V_a^2}\,\frac{W(M/M/r)}{W(M/M/r')} + \frac{1-V_a^2}{1+V_a^2}\,\frac{W(D/M/r)}{W(D/M/r')}\right\} W(GI/M/r')$$

(Cosmetatos, 1974)

$$W(GI/G/r) \simeq \frac{2(V_a^2+V_s^2) - (1-\rho)\,(1-V_a^2)\,\{V_s^2(3-V_s^2) + 4V_a^2(1-V_s^2)\}}{4} W(M/M/r)$$

(Rosenshine and Chandra, 1975)

$$W(GI/G/I) \simeq \frac{(1+V_s^2)\,(V_a^2+\rho^2 V_s^2)\rho}{2(1-\rho)\,(1+\rho^2 V_s^2)}\,\frac{1}{\mu}$$

(Marchal, 1976)

$$W(M/G/r) \simeq V_s^2\,W(M/M/r) + (1-V_s^2)\,W(M/D/r)$$

(Cosmetatos, 1976)

Expansion of the distribution function of processing times.

For a distribution function f(x) describing the processing times in a job-shop, the moment generating function is defined as:

$$M(t) = \int_{-\infty}^{\infty} e^{tx} f(x)\, dx$$

This can be expressed also as a MacLaurin series:

$$M(t) = M(o) + \frac{t}{1'} M'(o) + \frac{t^2}{2'} M''(o) + R_2$$

where R_2 represents the remaining part of the series, after the second moment:

$$R_2 = (t^3/3')\, M'''(\theta t) \qquad o<\theta<1$$

$$M''(o) = \sigma^2+\mu^2$$

thus $$M(t) = 1 + t\,\mu + (t^2/2)\,(\sigma^2+\mu^2) + R_2$$

Mean and variance of aggregate distribution of processing times in a network of queues.

In the model used, the number of operation per job (M) is from a uniform rectangular distribution between 1 and 5. Each operation is from an Erlang distribution E_k, the same for all machines. The mean μ and variance σ^2 of the aggregate distribution of processing times are:

$$\mu = E(P_{ij})\ E(M)$$

$$\sigma^2 = VAR(P_{ij})\ E(M) + \{E(P_{ij})\}^2\ VAR(M)$$

For the model used

$$E(M) = 3$$

$$VAR(M) = \frac{5^2-1}{12} = 2$$

and for Erlang distributed processing times

$$VAR\ (P_{ij}) = \{E(P_{ij})\}^2/k$$

and $$V^2=\sigma^2/\mu^2 = (2+3/k)\ /\ 9$$

For $k=1$ $\quad (V^2)_M = 5/9$

$k=\infty$ $\quad (V^2)_D = 2/9$

Table E1 Simulation results for Normally distributed processing times

(Q=35 ρ=0.8 μ=20)

Scheduling Rule	Standard Deviation σ 0.20	2.00	3.65	5.16	6.67	8.94	10.00	11.54	14.14	20.0
					AVERAGE WAITING TIME					
SI	243	228	222	217	212	206	204	202	203	227
SI*	215	210	206	202	197	195	195	195	196	222
FIFO	243	243	244	245	247	252	254	260	275	355
					MEAN QUEUE SIZE					
SI	3.2	3.0	3.0	2.9	2.8	2.7	2.7	2.7	2.7	3.0
SI*		2.9	2.8	2.7	2.7	2.6	2.6	2.6	2.6	3.0
FIFO	3.2	3.2	3.2	3.3	3.3	3.4	3.4	3.5	3.7	4.7
					AVERAGE LATENESS					
SI	183	163	162	157	152	146	143	142	142	162
SI*		155	150	146	142	137	135	134	135	158
FIFO	183	183	184	185	187	194	194	199	214	290

Table E2 Average waiting time for Erlang distributed processing times

Batch Size Q	Load Factor ρ	Parameter of Erlang distribution (k) 1	2	3	4	5	9	15	30	∞
					SI					
1	0.8	110	90	87	85	82	84	82	85	82
10	0.6	63	56	55	54	54	54	54	55	
10	0.8	117	103	100	99	98	103	101	105	101
10	0.9	201	192	200	197	192	222	220	227	
35	0.6	163	179	187	190	194	202	207	211	235
35	0.7	173	184	191	195	197	205	210	213	235
35	0.8	200	204	213	214	215	225	229	232	243
35	0.85	227	230	238	240	240	256	261	263	262
35	0.9	273	282	295	298	298	329	332	340	315
35	0.95	400	437	482	456	474	528	560	573	458
35	0.95	363	412	467	444	436	512	526	544	458
35	0.95 Aver.	382	425	475	450	455	520	543	560	458
					SI*					
1	0.8	136	113	103	102	98	96	91	93	81
10	0.8	141	122	116	113	113	113	111	112	102
35	0.8	197	199	206	206	208	215	219	219	220
					FIFO					
1	0.6	75	52		41		35			27
1	0.8	228	157	136	128	119	109	101	98	82
10	0.8	249	179	155	148	140	150	123	120	101
35	0.8	348	293	283	272	267	262	258	255	243

