

1976

A generalized scheduling technique for the multi-machines--multiple facilities system with an application in truck dispatching

AFM Anwarul Haque
Iowa State University

Follow this and additional works at: <https://lib.dr.iastate.edu/rtd>

 Part of the [Industrial Engineering Commons](#)

Recommended Citation

Haque, AFM Anwarul, "A generalized scheduling technique for the multi-machines--multiple facilities system with an application in truck dispatching " (1976). *Retrospective Theses and Dissertations*. 5744.
<https://lib.dr.iastate.edu/rtd/5744>

This Dissertation is brought to you for free and open access by the Iowa State University Capstones, Theses and Dissertations at Iowa State University Digital Repository. It has been accepted for inclusion in Retrospective Theses and Dissertations by an authorized administrator of Iowa State University Digital Repository. For more information, please contact digirep@iastate.edu.

INFORMATION TO USERS

This material was produced from a microfilm copy of the original document. While the most advanced technological means to photograph and reproduce this document have been used, the quality is heavily dependent upon the quality of the original submitted.

The following explanation of techniques is provided to help you understand markings or patterns which may appear on this reproduction.

1. The sign or "target" for pages apparently lacking from the document photographed is "Missing Page(s)". If it was possible to obtain the missing page(s) or section, they are spliced into the film along with adjacent pages. This may have necessitated cutting thru an image and duplicating adjacent pages to insure you complete continuity.
2. When an image on the film is obliterated with a large round black mark, it is an indication that the photographer suspected that the copy may have moved during exposure and thus cause a blurred image. You will find a good image of the page in the adjacent frame.
3. When a map, drawing or chart, etc., was part of the material being photographed the photographer followed a definite method in "sectioning" the material. It is customary to begin photoing at the upper left hand corner of a large sheet and to continue photoing from left to right in equal sections with a small overlap. If necessary, sectioning is continued again — beginning below the first row and continuing on until complete.
4. The majority of users indicate that the textual content is of greatest value, however, a somewhat higher quality reproduction could be made from "photographs" if essential to the understanding of the dissertation. Silver prints of "photographs" may be ordered at additional charge by writing the Order Department, giving the catalog number, title, author and specific pages you wish reproduced.
5. PLEASE NOTE: Some pages may have indistinct print. Filmed as received.

University Microfilms International

300 North Zeeb Road
Ann Arbor, Michigan 48106 USA
St. John's Road, Tyler's Green
High Wycombe, Bucks, England HP10 8HR

77-10,316

HAQUE, AFM Anwarul, 1941-
A GENERALIZED SCHEDULING TECHNIQUE FOR THE
MULTI-MACHINES--MULTIPLE FACILITIES
SYSTEM WITH AN APPLICATION IN TRUCK
DISPATCHING.

Iowa State University, Ph.D., 1976
Engineering, industrial

Xerox University Microfilms, Ann Arbor, Michigan 48106

A generalized scheduling technique for the multi-machines--multiple
facilities system with an application in truck dispatching

by

AFM Anwarul Haque

A Dissertation Submitted to the
Graduate Faculty in Partial Fulfillment of
The Requirements for the Degree of
DOCTOR OF PHILOSOPHY

Department: Industrial Engineering

Major: Engineering Valuation

Approved:

Signature was redacted for privacy.

In Charge of Major Work

Signature was redacted for privacy.

For the Major Department

Signature was redacted for privacy.

For the Graduate College

Iowa State University

Ames, Iowa

1976

TABLE OF CONTENTS

	Page
I. INTRODUCTION	1
II. SCHEDULING ALGORITHM	15
III. STOPPING RULES	53
IV. SINGLE-MACHINE ANALYSIS	66
V. MULTIPLE-MACHINE ANALYSIS	98
VI. MINIMUM BOUND ESTIMATION	139
VII. TRUCK ROUTING ALGORITHM	152
VIII. SUMMARY	193
IX. BIBLIOGRAPHY	198
X. ACKNOWLEDGEMENTS	202
XI. APPENDIX I. LEFT-SHIFTING ALGORITHM	203
XII. APPENDIX II. TRUCK ROUTING ALGORITHM	211

I. INTRODUCTION

A. Introduction

The problem of sequencing has been the subject of extensive research in recent years. In its general context, the sequencing problem is the problem of defining order (rank, priority, and the like) over a set of jobs (tasks, items, commodities) as they proceed from one machine (processor, facility, operation) to another or over the same machine. Thus, the sequencing problem involves the determination of the relative position of job j to all other jobs. Moreover, a sequence is obtained when a complete ordering of the jobs is given.

"Sequencing" is often used synonymously with "scheduling." The scheduling problem is to find the order in which jobs should be processed at each machine and their start and finish times at each machine. In the deterministic case, no distinction is made between sequencing and scheduling because, in the process of finding the best sequence, a schedule is automatically generated since it is always assumed that each activity is started as early as possible (Elmaghraby, 1968a).

Sequencing problems arise quite naturally in various activities of everyday life. In the area of operations research, one of the most classical examples is the problem of sequencing n jobs on m machines in which each job has its individual route (i.e., the order of operations on the job usually imposed by technological requirements, such as a hole cannot be tapped before it is drilled, etc.), which may be distinct from all others. The feasible sequence obtained thereof should

result in an optimal or near-optimal performance for the shop with respect to selected criteria.

B. Models

Different sequencing problems naturally lead to different models, which implies differences in the three basic constituents of a model: (i) parameters, (ii) assumptions, and (iii) criteria.

The realm of the job sequencing problem can be decomposed into two general groups: those in which job arrivals are considered to be static and those in which job arrivals are dynamic, that is, varying over time. In this dissertation, we will confine ourselves to the static or deterministic system.

In general, with respect to job sequencing problems, systems are divided into those with a single processor and those with multiple processors. We will be concerned with the latter system.

Multi-processor systems exhibit almost unlimited varieties of arrangements of facilities and of flow of work through the facilities. In general, the following list of typical simplifying assumptions is made:

1. Assumptions concerning jobs

a. All n jobs are simultaneously available at the beginning of the planning period.

b. Each job is an entity, even though the job is composed of individual parts. This eliminates "job splitting" between two or more

machines. It also eliminates assembly operations.

c. No job may be on two processors at the same time. This rules out "lap-phasing" in which the same job is started on a second operation as soon as a few units are available from the first operation.

d. The job routing is given and no alternate routings are permitted.

e. No pre-emption is allowed.

f. The processing time of each job is known and deterministic.

g. Processing times and setup times are independent of the sequence.

h. All jobs are of equal importance.

2. Assumptions concerning machines

a. All m machines are available at the beginning of the planning period and are ready to take up any of the jobs.

b. At most one job can be processed on a specific machine at any given time. This eliminates processes such as ironworker, heat treating ovens, chemical treatment tanks, all of which are commonplace processes that handle multiple jobs simultaneously.

c. There is only one machine of each type in the shop.

3. Others

a. In-process inventory is allowed.

b. Due-dates, if they exist, are fixed.

In this dissertation, assumptions 2b and 2c concerning machines have been removed. In Chapter 2, an algorithm is developed for the multi-

machines multiple facilities system. In Chapter 7, the algorithm is modified to remove assumption 2b. Chapter 8 includes a brief discussion employing other assumptions.

C. Criteria

In sequencing problems, the number of feasible schedules is so large that it is uneconomical to enumerate all the feasible schedules for any problem beyond relatively small ones. All feasible schedules are not equally desirable. A firm has limited resources and the management of the firm usually wishes to minimize cost. A few of the important costs are:

- (a) operation costs,
- (b) machine idle costs,
- (c) job waiting costs (in-process inventory costs),
- (d) penalty costs if the jobs are late

The sum total of the above four costs is called the "cost of production", and the scheduling problem is to find that feasible schedule which minimizes the cost of production. "Generally, research workers in the field of scheduling have adopted a simple measure of performance, called the make-span (or maximum flow time) as representative of the cost of production" (Gupta, 1971). Maximum flow time is defined as the elapsed time between the start of the first job in the sequence on the first machine. In this dissertation the maximum flow time criterion is used as a measure of performance. Chapter 8 presents a brief discussion on the evaluation of this criterion.

D. Methodologies

There are basically four different approaches used in solving sequencing problems: (a) combinatorial, (b) general mathematical programming, (c) reliable heuristics, and (d) Monte Carlo sampling (Elmaghraby, 1968a).

1. Combinatorial approaches

Combinatorial approaches rely on changing one permutation to another through the "switching around" of jobs that satisfy a given criterion. The fundamental concept in this approach is best expressed by the following theorems due to Smith (1956).

Theorem 1: Let $f(Q)$ be the measure of any permutation Q of jobs $\{1, 2, \dots, n\}$. It is desired to minimize its value. Let Q^* be a given permutation. Then a sufficient condition that $f(Q^*) \leq f(Q)$ for all permutations Q of the n job is that:

(i) There is a real valued function of ordered pairs of elements such that if Q is any permutation and Q' is the permutation obtained from Q by interchanges of the i th and $(i+1)$ st elements, then $f(Q) \leq f(Q')$ if $g(q_i, q_{i+1}) \leq g(q_{i+1}, q_i)$,

(ii) Q^* is such that the i th job precedes the j th job if $g(i,j) < g(j,i)$.

Theorem 2: Let Q be any permutation of the n jobs, $Q \equiv (1, 2, \dots, i, j, \dots, n)$ and let $f(Q)$ be a bounded function of satisfying condition (i) of Theorem 1 if $f(Q)$ can be represented as

$f(Q) = g(i,j) \circ \Phi\{Q - (i,j); g(i,j)\}$, where the symbol "o" denotes a binary operation which preserves inequalities on the real line, $\{Q - (i,j)\}$ denotes the sequence Q minus the pair i and j , in that order, and Φ is a monotone function of $g(i,j)$.

The proof of this theorem is given by Elmaghraby (1968b). By definition, the operation "o" satisfies the relationship; if a, b , and c are three points on the real line, and if $a \geq b$, then $a \circ c \geq b \circ c$.

If $Q' \equiv (1, 2, \dots, j, i, \dots, n)$, then $g(j,i)$ is \geq $g(i,j)$.

Suppose that $g(j,i) \geq g(i,j)$; then we have

$$\begin{aligned} f(Q') &= g(j,i) \circ \Phi \{Q' - (j,i); g(j,i)\} \\ &\geq g(i,j) \circ \Phi \{Q' - (j,i); g(j,i)\} \text{ by property of "o"} \\ &\geq g(i,j) \circ \Phi \{Q - (i,j); g(i,j)\} \text{ by property of "o" and} \\ &\quad \text{monotonocity of } g \\ &= f(Q). \end{aligned}$$

This combinatorial approach was the basis for the papers by Jackson (1956), Johnson (1954), and Smith (1956).

2. General mathematical programming

General mathematical programming includes linear, dynamic, quadratic and convex programming, integer programming, networks of flow, Lagrangian methods, and the like.

Bowman (1959), Wagner (1959), Manne (1960), and Balas (1967) regard the scheduling problem as a conventional programming

problem and suggest a linear programming solution with integer constraints on its solution. Based on the structural arguments of the problem, some integer constraints are listed and the objective function is defined as the minimization of the maximum flow time. This formulation of scheduling problems is sound from a mathematical viewpoint. However, due to excessive computational requirements, the cost of obtaining results is prohibitive even for small sized problems (Story and Wagner, 1963).

Dynamic programming has also been used with some success in formulating and solving the combinatorial problems. But in most instances, this formulation has served as a means of demonstrating its effectiveness only in small size problems.

3. Reliable heuristics

Reliable heuristics are sometimes known as "combinatorial programming" or "controlled enumeration." In essence, they are problem-solving procedures developed on the basis of two principal concepts: the use of a controlled enumeration technique for considering all potential solutions and the elimination from explicit consideration of particular solutions which are known from dominance, bounding, and feasibility considerations to be unacceptable.

An alternate name for such programs is branch-and-bound (Elmaghraby, 1968a) the name given to the ideas employed by Little, Murry, Sweeney and Karnel (1963) in their algorithms for solving the travelling salesman problem. The "branch" notion stems from the fact that in terms of

a tree the procedure is continually concerned with choosing a branch of the tree to elaborate and evaluate. The "bound" term denotes their emphasis on the effective use of means for bounding the value of the objective function at each node of the tree both for eliminating dominated parts and for selecting a branch for elaboration and evaluation.

The efficiency of the branch-and-bound procedure largely depends on lower bounds used in the process of generating schedules (Gupta, 1970; Nabeshima, 1967b). The papers by Ignal and Schraze (1965), Nabeshima (1967a), Gupta (1970), Ashour (1970) describe several bounding procedures which can be used in selecting a branch. However, the efficiency of the algorithm does not necessarily increase with the increase in the effectiveness of lower bounds because of excessive computations required in calculating the bounds (Ashour and Quraishi, 1969; Nabeshima 1967a; Gupta, 1970).

The two important points emphasized by Elmaghraby (1968a) can be mentioned here. First, reliable heuristic approaches guarantee the eventual finding of an optimal solution. "Second, such approaches need not have their upper limit on the number of alternatives generated in the complete space of all possible sequences. In some instances, the upper limit is much smaller than that, such as the number of alternatives that would be generated by some mathematical programming approaches. (The problem treated by Elmaghraby and Cole (1963) has its upper bound on the nodes of the decision tree the 2^n nodes of the dynamic programming formulation of the same problem, which is much smaller than the complete space of all possible sequences, which is $n!$). In these

instances, the approach through reliable heuristics can never do worse (from a computing point of view), and in all probability will do much better, than the best mathematical programming approach!"

4. Monte Carlo

Monte Carlo experimentation has been used in two somewhat different applications.

The natural "habit" of the Monte Carlo technique is evidently in stochastic systems, i.e., in systems in which the parameters vary in a probabilistic fashion over time. Since we will be concerned only with the deterministic sequencing system, our discussion on Monte Carlo approach will be made in a completely different context.

In Monte Carlo methods, schedules are generated at random and after "enough" schedules have been obtained, the best of them is retained. Chances of achieving a near-optimal solution are good if enough schedules are generated.

The primary advantage of a Monte Carlo approach is that a solution for a large complex problem can be obtained in a reasonable length of time, especially with the development of computer technology. Almost any sequencing problem can be approached by Monte Carlo methods since these procedures are easily stated and readily understood. Such a method allows "tentative" results to be obtained early for new sequencing problems without regard to the availability of algebraic solution procedures, constraints, or algorithm convergence.

A Monte Carlo version of the Giffler-Thompson program (1960)

does not guarantee the finding of an optimal schedule, but it does permit the computation of a fairly large number of feasible schedules chosen at random in a reasonable amount of time; consequently, the shortest one found can then be chosen. By continuing the process long enough it is possible to make the probability of not having observed an optimal schedule very small. Of course, the cost of computation must also be considered.

Heller and Logemann (1962) describe a Monte Carlo approach from a graph-theoretic point of view, as opposed to the Gantt-chart approach which motivated Giffler and Thompson (1960). This approach can generate schedules more rapidly than Giffler and Thompson procedure. "This time estimate is about half that tentatively reported by Giffler and Thompson" (Heller and Logemann, 1962).

Heller and Logemann (1962) indicated that a potential area "for further research" was the modification to give a better subset of feasible schedules by using the concept of left-shifting which permits an operation to "jump over" another operation into an interval of the time if that interval is large enough to accommodate the shifted operation. This is the starting point of the research for this dissertation. The Monte Carlo algorithm that we will develop will be for multi-machine facilities with the concept of left-shifting incorporated. This algorithm will be further modified to relax assumption 2 concerning machines as mentioned in section B. This modified algorithm can then be applied, as we shall see in Chapter 7, for transportation problems.

Two factors pose potential severe limitations on the utility of Monte Carlo methods for solving sequencing problems: the efficiencies of the algorithms and the rules for halting the sampling processes.

The random generation of sequences can be quite inefficient since such sequences are often far from optimal. Therefore, much computational effort is spent in generating sequences of mediocre value. However, neighborhood search techniques, as proposed by Reiter and Sherman (1965) tend to increase the efficiency of the Monte Carlo techniques by searching exhaustively all sequences in the neighborhood of a given solution. Various dispatch rules, such as those described by Conway, Maxwell and Miller (1967), are also available for improving the efficiency of Monte Carlo methods. Giffler, Thompson and Van Ness (1963) showed that a Monte Carlo process that uses rules as guides in its random choices will be considerably superior to a purely random choice device. In fact, Fisher and Thompson (1963) devised some learning strategies to guide the program in its use of rules. In this dissertation, biasing techniques are used to improve efficiency of the randomly generated solutions.

Another important aspect of the Monte Carlo approach is the need for rules to stop the sampling procedures. This problem has been recognized by many. In this respect, Heller's (1960) observation is worthwhile to mention. "By combinatorial means rooted partly in a lattice-theory framework, it has been shown that although there are many possible

schedules there are far fewer different schedule times. Because of the relatively small number of different schedule times for any given set of processing times, we might expect the probability distribution of the schedule times over the set of all schedules to have a simple form." He concludes that "the numerical experiments show that the distribution of schedule times is normal; theoretical analysis indicates that the schedule times are asymptotically normally distributed for schedules with a large number of jobs." The argument continues that this normality can be used to determine decision-theoretical rules to terminate sampling when the cost of continued sampling exceeds the expected gain from further sampling. This argument motivated different researchers such as Elmaghraby (1968a) to define closed form stopping rule functions. As we shall see in Chapter 3, Elmaghraby (1968a) himself casts some arguments against his own stopping rule function. There is some question about the asymptotic normality of the distribution, but aside from that there are at least two obstacles that lie in the path of a practical procedure (Conway, Maxwell and Miller, 1967):

(a) If the distribution of schedule-times is approximately and/or asymptotically normal, the departures from normality will be most pronounced in the tails of the distribution, which is precisely the area of interest. No one is concerned with estimating the mean of the schedule time distribution for a particular problem.

(b) It is inconceivable that anyone would be sampling from this distribution, for there are obviously more efficient subpopulations readily available.

By Bayesian analyses, Randolph (1968) proposed a multinomial stopping rule for Monte Carlo sampling. Different aspects of this rule have been presented in Chapter 3.

In the study of the behavior of the extreme points in samples drawn from a bounded population, the theory of extreme values plays an important role. "Based on the assumption that the observed phenomena can be described by the limiting distributions of greatest (or smallest) values in random occurrences, the theory has been applied to a variety of engineering and astronomical problems" (Bae, 1972). An application of the theory to the combinatorial problems first appeared in plant layout problems (McRoberts, 1971). This will be discussed in Chapter 3 and Chapter 6.

In Chapter 3, we have suggested some distribution-free stopping rules for halting the sampling procedure. These stopping rules which are easily understandable have been incorporated in the algorithm presented in this dissertation.

D. Summary of Research Objectives

In Chapter 2, a Monte Carlo algorithm will be developed for the multi-machines multiple facilities system. Left-shifting principle will be incorporated in the algorithm to get a better subset of the feasible solutions. The algorithm will work irrespective of whether the machines at a particular work center are either identical or not.

Chapter 3 discusses the different distribution-free stopping rules to be used in the algorithm for halting the Monte Carlo sampling.

Chapters 4 and 5 present analyses of different sample problems. Different parameters such as minimum schedule time, sample size needed, CPU time, distribution of the schedule time, etc., will be examined. The effects of the left-shifting principle and different biasing techniques in improving the efficiency of Monte Carlo sampling will be studied.

Chapter 6 gives some statistical analysis of the solutions. The main parameters of interest are the estimated minimum and the probability of further improvement of the solution. The Weibull distribution will be used as a tool to estimate the minimum bound value of the schedule time.

Chapter 7 shows a modification of the algorithm in Chapter 2 which is applied to a transportation-type problem. This modified version is also applicable to a system where one or more machines can process different jobs simultaneously.

Chapter 8 presents some guidelines for further research.

II. SCHEDULING ALGORITHM

A. Introduction

This chapter will deal with the actual development of an algorithm to minimize maximum flow time for multi-machine facilities with the concept of left-shifting incorporated in the algorithm. The following simple problem will be considered for illustration.

Consider five jobs and three facilities. Each of two facilities, A and B, has two machines with different efficiencies for doing the same operation. The two machines in facility A are A1 and A2, and those in B are B1 and B2. Facility C has only one machine. The problem has technological orderings and processing times as given in Tables 2.1 and 2.2., respectively.

Table 2.1. Technological orderings

Job	Operations					
	1		2		3	
1	A1	A2	C		A1	A2*
2	B1	B2	A1	A2		
3	C		B1	B2	C	
4	A1	A2	B1	B2		
5	C		A1	A2		

Table 2.2. Processing times

Job	Operations					
	1		2		3	
1	2	3	6		4	7
2	8	9	2	5		
3	3		3	4		
4	3	4	6			
5	2		5	7		

For the sake of illustration of the algorithm, A, B and C will refer to facilities 1, 2 and 3, respectively. A1 and A2 will refer, respectively, to machines 1 and 2 of facility 1. Similarly, B1 and B2 denote the machines 1 and 2 in facility B.

Let us now denote each of the operations by a quadruple of integers $(ijkl)$, where

i: facility

j: machine within facility

k: job to be processed

g: gth time job k is in machine j of facility i.

For example, Table 2.1 refers to operation number 3 of job 1 in machine A2 marked *. Here,

facility: A => i=1

machine: A2 => j=2

job: 1 => k=1

second time job 1 in facility A $\Rightarrow \ell = 2$
 so, the operation will be denoted by (1212).

B. Computer Aspects of the Algorithm

In section D, we will see that we have to pass through different iterations to come up with a feasible solution. Each iteration contains seven columns. The entries of some of the columns will remain the same and some will change as we proceed with the algorithm. Every time, before we start to generate a feasible solution, the entries of all the columns are to be changed to their initial values.

The different columns have the following interpretations.

Table 2.3. Columns 1 and 2

Column 1	Column 2
Operations designation	Operation/operations (if any) that immediately follow operations in column 1

Operations in columns 1 and 2 will be put into the notation $ijkl$ as mentioned in section A. Column 1 will contain all the operations, some of which will become inactive as the algorithm proceeds and will never be executed. This is because some operations are mutually exclusive in multi-machine facilities.

The number of operations in column 2 is 0, 1, or more than 1, depending on whether the operation in column 1 is, respectively, the

last operation of any job or the following operation will be processed in a single or multi-machine facility.

Table 2.4. Columns 3, 4 and 5

Column 3	Column 4	Column 5
Switch indicating the state of the operation in column 1	Processing time of the operation in column 1	Index signifying the number of times machine has been used

The entries in column 3 are 1, 0 and -1 indicating the state of the operation in column 1 in the following manner:

1 => not scheduleable

0 => scheduleable

-1 => scheduled

Initially, all the entries in column 5 are zero.

Table 2.5. Columns 6 and 7

Column 6	Column 7
Starting time of the operation in column 1	Completion time of the operation in column 1

Initially, all the entries in columns 6 and 7 are zero. It is to be noted here that the entries in columns 5, 6 and 7 can have realistic

meanings only when the corresponding operation in column 1 has been processed.

The different important arrays used in the computer program are :

IC1: which job to be processed
 IC2: which job must follow
 IC3: switches of column 3 (1, 0, -1)
 IC4: processing time
 IC5: indexes indicating number of times the machines have been engaged
 IC6: starting time of the operations
 IC7: completion time of the operations
 ISAVE: the best solution found so far from sampling
 IDT: total processing time for each sample
 ICS3: track on the initial setting in IC3
 IND: number of entries in IC2.

IC1 and IC2 remain the same during the entire sampling process. IC3 changes its value, but its initial setting is stored in ICS3. IC5, IC6, and IC7 are initially set to zero before sampling begins. IFIND is the row corresponding to the operation selected for processing, and MJM refers to the maximum job time.

The program can handle 500 operations and generate 1500 schedules. Steps of the computer program are as follows:

Step 1. Resolving or initializing: Resolve the data and make the proper entry to each of the arrays from IC1 to IC7. If any schedule

is already found, reinitialize all the entries.

Step 2. Randomization in selecting a process: Randomization procedures will be discussed in section C. Check IC3 to see if there is a zero in any row. If no zero entry is found, a feasible schedule has been obtained and step 6 follows. Otherwise, count the number of zeros and select one of the processes at random. IFIND is the row selected by randomization. Turn the switches from 0 (or 1) to -1 corresponding to IFIND and its counterparts.

Step 3. Job time and machine time: Calculate maximum job time MJM and maximum machine time corresponding to IFIND. If generation of schedules by left-shifting is desired, go to step 9; otherwise, process the operation at $\max(\text{Maximum machine time and MJM})$. This is C_6 .

Step 4. Adjustments after processing: Put C_6 in IC6. To this value add the entry in IC4 and put it in IC7. Find the maximum entry ($\text{MAXM} = \text{IMAX}$) in IC5 corresponding to this machine; increase it by 1 and put it in column 5. Note that all these changes are made with respect to IFIND.

Step 5. Operations to follow: Check IC2. If there is no entry corresponding to IFIND, go to step 2. Otherwise, count the number of operations (NC) that are associated with IC2. Select one operation at random. Turn the switch on in IC3 from 1 to 0 for this operation. Go to step 2.

Step 6. Schedule time: Find the maximum value in IC7. This is the schedule time C_7 . If this is less than or equal to the best previous C_7 , write out this tableau.

Step 7. Stopping rule: Update NPROB (number of feasible solutions) by 1. If NPROB is not a multiple of the specified sample size, go to step 1. Otherwise, call the desired stopping rule and check the criterion. If it is met, step 8 follows; otherwise, go to step 1.

Step 8. Printout: Print all of the processing times found so far in descending order. The program is terminated at this point.

Left shifting:

Step 9: Check IC5 and find maximum number of times the machine is used (MAXM). If MAXM = 0, process the operation starting at MJM and go to step 4; otherwise, step 10 follows.

Step 10: Check C_6 and C_7 corresponding to the i th entry of IC5 corresponding to machine in IFIND. If $C_{7i} \leq MJM$, then check whether $i = MAXM$. If it is, process job at MJM and go to step 4. Otherwise, go to the next entry ($i = i+1$) and repeat the process until $C_{7i} > MJM$. Now, if $C_{6i} \leq MJM$, go to step 12. Otherwise, step 11 follows.

Step 11: Check the interval between MJM and the starting point of the job corresponding to i th entry, i.e.,

$$IAVA = C_{6i} - MJM$$

Check IC4 and find the processing time INTV. If $IAVA \geq INTV$, then increase all the positive entries except less than i in IC5 by 1, and then process the job in that interval starting at MJM and go to step 4 to make the necessary adjustments; otherwise, step 12 follows.

Step 12: Check if i th entry is equal to MAXM. If it is, process the job at maximum machine time and go to step 4. Otherwise, go to the next entry and find the interval

$$IAVA = C_{6i+1} - C_{7i}.$$

If $IAVA \geq INTV$, increase all the positive entries greater than i in IC5. All these changes correspond to the machine in IFIND. Process the job in that interval and go to step 4 to make the necessary adjustments. If $IAVA < INTV$, go to the next entry of IC5 and continue the process. If nothing is found up to and including MAXM, then set IOUT to 0 (no left shifting is possible) and proceed to process the job at max (Maximum machine time and MJM) and go to step 4.

C. Random Selection of a Process

There are two procedures by which random selection of a process can be accomplished. Both procedures generate numbers by using an available computer routine called RANDU (IX, IY, XK), where

IX = starting number

IY = a number generated by the subroutine

After each operation, IX is given a value equal to IY in order to have a different starting number in a later call statement.

$XX =$ a random number generated between 0 and 1

Procedure 1: Let us assume that we have three processes and we have to select one at random. By using the subroutine, XX is generated with the result being

$XX < 1/3 \Rightarrow$ select process 1

$XX \in (1/3, 2/3) \Rightarrow$ select process 2

$XX \geq 2/3 \Rightarrow$ select process 3

Procedure 2: The random selection of any process in this dissertation has been based on a procedure different from the above.

Let us suppose the number of processes is b . After generating XX as before a new value of XX , XX^* , is calculated as follows:

$$XX^* = XX(b-1) + 1.5$$

where XX is a real random variable uniformly distributed between 0 and 1. A process is selected corresponding to the value of XX^* which is an integer obtained after dropping the decimal part in XX^* .

D. Discussion of Iteration Procedure

Referring to the discussion in the previous sections, let us now proceed to apply the algorithm to the problem given in section A. CL_1 , CL_2 , CL_3 , CL_4 , CL_5 , CL_6 and CL_7 refer to the seven columns of the tableaux in different iterations.

In Table 2.6 the problem has been defined completely in algorithmic notations, showing the technological orderings between the operations. The processing times have been entered in column 4.

Table 2.6. Iteration 0

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	0	2	0	0	0
1112		1	4	0	0	0
1121		1	2	0	0	0
1141	2141, 2241	0	3	0	0	0
1151		1	5	0	0	0
1211	3111	0	4	0	0	0
1212		1	7	0	0	0
1221		1	5	0	0	0
1241	2141, 2241	0	4	0	0	0
1251		1	7	0	0	0
2121	1121, 1221	0	8	0	0	0
2131	3132	1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	0	9	0	0	0
2231	3132	1	5	0	0	0
2241		1	7	0	0	0
3111	1112, 1212	1	6	0	0	0
3131	2131, 2231	0	2	0	0	0
3132		1	4	0	0	0
3151	1151, 1251	0	2	0	0	0

To start the scheduling procedure of the example problem, we see that certain operations are scheduleable; i.e., there are 0 entries in CL3 corresponding to these operations.

Iteration 1: From the scheduleable operations, 1111, 1141, 1211, 1241, 2121, 2221, 3131, 3151, let us randomly select 1211. We turn the switches in IC3 corresponding to 1211 and its counterpart 1111 from 0 to -1.

Job 1 was not processed, and the machine was not engaged previously. So

$$\max (\text{Maximum machine time and MJM}) = \max (0,0) = 0$$

which means starting time is 0, and the completion time is $0 + C_4 = 0 + 4 = 4$. The index corresponding to IFIND in IC5 is increased by 1 which updates MAXMJ to 1.

The identification in column 2 corresponding to IFIND is 3111. So the switch in IC3 corresponding to 3111 is reset from 1 to 0, showing that the operation is now scheduleable. All these changes have been reflected in Table 2.7.

Iteration 2: Now we have the possibility of scheduling one of the operations 1141, 1241, 2121, 2221, 3111, 3131, 3151. Let us suppose we select 2121. We turn the switches in IC3 corresponding to 2121 and its counterpart 2221 from 0 to -1. Job 2 was not processed, and the machine was not engaged previously. So, $\max (\text{Maximum machine time and MJM}) = \max (0,0) = 0$.

Table 2.7. Iteration 1

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		1	4	0	0	0
1121		1	2	0	0	0
1141	2141, 2241	0	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	4	1	0	4
1212		1	7	0	0	0
1221		1	5	0	0	0
1241	2141, 2241	0	4	0	0	0
1251		1	7	0	0	0
2121	1121, 1221	0	8	0	0	0
2131	3132	1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	0	9	0	0	0
2231	3132	1	5	0	0	0
2241		1	7	0	0	0
3111	1112, 1212	0	6	0	0	0
3131	2131, 2231	0	2	0	0	0
3132		1	4	0	0	0
3151	1151, 1251	0	2	0	0	0

So the starting time is 0 and the completion time is $0 + C_4 = 0 + 8 = 8$. $MAXMJ = 0$, so the index in IC5 is now increased to 1. In column 2 corresponding to IFIND, there are two operations, 1121, 1221. Let us randomly select 1221 and turn the switch in IC3 corresponding to 1221 from 1 to 0. Table 2.8 reflects the above changes.

Iteration 3: Among the scheduleable operations 1141, 1221, 1241, 3111, 3131, 3151, we randomly select 3111 and turn the switches in IC3 as before. Now machine 3 was not previously engaged, but job 1 was processed in machine 1. Therefore,

$$\begin{aligned} & \max (\text{Maximum Machine time and MJM}) \\ & = \max (0, 4) = 4 \end{aligned}$$

So, the starting time is 4 and completion time is $4 + 6 = 10$.

Adjustment in IC5 is made as outlined above. From column 2, we randomly select 1112 and make the necessary adjustment in IC3 corresponding to 1112 as before. These are shown in Table 2.9.