Table E3 Average queue size for Erlang distributed processing times

Batch Size Q	Load Factor ρ	Parameter of Erlang processing times distribution (k) 1	2	3	4	5	9	15	30	∞
					SI					
1	0.8	1.46	1.20	1.15	1.13	1.09	1.12	1.09	1.14	1.09
10	0.6	0.63	0.56	0.55	0.54	0.54	0.54	0.54	0.55	
10	0.8	1.56	1.37	1.32	1.31	1.31	1.37	1.35	1.40	1.34
10	0.9	3.02	2.90	3.01	2.96	2.89	3.33	3.30	3.40	
35	0.6	1.63	1.79	1.86	1.91	1.94	2.01	2.06	2.10	2.34
35	0.7	2.01	2.14	2.22	2.27	2.30	2.39	2.44	2.48	2.73
35	0.8	2.66	2.71	2.83	2.84	2.86	2.99	3.04	3.08	3.23
35	0.85	3.21	3.25	3.37	3.40	3.40	3.61	3.69	3.71	3.70
35	0.9	4.09	4.23	4.43	4.46	4.45	4.92	4.97	5.09	4.71
35	0.95	6.42	6.94	7.72	7.24	7.53	8.38	8.89	9.15	7.22
35	0.95	6.06	6.72	7.73	7.18	7.20	8.16	8.41	8.71	7.24
					SI*					
1	0.8	1.81	1.50	1.37	1.36	1.30	1.28	1.21	1.24	1.08
10	0.8	1.88	1.62	1.54	1.50	1.50	1.50	1.47	1.49	1.36
35	0.8	2.62	2.65	2.74	2.74	2.76	2.85	2.91	2.91	3.00
					FIFO					
1	0.6	0.75	0.52		0.40		4.34			0.27
1	0.8	3.04	2.09	1.82	1.70	1.58	1.45	1.35	1.30	1.09
10	0.8	3.31	2.38	2.06	1.97	1.86	1.73	1.60	1.60	1.34
35	0.8	4.63	3.89	3.76	3.61	3.55	3.48	3.42	3.39	3.23

Table E4 Average lateness (missed due dates) for Erlang distributed processing times

Batch Size Q	Load Factor ρ	Parameter of Erlang processing times distribution (k) 1	2	3	4	5	9	15	30	∞
					SI					
1	0.8	48	28	25	23	21	23	21	24	22
10	0.6	2	-5	-6	-7	-7	-7	-7	-6	
10	0.8	56	42	38	38	37	41	40	44	41
10	0.9	139	131	139	136	131	160	158	165	
35	0.6	102	118	125	121	133	141	146	150	175
35	0.7	111	123	129	133	136	143	148	152	175
35	0.8	138	142	151	152	154	164	167	170	183
35	0.85	166	168	177	179	180	194	200	201	202
35	0.9	211	220	234	237	236	267	271	279	255
35	0.95	339	375	421	394	413	467	498	512	398
35	0.95	302	351	406	382	374	451	465	482	398
					SI*					
1	0.8	74	51	42	41	37	35	29	32	21
10	0.8	79	61	54	52	51	52	49	51	42
35	0.8	136	138	145	144	146	153	157	158	162
					FIFO					
1	0.6	14	-9		-21			-27		-33
1	0.8	167	96	75	66	58	48	40	37	22
10	0.8	187	117	94	87	78	68	61	59	41
35	0.8	287	231	221	210	206	201	196	194	183

Table E5 Effects of inaccuracy of processing times estimates.
(Normal distribution about expected values)

V_s (Expected Processing Times)	Vr (Real Processing Times)	Average waiting time FIFO	SI	SI*
0	0	243	243	220
(E_∞)	0.5	261	261	240
	1.0	388	388	375
0.5	0	271	214	206
(E_4)	0.5	303	236	229
	1.0	516	370	363
1.0	0	347	200	197
(E_1)	0.5	458	240	240
	1.0	938	410	414

Table E6 Similarity of job shops with identical machines in parallel.