Iteration 4: Among the scheduleable operations 1112, 1141, 1221, 1241, 3131, 3151, we randomly select 3131. Here $MJM = 0$ and C_6 corresponding to the 1st operation on the machine is 4. So $IAVA$ (gap) = $C_6 - MJM = 4 - 0 = 4$.

$$INTV = C_4 (\text{IFIND}) = 2$$

So, $IAVA > INTV$. We therefore process the job in the gap corresponding to IAVA and increase the positive entry in IC5 corresponding to this

Table 2.8. Iteration 2

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		1	4	0	0	0
1121		1	2	0	0	0
1141	2141, 2241	0	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	4	1	0	4
1212		1	7	0	0	0
1221		0	5	0	0	0
1241	2141, 2241	0	4	0	0	0
1251		1	7	0	0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	1	5	0	0	0
2241		1	7	0	0	0
3111	1112, 1212	0	6	0	0	0
3131	2131, 2231	0	2	0	0	0
3132		1	4	0	0	0
3151	1151, 1251	0	2	0	0	0

Table 2.9. Iteration 3

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		0	4	0	0	0
1121		1	2	0	0	0
1141	2141, 2241	0	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	4	1	0	4
1212		1	7	0	0	0
1221		0	5		0	0
1241	2141, 2241	0	4	0	0	0
1251		1	7		0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	1	5	0	0	0
2241		1	7	0	0	0
3111	1112, 1212	-1	6	1	4	10
3131	2131, 2231	0	2	0	0	0
3132		1	4	0	0	0
3151	1151, 1251	0	2	0	0	0

machine by 1. Other adjustments are made as usual. This iteration is shown in Table 2.10. In the incomplete Gantt charts¹ shown in figures 2.1 and 2.2, we see the improvement by left shifting accomplished in this iteration.

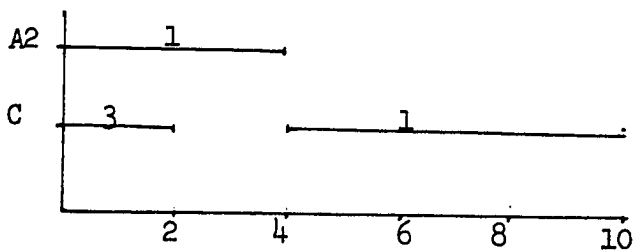


Figure 2.1. Left-shifting

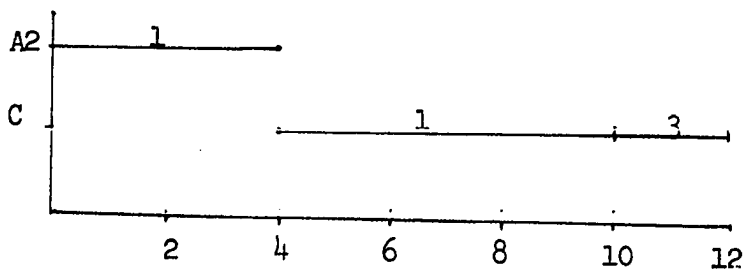


Figure 2.2. Non-left-shifting

¹In the Gantt charts shown in figures 2.1 and 2.2, the horizontal and the vertical axes, respectively, refer to time scale and machines. The numbers above the horizontal bars refer to the job numbers. This notation will be used in subsequent Gantt charts.

Table 2.10. Iteration 4

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		0	4	0	0	0
1121		1	2	0	0	0
1141	2141, 2241	0	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	4	1	0	4
1212		1	7	0	0	0
1221		0	5	0	0	0
1241	2141, 2241	0	4	0	0	0
1251		1	7	0	0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	0	5	0	0	0
2241		1	7	0	0	0
3111	1112, 1212	-1	6	2	4	10
3131	2131, 2231	-1	2	1	0	2
3132		1	4	0	0	0
3151	1151, 1251	0	2	0	0	0

Iteration 5: Among the scheduleable operations 1112, 1141, 1221, 1241, 2131 and 3151, we randomly select 1221 and process this operation as usual. The switch in IC3 corresponding to IFIND changes from 0 to -1 and for 1121 (its counterpart) from 1 to -1. All other adjustments are made as before. This iteration is shown in Table 2.11.

Iteration 6: From the set of scheduleable operations 1112, 114, 1241, 2131 and 3151, 1241 is randomly selected. Here, MAXM = 2 and MJM = 0. We check the first positive entry in IC5 corresponding to IFIND. We check IAVA.

$$IAVA = C_6 - MJM = 0 - 0 = 0$$

and, $IAVA < INTV = C_4 (IFIND) = 2$. This implies that left shifting is not possible and we proceed to the next higher integer in IC5 which is 2. Here C_7 corresponding to entry 1 of IC5 and C_6 corresponding to entry 2 of IC5 are greater than MJM and

$$IAVA (gap) = 8 - 4 = 4 > INTV = 2;$$

so left shifting is possible and we can process this job in that interval. We must also update entry in IC5 by changing the entry 2 to 3. All other adjustments are made as outlined before. This iteration is shown in Table 2.12. Incomplete Gantt charts are shown in figures 2.3 and 2.4.

Table 2.11. Iteration 5

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		0	4	0	0	0
1121		-1	2	0	0	0
1141	2141, 2241	0	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	4	1	0	4
1212		1	7	0	0	0
1221		-1	5	2	8	13
1241	2141, 2241	0	4	0	0	0
1251		1	7	0	0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	0	5	0	0	0
2241		1	7	0	0	0
3111	1112, 1212	-1	6	2	4	10
3131	2131, 2231	-1	2	1	0	2
3132		1	4	0	0	0
3151	1151, 1251	0	2	0	0	0

Table 2.12. Iteration 6

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		0	4	0	0	0
1121		-1	2	0	0	0
1141	2141, 2241	-1	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	0	1	0	4
1212		1	7	0	0	0
1221		-1	5	3	8	13
1241	2141, 2241	-1	4	2	4	8
1251		1	7	0	0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	0	5	0	0	0
2241		0	7	0	0	0
3111	1112, 1212	-1	6	2	4	10
3131	2131, 2231	-1	2	1	0	2
3132		1	4	0	0	0
3151	1151, 1251	0	2	0	0	0

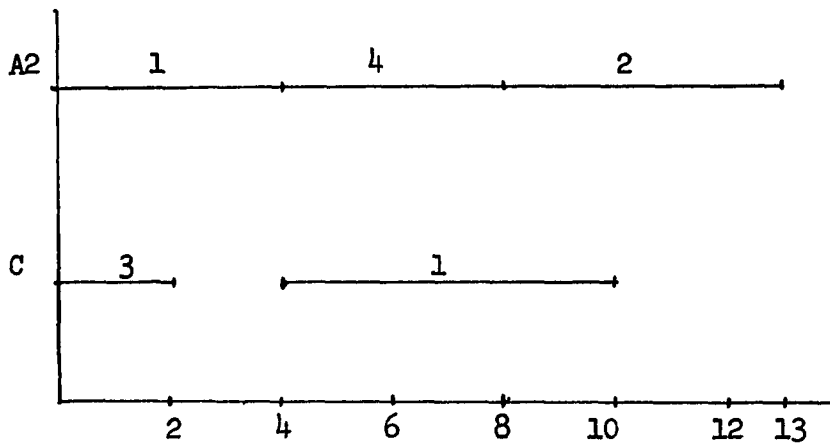


Figure 2.3. Left-shifting

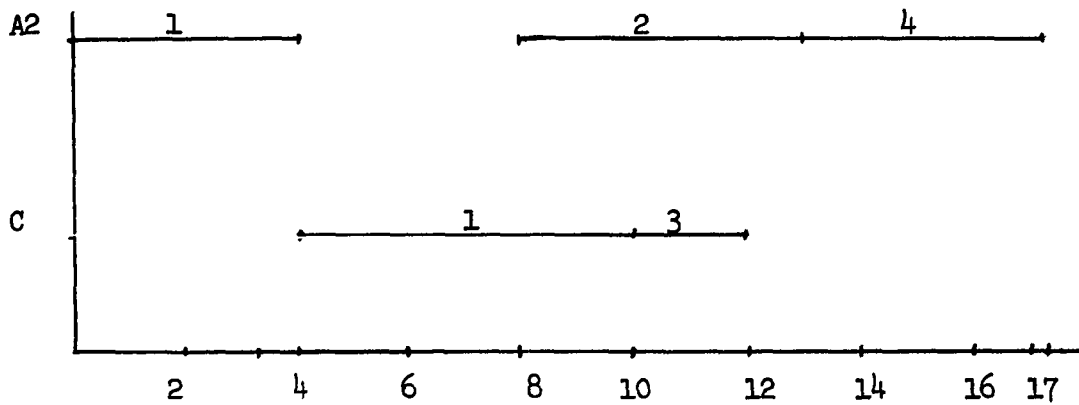


Figure 2.4. Non-left-shifting

Iteration 7: We select randomly 1112 from the scheduleable set of operations 1112, 2131, 2410 and 3151 and process it as before. This iteration is shown in Table 2.13.

Table 2.13. Iteration 7

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		-1	4	1	10	14
1121		-1	2	0	0	0
1141	2141, 2241	-1	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	4	1	0	4
1212		-1	7	0	0	0
1221		-1	5	3	8	13
1241	2141, 2241	-1	4	2	4	8
1251		1	7	0	0	0
2121	1121, 1212	-1	8	1	0	8
2131	3132	1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	0	5	0	0	0
2241		0	7	0	0	0
3111	1112, 1212	-1	6	2	4	10
3131	2131, 2231	-1	2	1	0	2
3132		1	4	0	0	0
3151	1151, 1251	0	2	0	0	0

Iteration 8: We select randomly 2231 from the scheduleable set of operations 2231, 2241 and 3151 and process it as before. Referring to figures 1 and 2, we see that the process starts at time 2 in left shifting and at 12 in non-left shifting. Adjustments have been shown in Table 2.14.

Iteration 9: From the scheduleable set of operations 2241, 3132, 3151, we randomly select 3132 and process this operation. Adjustments are shown in Table 2.15.

Iteration 10: From the scheduleable set of operations 2241 and 3151, we randomly select 2241. Here, $\max(\text{Maximum machine time, MJM}) = \max(7, 8) = 8$. So the process must start at 8 and is completed at 15. All the adjustments are shown in Table 2.16.

Iteration 11: We now select the only possible scheduleable operation 3151. Here $\text{MJM} = 0$ and $\text{MAXM} = 3$. We check the first positive entry in IC5 corresponding to IFIND and check IAVA. Here, $\text{IAVA} = C_6 - \text{MJM} = 0 - 0 = 0$.

So $\text{IAVA} < \text{INTV} = 2$ and no left shifting is possible. We now check IAVA with C_6 corresponding to entry 2 and C_7 corresponding to entry 1 for this machine.

$$\text{IAVA} = C_6 - C_7 = 4 - 2 = 2$$

Now $\text{IAVA} = \text{INTV}$ and left shifting is possible. We process this job in this interval and increase all positive entries in IC5 except 1

Table 2.14. Iteration 8

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		-1	4	1	10	14
1121		-1	2	0	0	0
1141	2141, 2241	-1	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	4	1	-	4
1212		-1	7	0	0	0
1221		-1	5	3	8	13
1241	2141, 2241	-1	4	2	4	8
1251		1	7	0	0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	-1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	-1	5	1	2	7
2241		0	7	0	0	0
3111	1121, 1221	-1	6	2	4	0
3131	2131, 2231	-1	2	1	0	2
3132		0	4	0	0	0
3151	1151, 1251	0	2	0	0	0

Table 2.15. Iteration 9

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		-1	4	1	10	14
1121		-1	2	0	0	0
1141	2141, 2241	-1	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	4	1	0	4
1212		-1	7	0	0	0
1221		-1	5	3	8	13
1241	2141, 2241	-1	4	2	4	8
1251		1	7	0	0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	-1	3	0	0	0
2141		1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	-1	5	1	2	7
2241		0	7	0	0	0
3111	1112, 1212	-1	6	2	4	10
3131	2131, 2231	-1	2	1	0	2
3132		-1	4	3	10	14
3151	1151, 1251	0	2	0	0	0

Table 2.16. Iteration 10

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		-1	4	1	10	14
1121		-1	2	0	0	0
1141	2141, 2241	-1	3	0	0	0
1151		1	5	0	0	0
1211	3111	-1	4	1	0	4
1212		-1	7	0	0	0
1221		-1	5	3	8	13
1241	2141, 2241	-1	4	2	4	8
1251		1	7	0	0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	-1	3	0	0	0
2141		-1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	-1	5	1	2	7
2241		-1	7	2	8	15
3111	1112, 1212	-1	6	2	4	10
3131	2131, 2231	-1	2	1	0	2
3132		-1	4	3	10	14
3151	1151, 1251	0	2	0	0	0

corresponding to this machine. All other adjustments have been made as usual and are shown in Table 2.17. The left shifting is shown in the final Gantt chart, figure 2.5 after the last iteration 12.

Iteration 12: We now select the only scheduleable operation 1151. Here $MJM = 4$ and C_6 corresponding to the first positive entry in $IC5$ is 10. We again must calculate IAVA.

$$IAVA = 10 - 4 = 6 > INTV = C_4 (IFIND) = 5$$

which implies left shifting is possible. We process this last operation in this interval and increase the only positive entry in $IC5$ by 1. All other adjustments have been made as usual and are shown in Table 2.18. We notice that there is no zero entry in $CL3$ which means we have obtained a feasible schedule.

Next we check $IC7$ and see that the highest entry in this array is 15. This represents the total processing time for this schedule by left shifting procedures.

The final Gantt charts are shown in figures 2.5 and 2.6.

Table 2.17. Iteration 11

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		-1	4	1	10	14
1121		-1	2	0	0	0
1141	2141, 2241	-1	3	0	0	0
1151		0	5	0	0	0
1211	3111	-1	4	1	0	4
1212		-1	7	0	0	0
1221		-1	5	3	8	13
1241	2141, 2241	-1	4	2	4	8
1251		1	7	0	0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	-1	3	0	0	0
2141		-1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	-1	5	1	2	7
2241		-1	7	2	8	15
3111	1112, 1212	-1	6	3	4	10
3131	2131, 2231	-1	2	1	0	2
3132		-1	4	4	10	14
3151	1151, 1251	-1	2	2	2	4

Table 2.18. Iteration 12

CL1	CL2	CL3	CL4	CL5	CL6	CL7
1111	3111	-1	2	0	0	0
1112		-1	4	2	10	14
1121		-1	2	0	0	0
1141	2141, 2241	-1	3	0	0	0
1151		-1	5	1	4	9
1211	3111	-1	4	1	0	4
1212		-1	7	0	0	0
1221		-1	5	3	8	13
1241	2141, 2241	-1	4	2	4	8
1251		-1	7	0	0	0
2121	1121, 1221	-1	8	1	0	8
2131	3132	-1	3	0	0	0
2141		-1	6	0	0	0
2221	1121, 1221	-1	9	0	0	0
2231	3132	-1	5	1	2	7
2241		-1	7	2	8	15
3111	1112, 1212	-1	6	3	4	10
3131	2131, 2231	-1	2	1	0	2
3132		-1	4	4	10	14
3151	1151, 1251	-1	2	2	2	4

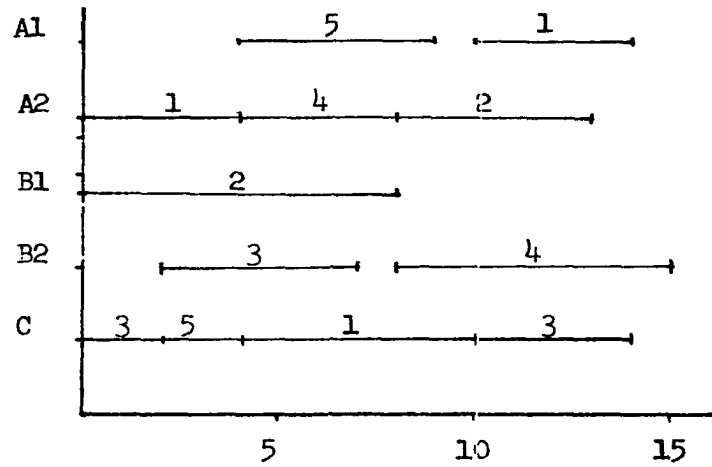


Figure 2.5. Left-shifting

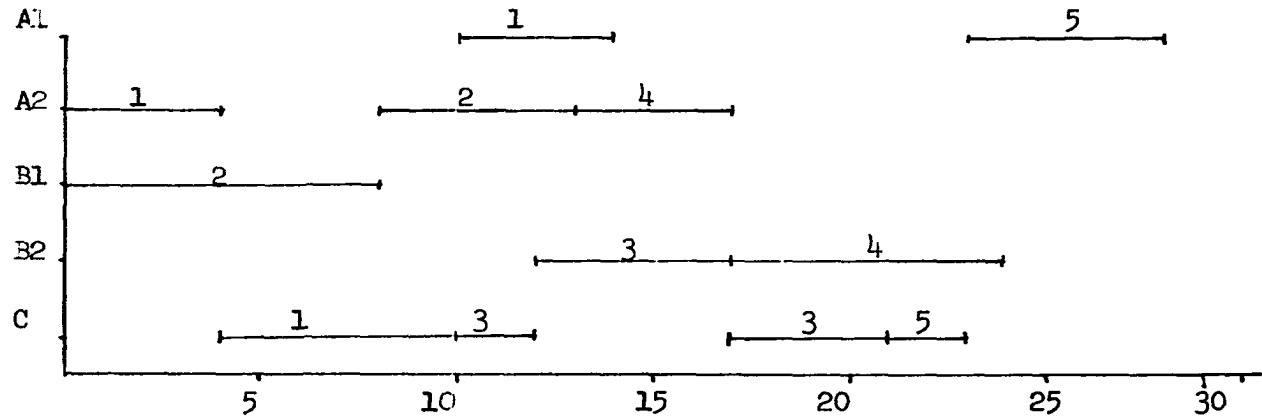


Figure 2.6. Non-left-shifting

D. Input Format Used in the Computer Program

1. Format (4I4)

The first card in the data set includes four numbers.

1st number: total number of possible operations

2nd number: number of schedules (sample size) at multiple of which stopping rules are applied

3rd number: starting random number (must be odd)

4th number: number of machines in the system

2. Format (I2)

The second set of data cards refers to the number of operations in column 2. In this problem, there are 20 numbers corresponding to each operation. Each number is punched in a different data card and is arranged according to the arrangement of operations in column 1.

3. Format (I3, (I6))

The third set of data cards refers to the entries in the first and second columns of any tableau in the algorithm.

4. Format (I2, I2)

The fourth set of data cards refers to the entries in the third and fourth columns of Table 6 (Iteration 0).

E. Output Format of the Computer Output

Referring to Table 18, in the computer output only CL3, CL5, CL6 and CL7 are printed. In CL5, CL6 and CL7 of Table 18, there are some zero entries. It means the operations corresponding to those

entries have never been processed. Because this is a multi-machines facility system, other machines in the facility were used to perform those operations.

F. Input Stream for the Sample Problem

Columns →	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16...80
			2	0			5	0		2	1	7				5
	1															
	0															
	0															
	2															
	.															
	.															
	.															
	2															
	1	1	1	1	3	1	1	1								
	1	1	1	2												
												
												
												
	3	1	5	1	1	1	5	1	1	1	2	5	1			
	0	2														
	1	4														
	.	.														
	.	.														
	.	.														
	0	2														

By changing the format statement we can reduce the number of cards in the input stream to one fourth.

G. Flow Chart of the Main Program

Figure 2.7 depicts the flow chart of the main program discussed in section C. The different biasing techniques and the different stopping rules used for analysis are not included.

In the flow chart, two different types of lines with arrowheads are used. One is solid and the other is dotted. The former refers to the active path indicating the relationship between one block and the next block of the flow chart; whereas. The latter refers to a "dummy" path which merely keeps track of the "ordered" sequence of the two steps.

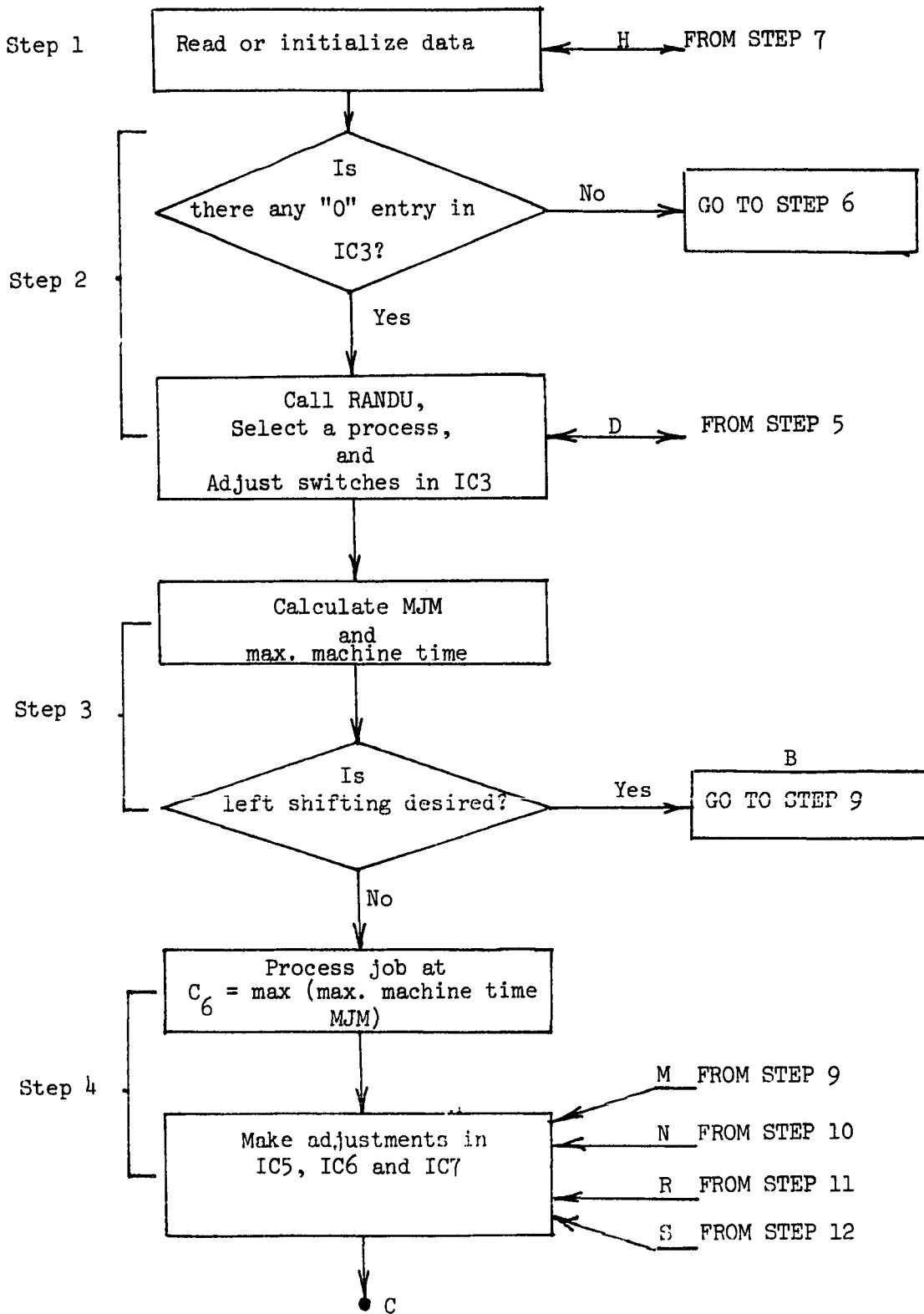


Figure 2.7. Flow chart

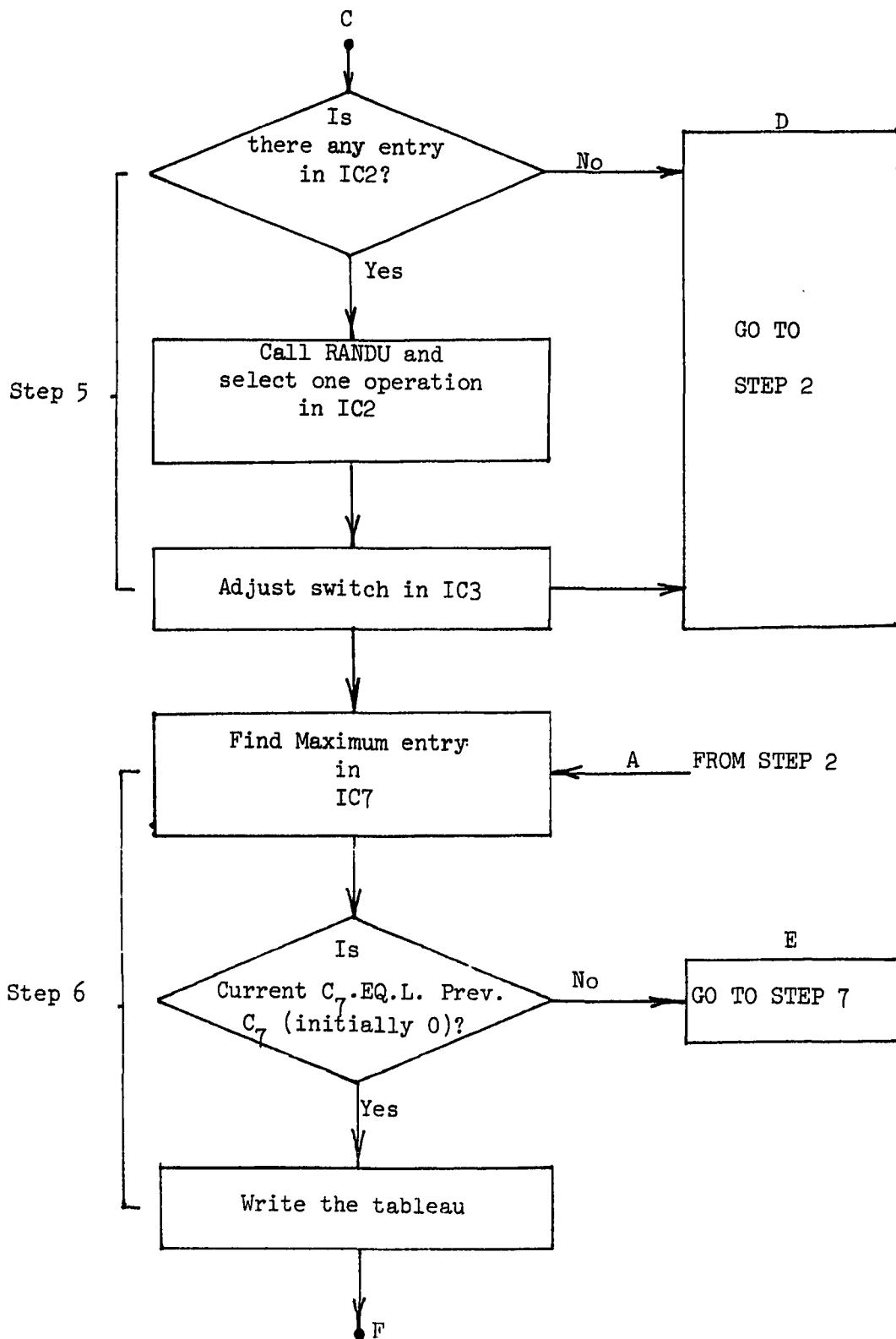


Figure 2.7. (continued)

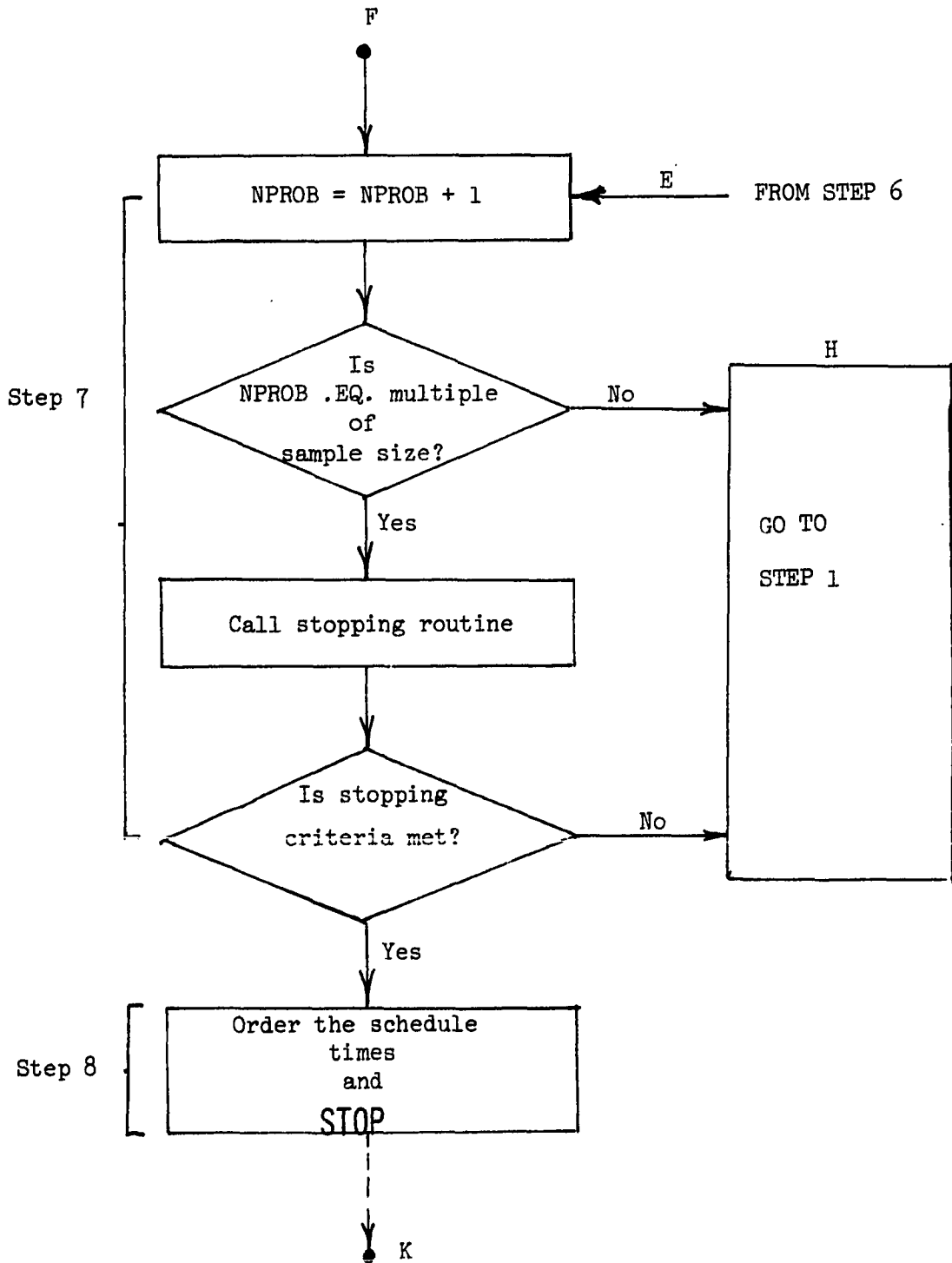


Figure 2.7. (continued)