	Load factor ρ										
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.85	0.9	0.95
μ_1	234	234	234	234	234	234	234	244	263	314	457
μ_3	50	50	50	50	50	50	50	56	66	90	143
μ_5	15	15	15	15	15	15	17	25	36	47	84
μ_3/μ_1	0.205	0.205	0.205	0.205	0.205	0.205	0.205	0.230	0.250	0.286	0.312
μ_5/μ_1	0.064	0.064	0.064	0.064	0.064	0.064	0.073	0.103	0.136	0.149	0.181
σ_1	122	122	122	122	122	122	122	127	144	185	276
σ_3	29	29	29	29	29	29	29	34	42	59	90
σ_5	14	14	14	14	14	14	17	19	26	34	55
σ_3/σ_1	0.238	0.238	0.238	0.238	0.238	0.238	0.238	0.268	0.291	0.319	0.326
σ_5/σ_1	0.115	0.115	0.115	0.115	0.115	0.115	0.139	0.149	0.180	0.184	0.199
L_1	0.39	0.78	1.17	1.56	1.95	2.34	2.73	3.23	3.70	4.71	7.22
L_3	0.25	0.50	0.75	1.00	1.24	1.49	1.75	2.25	2.87	4.09	6.81
L_5	0.13	0.26	0.39	0.52	0.65	0.78	1.03	1.72	2.56	3.64	6.70
L_3/L_1	0.641	0.641	0.641	0.641	0.636	0.636	0.641	0.645	0.776	0.868	0.943
L_5/L_1	0.33	0.33	0.33	0.33	0.33	0.33	0.37	0.53	0.69	0.77	0.93

μ_r Average waiting time for r identical machines is parallel

σ_r Standard deviation of waiting times for r identical machines is parallel

L_r Average queue length for r identical machines is parallel

Table E7 Average waiting time from simulation and approximate formulae

Q=1, ρ=0.8

Erlang parameter k	FIFO			SI			SI*		
	Wsim	Wapprox	ε%	Wsim	Wapprox	ε%	Wsim	Wapprox	ε%
1	228	-	-	110	-	-	136	-	-
2	157	155	1.3	90	96	6.7	113	109	-3.5
3	136	131	-3.7	87	91	4.6	103	99	-3.9
4	128	119	-7.0	85	89	4.7	102	95	-6.9
5	119	111	-6.7	82	88	7.3	98	92	-6.1
9	109	98	-10.0	84	85	1.2	96	87	-9.4
15	101	92	-8.9	82	84	2.4	91	85	-6.6
30	98	87	-11.2	85	83	-2.3	93	83	-10.7
∞	82	-	-	81	-	-	81	-	-

Q=35, SI

	ρ = 0.7			ρ = 0.85			ρ = 0.9		
1	173	-	-	227	-	-	273	-	-
2	184	204	11	230	244	6.0	282	294	4.2
3	191	214	12	238	250	5.0	295	301	2.0
4	195	219	12	240	253	5.4	298	304	2.0
5	198	222	12	241	254	5.8	298	307	3.0
9	205	228	11	256	255	-0.4	329	310	-5.8
15	210	231	10	261	260	-0.4	332	312	-6.0
30	213	233	10	263	261	-0.8	340	314	-7.6
∞	235	-	-	262	-	-	315	-	-

Table E7 (continued) Global composite scheduling rule

Q=1

	Coefficients		W_M	W_D	$W_{k=2}$			$W_{k=9}$		
ρ	w_1	w_2			SIM	APPROX	ε%	SIM	APPROX	ε%
0.8	30	30	178	74	130	126	-3.0	91	86	-5.5
0.8	0	0	191	72	135	131	-3.0	89	85	-4.5
0.8	100	0	231	87	164	159	-3.0	113	103	-8.8
0.8	33	1	182	72	131	127	-3.0	89	84	-5.6
0.6	30	30	63	27	49	45	-8.1	34	31	-8.8
0.6	0	0	69	27	51	48	-5.9	34	32	-5.9
0.6	100	0	68	28	51	48	-5.9	35	32	-8.5
0.6	33	1	67	27	50	47	-6.0	34	31	-7.3

Q=10

	Coefficients		W_M	W_D	$W_{k=2}$			$W_{k=9}$		
ρ	w_1	w_2			SIM	APPROX	ε%	SIM	APPROX	ε%
0.8	0	0	211	83	149	147	-1.3	105	97	-7.6
0.8	30	30	183	90	141	137	-3.2	110	100	-9.1
0.8	100	0	251	108	183	177	-1.9	131	124	-5.3
0.8	33	1	190	86	144	138	-4.2	106	98	-7.5
0.6	0	0	89	50	69	69	0.0	58	54	-6.9
0.6	30	30	80	50	67	65	-3.0	57	53	-7.0
0.6	100	0	91	52	72	72	0.0	59	56	-4.6
0.6	33	1	84	50	67	67	0.0	57	54	-5.3

Q=35

	Coefficients		W_1	W_∞	W_2			W_9		
ρ	w_1	w_2			SIM	APPROX	ε%	SIM	APPROX	ε%
0.8	0	0	305	199	247	252	2.0	219	211	-3.6
0.8	30	30	257	213	239	235	-1.7	225	218	-3.1
0.8	100	0	339	216	272	277	2.0	229	229	0.0
0.8	33	1	272	192	236	232	-1.7	208	201	-3.4
0.6	0	0	225	188	205	206	0.7	201	192	-4.5
0.6	30	30	196	202	199	199	0.0	209	201	-3.8
0.6	100	0	198	203	196	200	2.3	205	202	-1.2
0.6	33	1	199	179	191	189	-1.0	191	181	-5.2