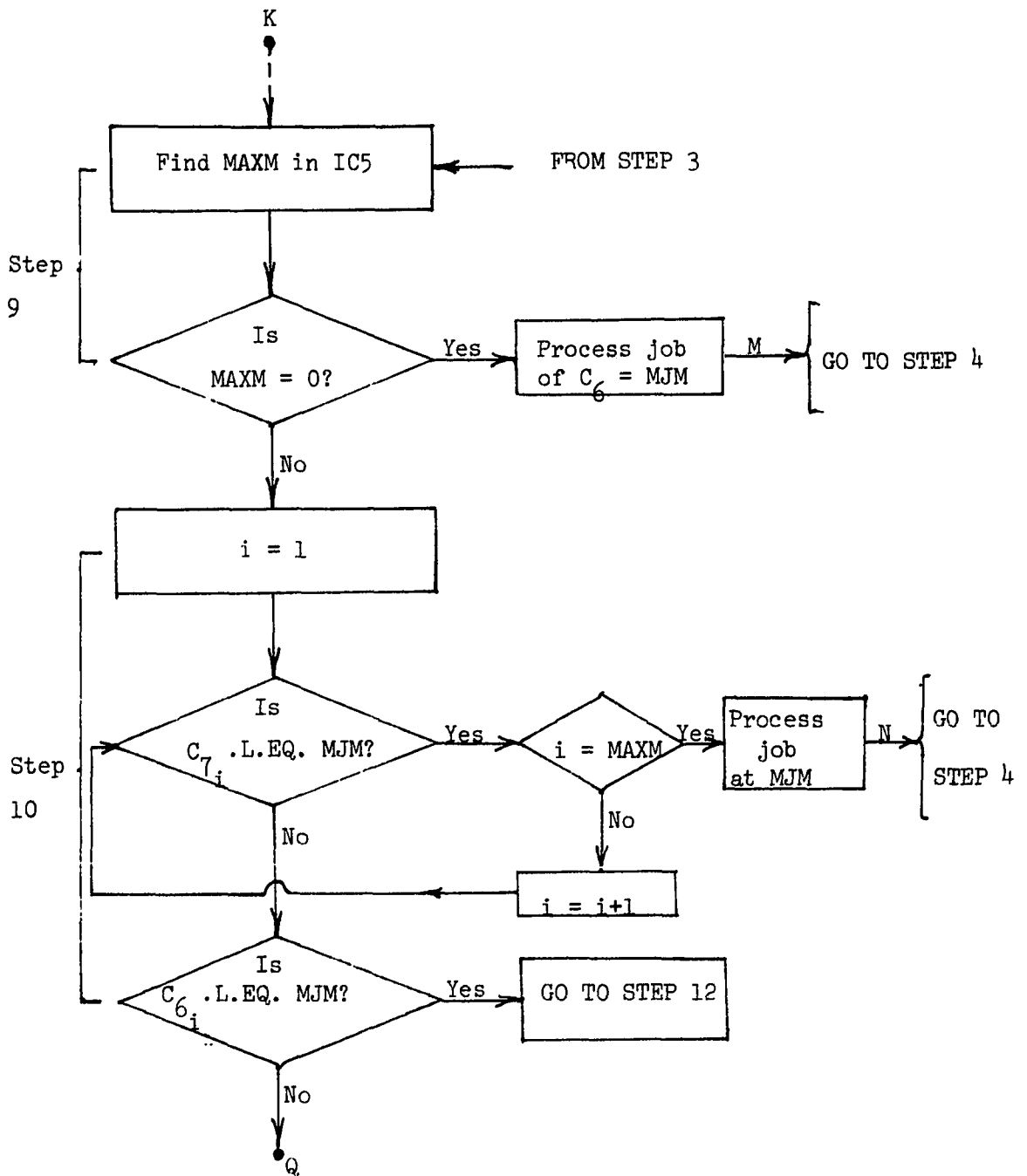


Figure 2.7. (continued)

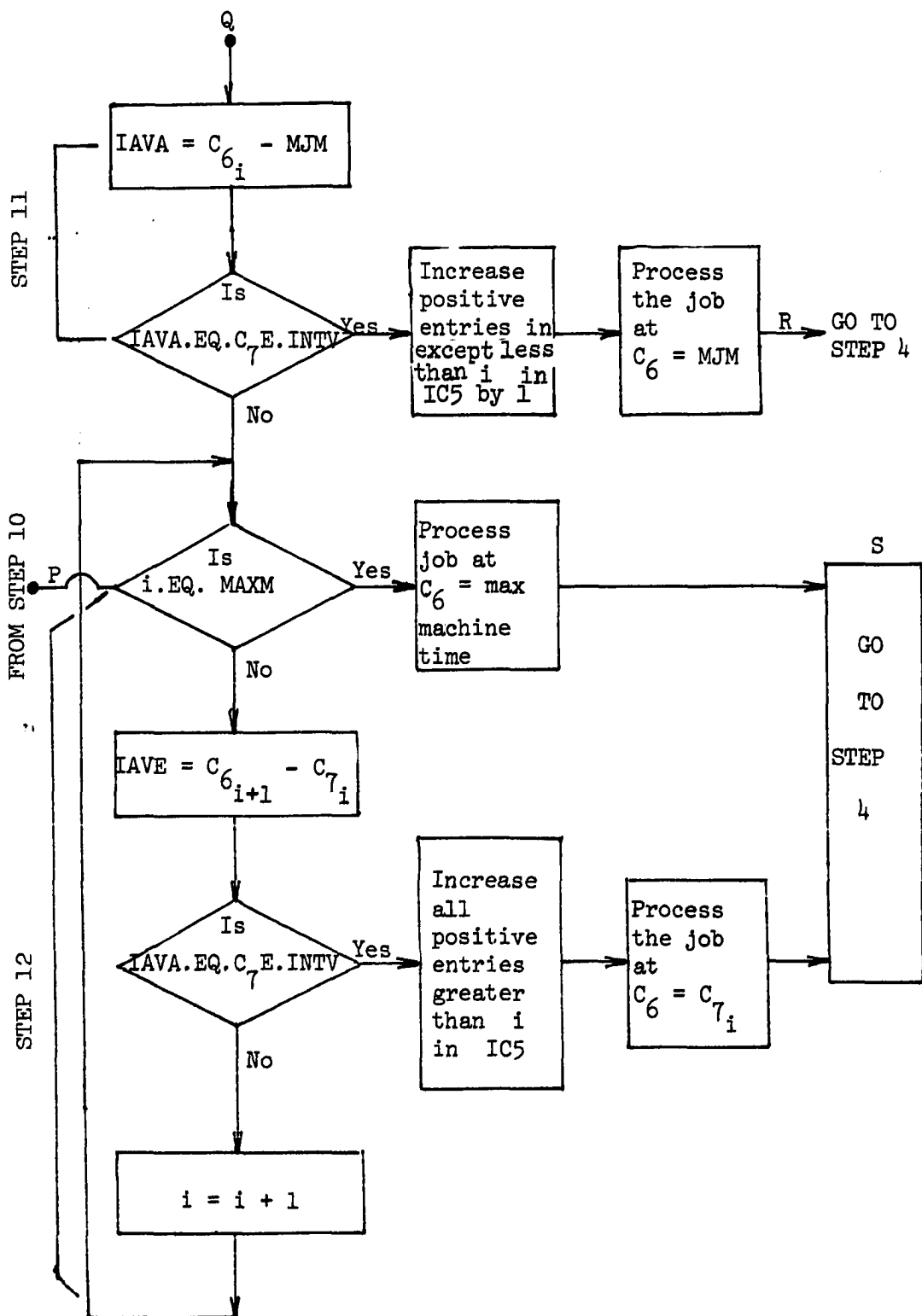


Figure 2.7. (continued)

III. STOPPING RULES

A. Introduction

As we mentioned in Chapter 1, without regard to the availability of algebraic procedures, problem-size constraints, etc., the Monte Carlo method allows "tentative" results to be obtained for almost any sequencing problem. This method is basically characterized by randomly generating sequences and then selecting the best from a large number of such sequences.

For most of the combinatorial problems, the total number of possible combinations is very large. The expression $(n!)^m$ is often cited for the number of schedules for an n job m machine ($n \times m$) problem. To get a feel for the size of $(n!)^m$, it should be sufficient to note that $(6!)^5$ is approximately 1.93×10^{14} , which is more than the number of microseconds in six years. But the above expression provides in general neither a very good estimate nor an upper bound. It is presumably based on the special symmetric problem in which each job has one and only one operation on each machine (Conway, Maxwell and Miller (1967)). Even a present generation computer will not be economically able to examine more than a fraction of the total number of possible combinations for most combinatorial problems. Thus, a sample of the combinations must be selected for examination.

Monte Carlo technique, being a random sampling procedure, has to choose sample points from a large sample set in sequencing problems. Consequently, it poses potential serious limitations on the utility of Monte Carlo methods for solving these problems as to how many samples

to draw from the set before a decision can be arrived at and what is the statistical measure of efficiency in relation to the "optimum" solution. Otherwise, even for a small problem, it is not unlikely that by the same algorithm, different schedules will be generated unless the entire set is sampled. This would require a huge expenditure of resources to generate all possible combinations.

The need for rules to stop Monte Carlo sampling procedures for sequencing problems has been recognized by many. Elmaghraby (1968a) sketched a process for halting Monte Carlo sampling for the job-shop scheduling problem. His paper relied heavily on prior knowledge of the distribution of sequence payoff. The short outline of his approach is given below. To arrive at the optimal stopping rule, the scheduler must construct a "loss function" which expresses dependence on two variables: the minimum schedule time d_n^* actually obtained after n trials, and the changes of obtaining, through random sampling, a better sequence. Let

a = cost of experimentation per experiment

b = measure of utility of a unit of time saved in the duration of the sequences and

$\phi(x)dx$ = normal probability density function of schedule time distribution with the estimated mean μ and standard deviation σ , i.e.,

$$\phi(x)dx = \frac{1}{\sqrt{2\pi}\sigma} \exp \left[\frac{-(x - \mu)^2}{2\sigma^2} \right] dx$$

The expected saving in time from the (n+1)st experiment is given by

$$b \int_{-\infty}^{d_n^*} (d_n^* - x) \phi(x) dx = b d_n^* \Phi\left(\frac{d_n^* - \mu}{\sigma}\right) - b \int_{-\infty}^{d_n^*} x \phi(x) dx$$

where (Φ) is the cumulative standard normal probability function.

Naturally, an experiment will be conducted when the expected marginal gain is greater than the marginal cost, i.e., the extra sample will be taken if

$$b d_n^* \Phi\left(\frac{d_n^* - \mu}{\sigma}\right) - \int_{-\infty}^{d_n^*} x \phi(x) dx > a.$$

Otherwise, experimentation halts, and the best sequence thus far obtained is implemented.

Although Elmaghraby's stopping rule is conceptually valid, it does not have wide applicability as an operational tool. In this regard it is sufficient to mention only two of his own remarks. First, the measure of b is difficult because there is no "a priori" knowledge of where the saving in time is going to occur. Consequently, an average figure over all the machines is to be assumed. Second, "the use of the theoretical density function (on which the stopping rule is based) is naturally crucially dependent on the stability of the 'product mix' from period to period, as well as on the stability of the number of jobs to be sequenced. That is to say, a different set of jobs requiring different processing times (i.e., sample points are drawn from different populations) would lead to a

different density function. On the other hand, if n , the number of jobs to be sequenced, is different from period to period, it is not immediately obvious how the theoretical density function should be modified to cope with such variation" (Elmaghraby, 1963a).

In scheduling a problem, the experimenter is interested in the extreme values, such as to maximize "pay-offs," to minimize make-span time, etc. Assumed distributions may not truly recognize the behavior of extreme values. Therefore, placing total emphasis on an assumed distribution, particularly where techniques for improving efficiency are being applied, can lead to non-optimal stopping rules and inefficient sampling algorithms.

Randolph, Swinson and Ellingsen (1973) developed stopping rules for sequencing problems in which a minimum of assumptions is made about the characteristics of the pay-off distribution. The basic tool employed for their analysis is the sequential Bayesian decision procedure. The approach is briefly described below.

Suppose observations x_1 and x_2 are taken sequentially from a given multinomial population. Each observation costs a finite amount c . After each observation, the experimenter has a choice. He can stop and receive a pay-off which is based on the values of the observations, or he can pay a fixed fee and continue with another observation. Let $y_n = x_n$ if sampling is without recall and $y_n = \max(x_1, \dots, x_n)$ if sampling is with recall.

Let $P(j) =$ unknown probability of pay-off, $j = 1, \dots, k$.

In the modified Bayes rule a prior distribution to the set of

probabilities is specified and after an observation is made, the prior distribution is modified to reflect the information this observation has given. After going through the Bayesian analysis, they showed that the expected gross improvement in the sequence pay-off for the (n+1)st sequence is equal to the stopping rule function $T_{n+1}(y_n)$ which is given by

$$T_{n+1}(y_n) = \sum_{j=y_{n+1}}^k (j - y_n) P_n(j)$$

$$= \frac{1}{m+n} \sum_{j=y_{n+1}}^k m_j (j - y_n),$$

where m_j refers to the confidence coefficient corresponding to the personal probability $P(j)$.

Comparing the value of the above stopping rule function with the value of c will determine the stopping point; that is, if $T_{n+1}(y_n) \leq c$, the sampling should be stopped. Since y_n is a monotonically non-decreasing function of n , $T_{n+1}(y_n)$ is a decreasing function of n , which approaches zero as n increases. Thus, sampling will always stop eventually.

The above stopping rule is derived independent of the prior distribution of the sequence pay-offs. But apart from the mathematical complexity involved in converting the prior to the posterior distribution after each sample and the difficulty in calculating true

values of the pay-offs and cost, purely subjective confidence coefficients m_j 's used in the stopping rule function must have a biasing effect in the decision. Different experimenters having different degrees of confidence will come up with different "optimal" sample sizes. For example, suppose two experimenters have exactly the same personal probabilities, but they have different degrees of confidence m in the personal probabilities. Also, suppose that the experimenters have made n observations with the same y_n for each experiment. Then since $T_{n+1}(y_n)(m)$ is a strictly increasing function of m (Randolph, 1968), there exist values of c such that for a given y_n the less confident experimenter will stop, but the more confident experimenter will continue sampling.

Samual H. Brooks (1958) discussed different random methods for seeking maxima. There he mentioned how to determine the number of trials required in random sampling. Quite analogously, Giffler, Thompson and Van Ness (1963) calculated the number of trials needed in random generation to come to a decision. They consider Monte Carlo process as a binomial trials process in which either an "optimal" schedule is obtained or not. Let

P = probability of favorable event

$1 - P$ = probability of unfavorable event

$1 - (1 - P)^n$ = probability of getting an "optimal" schedule in n trials.

If P can be considered as some specified lowest percent of schedule time, then the stopping rule can be identified as determining the number of samples corresponding to any confidence coefficient. But

as McRoberts (1971) pointed out that in the explosive type combinatorial problems, even a small value of P may require a sample size which is computationally uneconomical.

McRoberts (1971) outlines and provides examples of the application of the extreme value distribution as mechanisms for estimating optimum limits. This will be discussed in more detail in Chapter 6. From the distribution of the extreme values obtained from randomly generated independent samples, a minimum bound can be estimated. By applying procedures outlined, two important variables of interest can be studied: first, for a particular sample the deviation from the estimated minimum can be found, and second, the corresponding probability of improvement can be measured. The useful decision model described on the basis of the estimated minimum is as follows:

$$\text{if } \int_{Z \text{ min}}^L (L - Z) dF(Z) < C_T/C_0,$$

then accept the best criterion measure so far obtained and stop. Otherwise, continue the search where C_T = cost per trial, C_0 = opportunity cost of improving the criterion measure and L = estimated value of the lower bound Z .

The distribution of the extreme values has the advantage of being independent of the parent distribution and the parameters, as we shall see, can be easily obtained and interpreted. If the experimenter is interested in a schedule time within some specified percentage of the estimated lowest value, this can be used as a stopping rule for Monte Carlo sampling.

B. Distribution-free Stopping Rules

In this section, we explain the different stopping rules that we will use in the dissertation to halt the Monte Carlo sampling. All of these are independent of the parent distribution of the schedule time and can be easily incorporated into the scheduling algorithm. The algorithm will terminate the sampling procedure when the stopping rule function converges to a specified value. The fact that the schedule time distribution is truncated at both ends will be useful in the discussion of stopping rules.

1. Stopping rule 1

Let i th sample be drawn from the set of the feasible schedules and the maximum (MAX_i) and the minimum (MIN_i) schedule times are determined. As shown in figure 3.1, A and B refer to the two extreme points of the distribution.

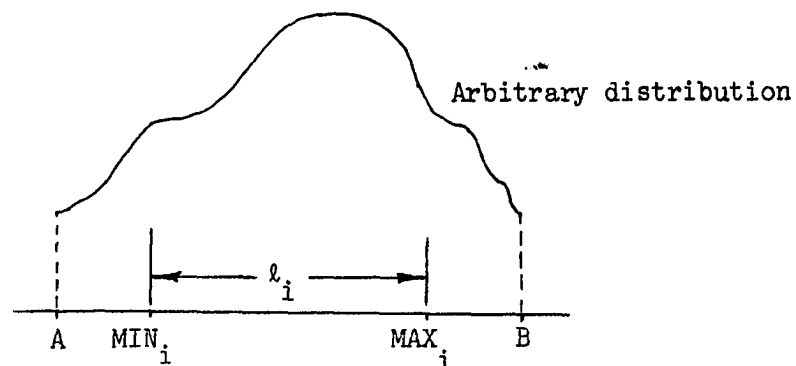


Figure 3.1. Variables in stopping rule 1

$$\text{Let } \rho_i = \text{MAX}_i - \text{MIN}_i$$

Next, the $(i+1)$ st sample is drawn and as before (MAX_{i+1}) and (MIN_{i+1}) are determined.

Take

$$\text{MAX}_{i+1} = \max(\text{MAX}_{i+1}, \text{MAX}_i)$$

and

$$\text{MIN}_{i+1} = \min(\text{MIN}_{i+1}, \text{MIN}_i)$$

As before,
$$\rho_{i+1} = \text{MAX}_{i+1} - \text{MIN}_{i+1}$$

Now define the stopping rule function (ρ) by

$$\rho = \rho_{i+1} - \rho_i$$

If $\rho < \epsilon_1$ (a specified value), then sampling is stopped. Otherwise, we make $\rho_i = \rho_{i+1}$ and continue sampling. Since A and B are two fixed points, stopping rule function (ρ) will converge eventually.

2. Stopping rule 2

Corresponding to the i th sample, let us determine three variables, MAX_i , MIN_i and mean schedule time (\bar{X}_i) . Then the $(i+1)$ st sample is drawn and the corresponding three variables are determined. In figure 3.2, let A and B again refer to the extreme points of the distribution. After each sample, the mean is always calculated on the entire number of schedules drawn so far. For example, let the sample size be n and the total number of schedules drawn at

the end of the i th sample be N_i . In this case \bar{X}_i and \bar{X}_{i+1} are calculated as follows:

$$\bar{X}_i = \frac{\sum_{K=1}^{N_i} T_K}{N_i}, \text{ where } T_K \text{ is the time for } K\text{th schedule}$$

and

$$\bar{X}_{i+1} = \frac{\sum_{K=1}^{N_{i+n}} T_K}{N_{i+n}}$$

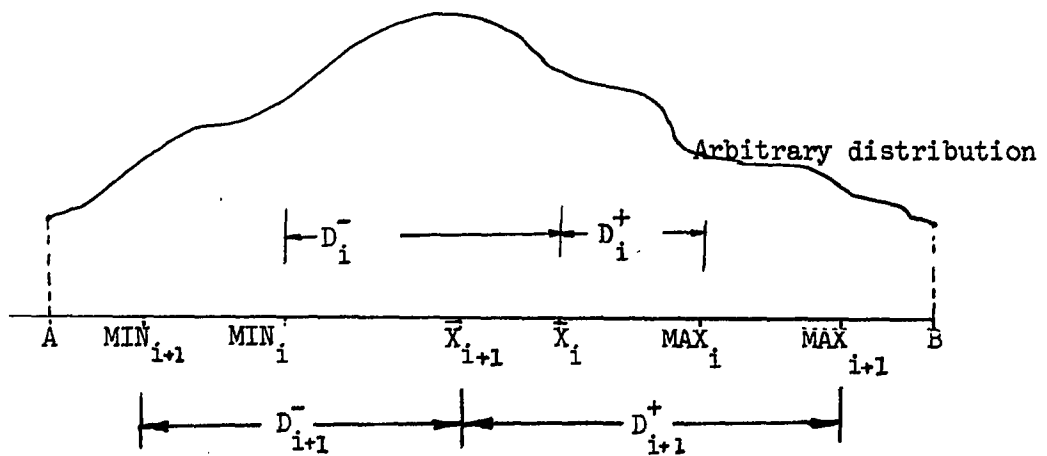


Figure 3.2. Variables in stopping rule 2

Let us now take

$$\text{MAX}_{i+1} = \max(\text{MAX}_{i+1}, \text{MAX}_i)$$

$$\text{MIN}_{i+1} = \min(\text{MIN}_{i+1}, \text{MIN}_i)$$

$$D_i^- = \bar{X}_i - \text{MIN}_i$$

$$D_{i+1}^- = \bar{X}_{i+1} - \text{MIN}_{i+1}$$

$$D_i^+ = \text{MAX}_i - \bar{X}_i$$

$$D_{i+1}^+ = \text{MAX}_{i+1} - \bar{X}_{i+1}$$

Let us now define the stopping rule functions (D^-) and (D^+) as follows:

$$D^- = |D_i^- - D_{i+1}^-|$$

and

$$D^+ = |D_i^+ - D_{i+1}^+|$$

If $D^- \leq \epsilon_2$ (a specified value)

and

$$D^+ \leq \epsilon_2,$$

sampling is stopped. Otherwise, we continue sampling. Since \bar{X}_i 's are approximately normally distributed, the value of \bar{X}_i will be stabilized as more and more samples are drawn. As A and B are fixed, MAX_i and MIN_i will also be stabilized with more samples. Therefore, stopping rule functions will converge eventually.

3. Stopping rule 3

This rule is the same as stopping rule 2 except that it does not take into account the maximum value of the schedule in the sample.

Only the mean value and the left tail of the distribution are the considered. This rule can be estimated from rule 2 as follows:

If $D^- \leq \epsilon_3$ (a specified value), sampling is stopped. Otherwise, we continue sampling.

4. Stopping rule 4

Assume the i th and $(i+1)$ st samples are drawn and as before MIN_i and MIN_{i+1} are determined. In this case, the stopping rule function will be defined by

$$M = MIN_i - MIN_{i+1}$$

If $M = 0$, sampling is stopped; otherwise, set

$$MIN_i = \min (MIN_i, MIN_{i+1})$$

and continue sampling. Since the lowest value of the distribution is fixed, function M is converging.

5. Stopping rule 5

Let $P(i,j)$ be the processing time of job i on machine j . Then the lower limit of the schedule time is defined by

$$LL = \max \left\{ \sum_i P(i,j), \sum_j P(i,j) \right\} \text{ for all } i,j.$$

This lower limit is not usually obtainable. In Chapter 6, we shall estimate the minimum schedule time. Let the estimated minimum be e .

Let Ba be any desired lower limit so that it is within some specified percentage of LL or e . Referring to the stopping rule 4, we determine both maximum and minimum values. We define the function

M and N by

$$M = \text{MIN}_i - \text{MIN}_{i+1}$$

and

$$N = \text{MAX}_{i+1} - \text{MAX}_i.$$

Now the stopping rule is defined as follows:

$$\text{If } \text{MIN}_i \leq B_d$$

or

$$M = N = 0,$$

sampling is stopped. Otherwise, we make

$$\text{MIN}_i = \min(\text{MIN}_i, \text{MIN}_{i+1})$$

and

$$\text{MAX}_i = \max(\text{MAX}_i, \text{MAX}_{i+1})$$

and continue sampling.

6. Stopping rule 6

As explained in stopping rule 5, the lower limit LL is not normally obtainable. In a system having facilities with non-identical multiple machines, it will be even more difficult to obtain the lower limit. Therefore, the scheduler may not be interested in specifying the stopping criterion on the basis of this lower limit. In this case, if estimated minimum e is not available, the stopping rule 5 completely ignores B_d and sampling depends only on the converging rate of M and N .

IV. SINGLE MACHINE ANALYSIS

A. Introduction

This chapter will deal with a special case of the multiple machine algorithm which was presented in Chapter 2. Instead of multiple machines in each work center, a single machine will be considered here.

As indicated in Chapter 2, the algorithm can generate two sets of feasible solutions, one by Monte Carlo sampling and the other by incorporating in it the principle of left shifting. The difference between these two feasible sets of solutions will be examined in detail with respect to the different factors such as minimum schedule time, sample size needed, CPU time, and distribution of the schedule time, etc.

The effects of biasing techniques on the above solutions will also be explored.

The different distribution-free stopping rules discussed in Chapter 3 will be studied with respect to the above factors. A decision rule will be suggested regarding the use of those stopping rules.

Before we enter into the different phases of analysis, let us make some short remarks on the different parameters and variables that will frequently be used in our discussion.

The most important measure of the worth of the Monte Carlo process in our discussion will be the length of the shortest schedule observed.

In order to make a comparative analysis between the two sets of feasible solutions obtained by the Monte Carlo process, a worthwhile parameter to consider is the most probable schedule observed. This provides some indication as to how close the shortest schedule

observed is to the schedule having the highest probability.

In conjunction with the shortest and the most probable schedules, another important parameter of interest will be the range which is defined as the difference between the longest and the shortest schedules observed. Figure 4.1 shows the above parameters.

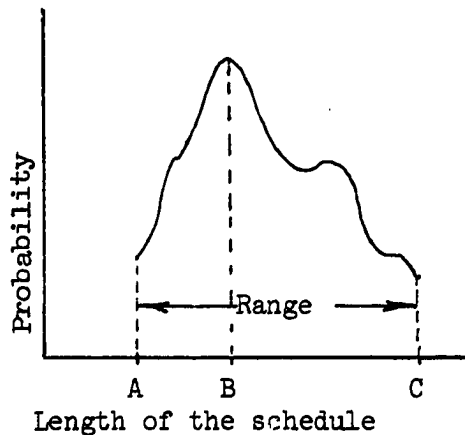


Figure 4.1. Different Parameters

In Figure 4.1, A = shortest schedule

B = most probable schedule

C = longest schedule

Usually a scheduler is interested in obtaining a "good" solution at the cost of a reasonable amount of resources. Therefore, in Monte Carlo sampling, the number of schedules needed to arrive at a "good" solution and the corresponding computer time should also be considered as measures of the quality of the solution.

The following short notations will be used in our subsequent discussions:

COMB: combination technique

MIN: time of the shortest schedule

MAX: time of the longest schedule

MPS: time of the most probable schedule

- R: range of the distribution
- N: number of schedules
- T: CPU time per schedule (in seconds)
- NS: non-left-shifting
- LS: left-shifting
- BT: biasing technique (minimax technique)
- LL: lower limit of the schedule time (discussed in section B, Chapter 3, to explain stopping rule 5)
- UL: upper limit of the schedule time. This is equal to the sum of all processing times
- SR: stopping rule

B. Left-Shifting and Non-Left-Shifting

In order to make the shop situation more general and realistic, the algorithm also considered the passing and backtracking of jobs on some of the machines. This is why the total number of operations is not necessarily the product of the numbers of jobs and machines as shown in the sample problems of Table 4.1.

Processing times are uniformly distributed over the interval [1,9] for the first four problems in Table 4.1, and for the remaining problems the interval was changed to [10,99]. The technological ordering was chosen with the help of a random table. Solution parameters and variables for the sample problems have been entered in Tables 4.2 through 4.8.

Tables 4.2 through 4.8 indicate the improvement in the solution by left-shifting. In all the sample problems, the span of the minimum schedule obtained by this technique is considerably lower than that by

Table 4.1. Sample problems

Sample Problem	Total Number of Operations
5 x 4	16
9 x 4	41
9 x 6	60
15 x 4	68
6 x 7	46
6 x 10	60
6 x 15	99

Table 4.2. Sample problem 5 x 4 (LL = 27; UL = 79)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$l_{i+1} - l_i < 1$	32	30	61	45	29	15	45	36	150	100	.026	.0334
$ D_{i+1}^- - D_i^- \leq .5$ and $ D_{i+1}^+ - D_i^+ \leq .5$	32	30	61	45	29	15	45	36	150	150	.027	.0361
$ D_{i+1}^- - D_i^- \leq .5$	32	30	61	45	29	15	45	36	100	150	.027	.0334
$MIN_i - MIN_{i+1} = 0$	32	30	61	45	29	15	45	36	150	100	.027	.0334
$MIN_i \leq Bd = LL$ or $MIN_i - MIN_{i+1} = 0$ and $MAX_{i+1} - MAX_i = 0$	32	30	61	45	29	15	45	36	150	100	.027	.0348

Table 4.3. Sample problem 9 x 4 (LL = 56; UL = 246)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$\ell_{i+1} - \ell_i < 1$	68	62	122	99	54	37	110	71	80	80	.143	.143
$ D_{i+1}^- - D_i^+ \leq .5$ and $ D_{i+1}^- - D_i^+ \leq .5$	66	58	124	107	58	49	107	77	120	120	.161	.170
$ D_{i+1}^- - D_i^- \leq .5$	66	58	124	107	58	49	107	77	120	120	.144	.146
$MIN_i - MIN_{i+1} = 0$	68	62	122	99	54	37	110	71	80	80	.133	.136
$MIN_i \leq Bd = 1.2 LL$ or $MIN_i - MIN_{i+1} = 0$ and $MAX_{i+1} - MAX_i = 0$	66	62	124	99	54	37	107	71	120	40	.143	.145

Table 4.4. Sample problem 9 x 6 (LL = 101; UL = 317)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$l_i - l_{i+1} < 5$	137	123	222	175	85	52	181	141	60	60	.29	.32
$ D_i^- - D_{i+1}^- \leq .5$ and	126	114	245	183	119	69	190	135	150	150	.31	.36
$ D_i^+ - D_{i+1}^+ \leq .5$												
$ D_i^- - D_{i+1}^- \leq .5$	132	116	233	183	101	65	197	135	120	120	.29	.33
$MIN_i - MIN_{i+1} = 0$	132	116	233	183	101	65	197	135	120	120	.29	.32
$MIN_i \leq Bd = LL$ or												
$MIN_i - MIN_{i+1} = 0$ and	132	114	233	183	101	65	197	135	120	150	.33	.35
$MAX_{i+1} - MAX_i = 0$												

Table 4.5. Sample problem 15 x 4. (LL = 126; UL = 351)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$\ell_i - \ell_{i+1} < 1$	130	127	230	158	100	31	179	141	40	40	.49	.52
$ D_i^- - D_{i+1}^- \leq .5$							169	133				
and								131				
$ D_i^+ - D_{i+1}^+ \leq .5$	130	127	241	170	111	43	179	133	60	60	.52	.55
$ D_i^- - D_{i+1}^- \leq .5$	130	127	241	170	111	43	179	138	60	60	.52	.55
$MIN_i - MIN_{i+1} = 0$	130	127	230	158	100	31	179	141	40	40	.49	.52
							169	133				
								131				
$MIN_i \leq Bd = 1.01 LL$												
or												
$MIN_i - MIN_{i+1} = 0$	130	127	230	158	100	31	179	133	40	20	.51	.55
and							169					
$MAX_{i+1} - MAX_i = 0$												

Table 4.6. Sample problem 6 x 7 (LL = 379; UL = 2123)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$l_i - l_{i+1} < 10$	561	497	1072	739	511	242	779	515	100	100	.22	.25
$ D_i^- - D_{i+1}^- \leq .5$ and $ D_i^+ - D_{i+1}^+ \leq .5$	537	491	1103	739	556	248	763	515	150	200	.22	.25
$ D_i^- - D_{i+1}^- \leq .5$	561	497	1072	739	511	242	779	685 515	150	150	.22	.25
$MIN_i - MIN_{i+1} = 0$	561	497	1072	739	511	242	779	685 515	100	150	.22	.25
$MIN_i \leq Bd = LL$ or $MIN_i - MIN_{i+1} = 0$	561	497	1072	739	511	243	779	685 515	100	150	.22	.25
$MAX_{i+1} - MAX_i = 0$												

Table 4.7. Sample problem 6 x 10 (LL = 507; UL = 3162)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$l_{i+1} - l_i < 35$	873	683	1502	1022	629	339	1272	729	100	100	.42	.47
< 20		663								150		
$ D_{i+1}^- - D_i^- \leq .5$ and $ D_{i+1}^+ - D_i^+ \leq .5$	845	656	1502	1022	707	367	1242	719	200	250	.51	.52
$ D_{i+1}^- - D_i^- \leq .5$	873	663	1502	1022	629	359	1272	769 719 714	150	150	.45	.48
$MIN_i - MIN_{i+1} = 0$	873	663	1502	1022	629	359	1272	791 769 719 714 705	150	200	.42	.47
$MIN_i \leq Bd = LL$ or $MIN_i - MIN_{i+1} = 0$ and $MAX_{i+1} - MAX_i = 0$	873	663	1502	1022	629	359	272	791 769 719 714 705	150	200	.50	.51

Table 4.8. Sample problem 6 x 15 (LL = 750; UL = 4738)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$l_{i+1} - l_i < 30$	1152	847	1859	1232	707	385	1412	949	100	100	.89	.97
< 5		847								150		
$ D_{i+1}^- - D_i^- \leq .5$ and	1095	843	1859	1236	764	393	1404	917	200	200	.93	1.02
$ D_{i+1}^+ - D_i^+ \leq .5$												
$ D_{i+1}^- - D_i^- \leq .5$	1131	847	1859	1236	728	389	1434	1008 984	150	150	.94	.99
$MIN_i - MIN_{i+1} = 0$	1131	847	1859	1236	728	389	1434	1008 984	150	150	.89	.97
$MIN_i \leq Bd = LL$ or												
$MIN_i - MIN_{i+1} = 0$ and	1131	843	1859	1236	728	393	1434	917	150	250	.94	.99
$MAX_{i+1} - MAX_i = 0$												

non-shifting. The range of the distribution of the schedule time is always much smaller in left-shifting.

From figures 4.2 through 4.10, we find that in some cases, the most probable schedule (MPS) provides a rough indication about the nature of the distribution. But, in most of the cases, it does not provide any good indication. In this respect, the number of "peaks" in the distribution might be an important parameter of interest to us. A peak is defined to be a point at which the observed frequency stops increasing and starts decreasing. If the number of peaks is large, it becomes difficult to approximate the nature of the distribution by a single parameter like MPS. Schedules obtained by an improved technique such as LS will have relatively fewer number of peaks in the distribution, though no generalization to this claim can be made. However, in Monte Carlo sampling related to our scheduling problems, instead of specifying the most probable schedule (MPS) only, knowledge of the schedules (mostly more than one) having relatively higher frequency with their corresponding probabilities will perhaps be more helpful to predict the approximate nature of the schedule time distribution.

C. Biasing Techniques

In any random sampling, the use of a biasing technique is usually intended to have an improved subset of the set of all feasible solutions. It is desired that this subset will contain the desired solution with relatively high probability.

BT1: This biasing technique refers to the sampling procedure which, instead of selecting the process at random, selects the one having

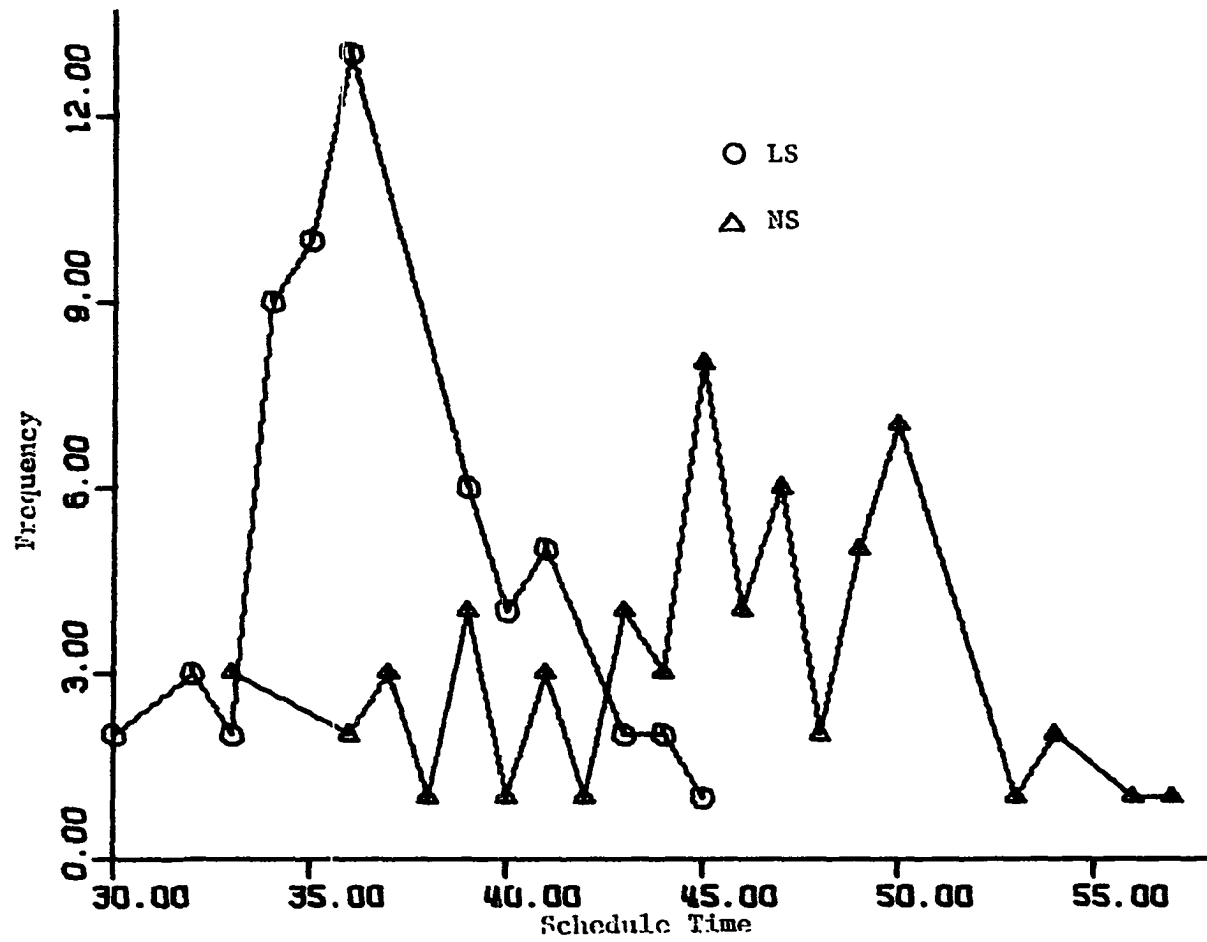


Figure 4.2. Sample problem 5 x 4 (N = 20)

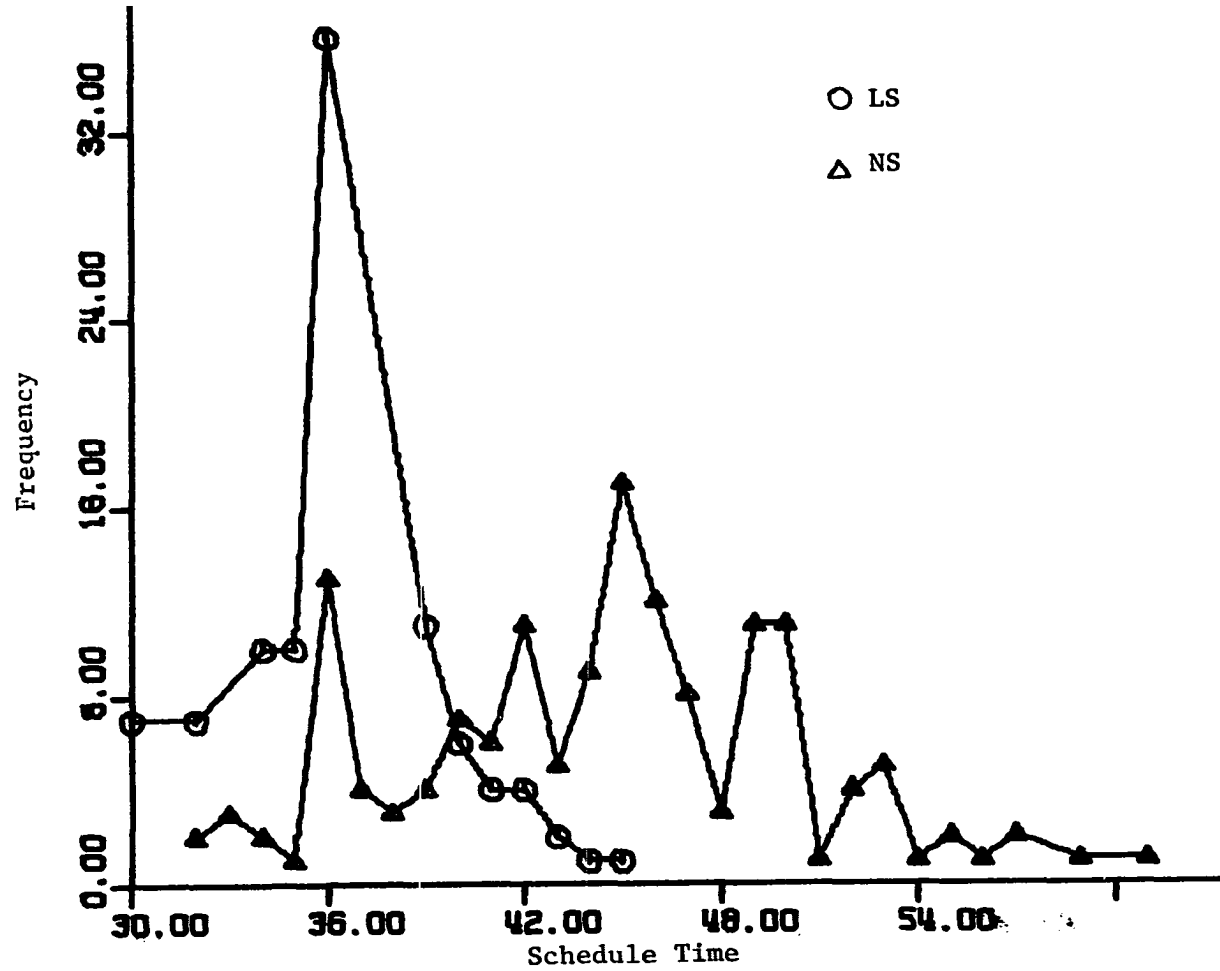


Figure 4.3. Sample problem 5 x 4 (N = 50)

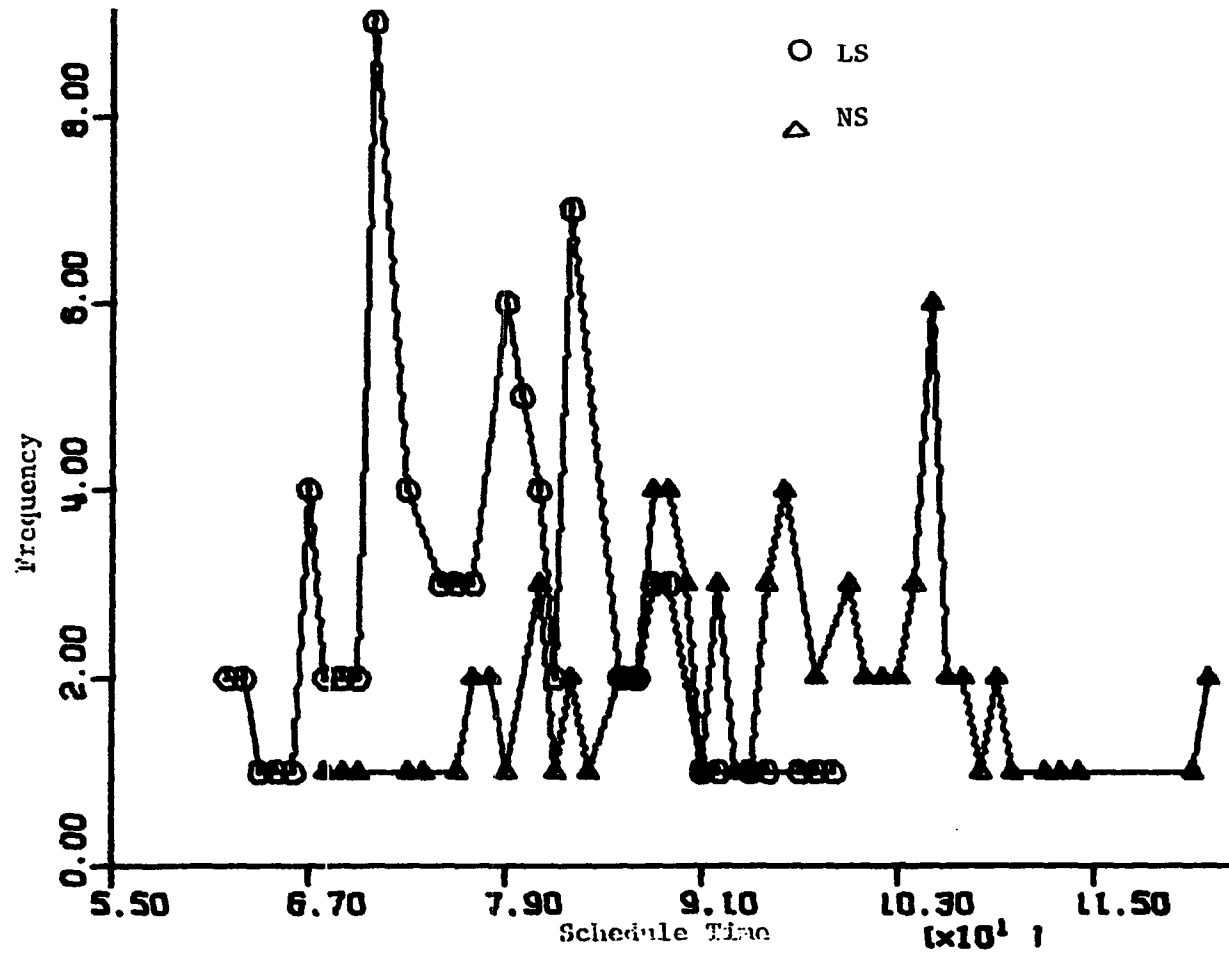


Figure 4.4. Sample problem 9 x 4 (SR 1, 4)

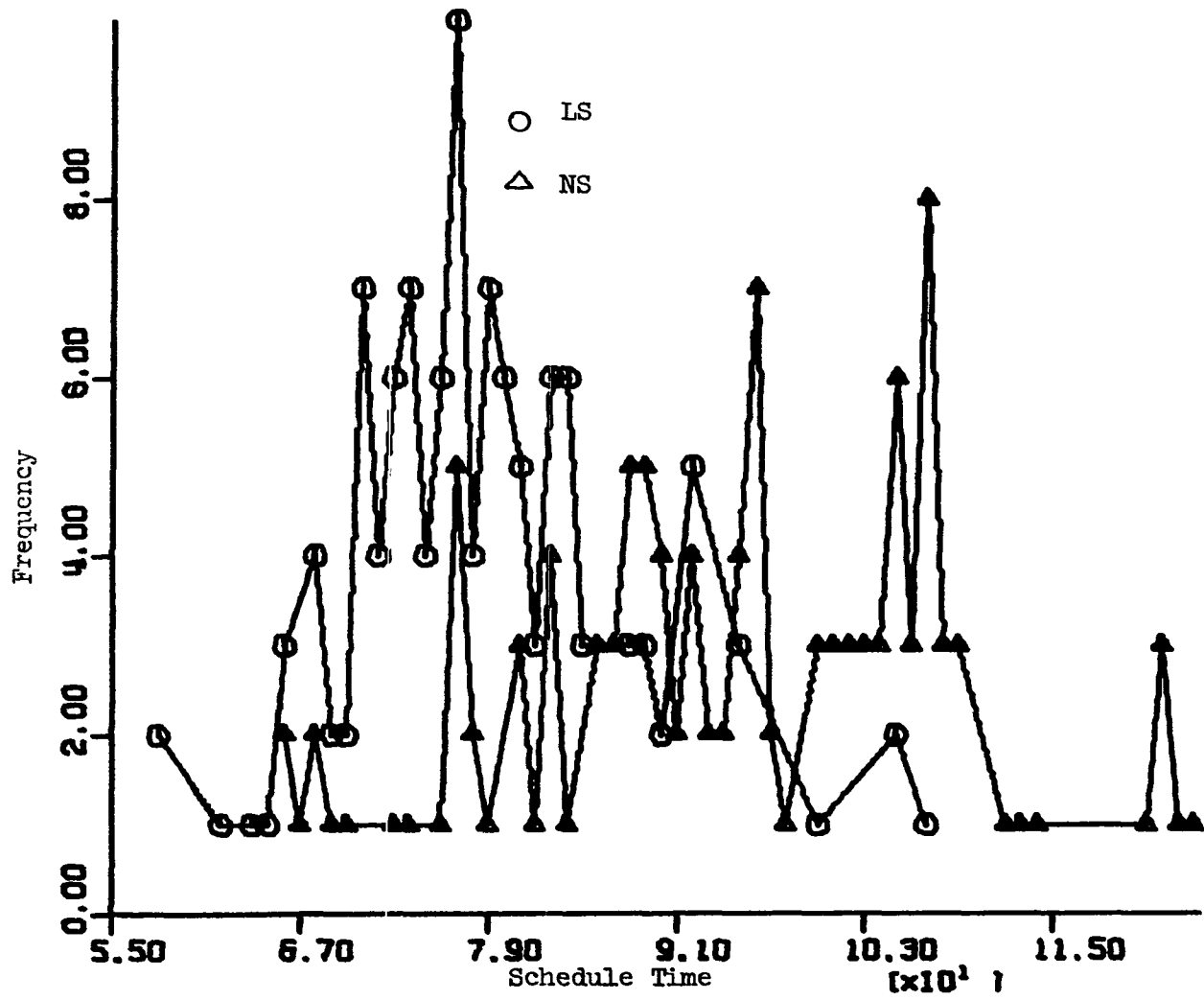


Figure 4.5. Sample problem 9 x 4 (SR 2, 3)

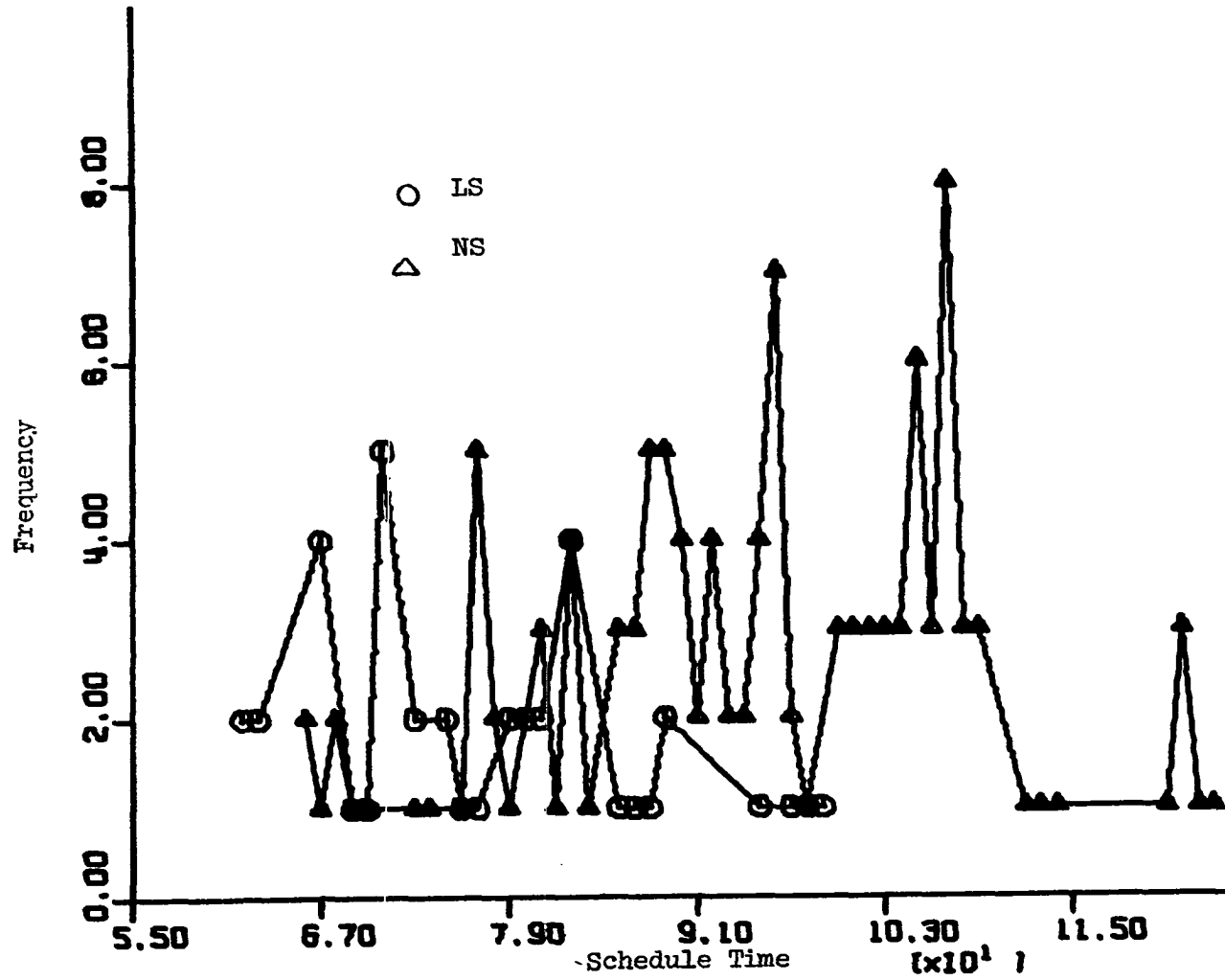


Figure 4.6. Sample problem 9×4 (SR 5)

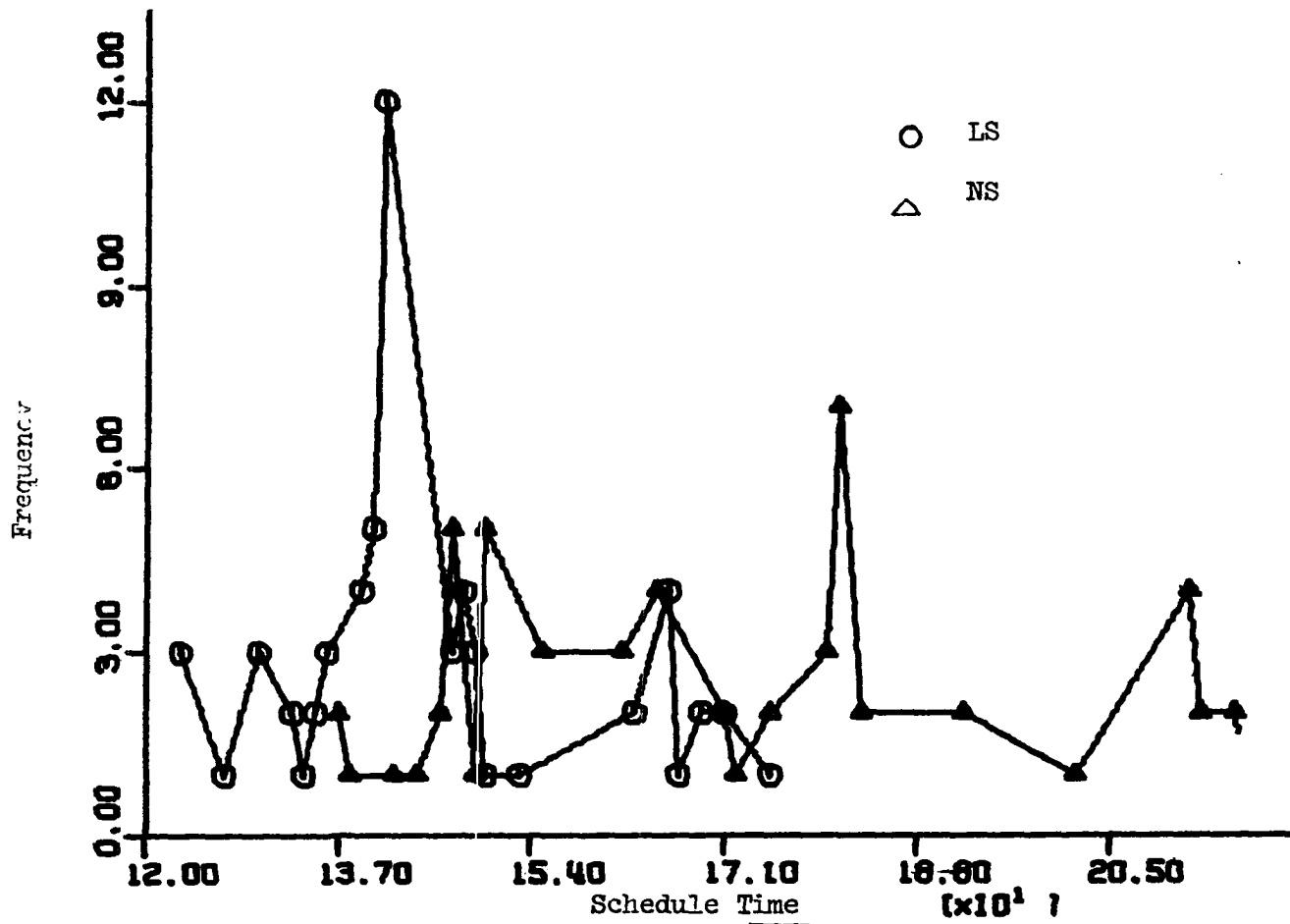


Figure 4.7. Sample problem 9 x 6 (SR 1)

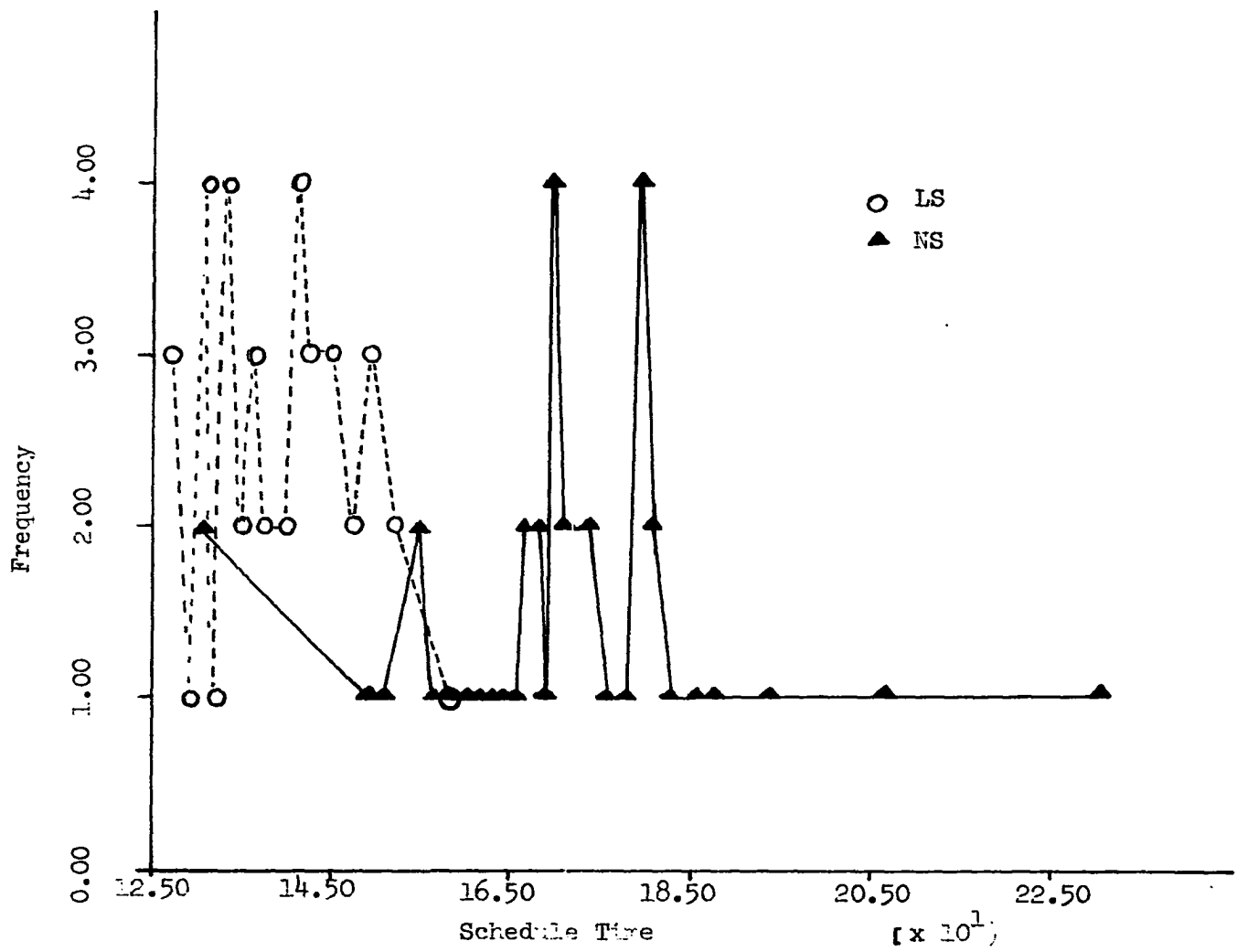


Figure 4.8. Sample problem: 15 x 4 (SR 1,4)

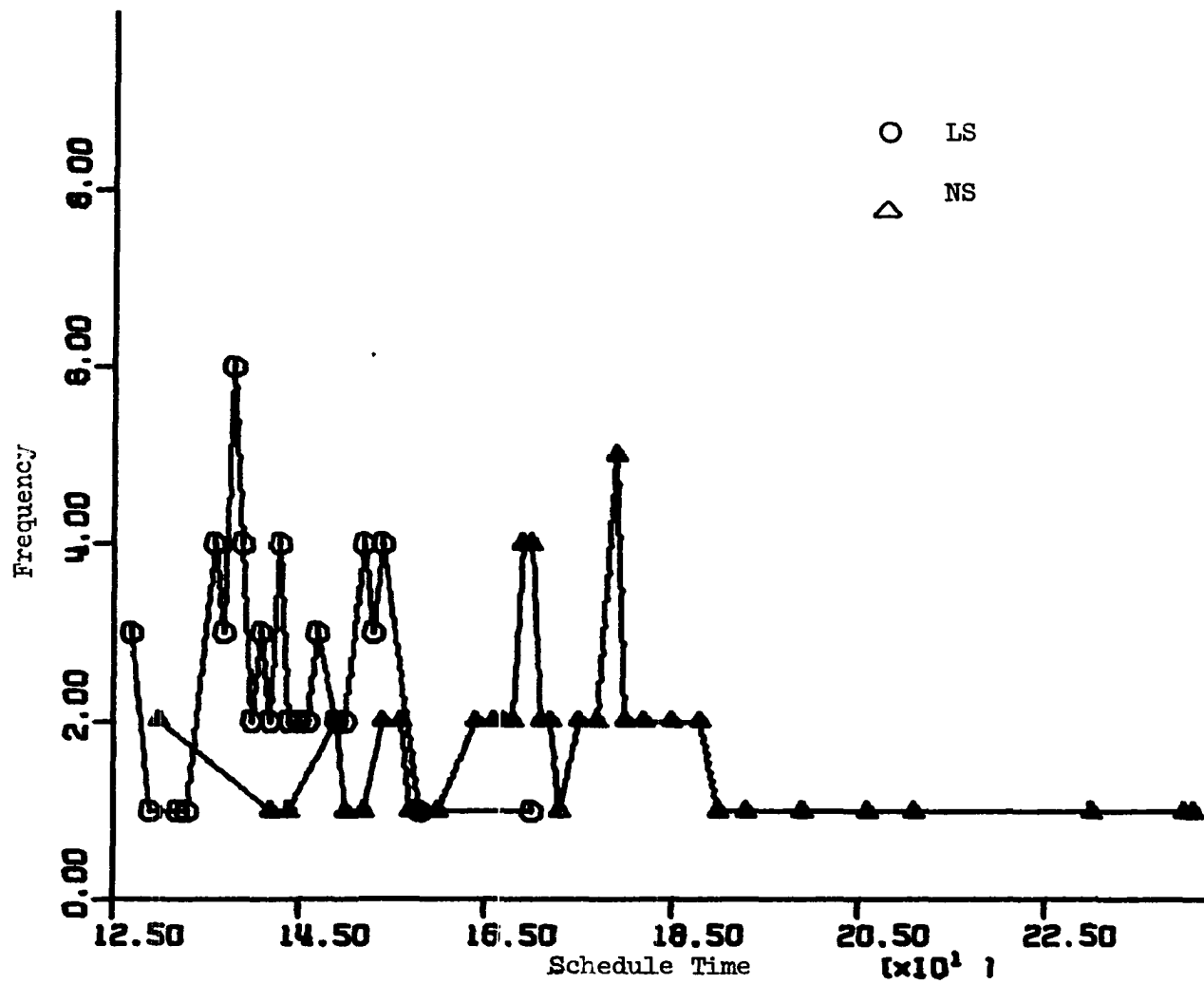


Figure 4.9. Sample problem 15 x 4 (SR 2,-3)

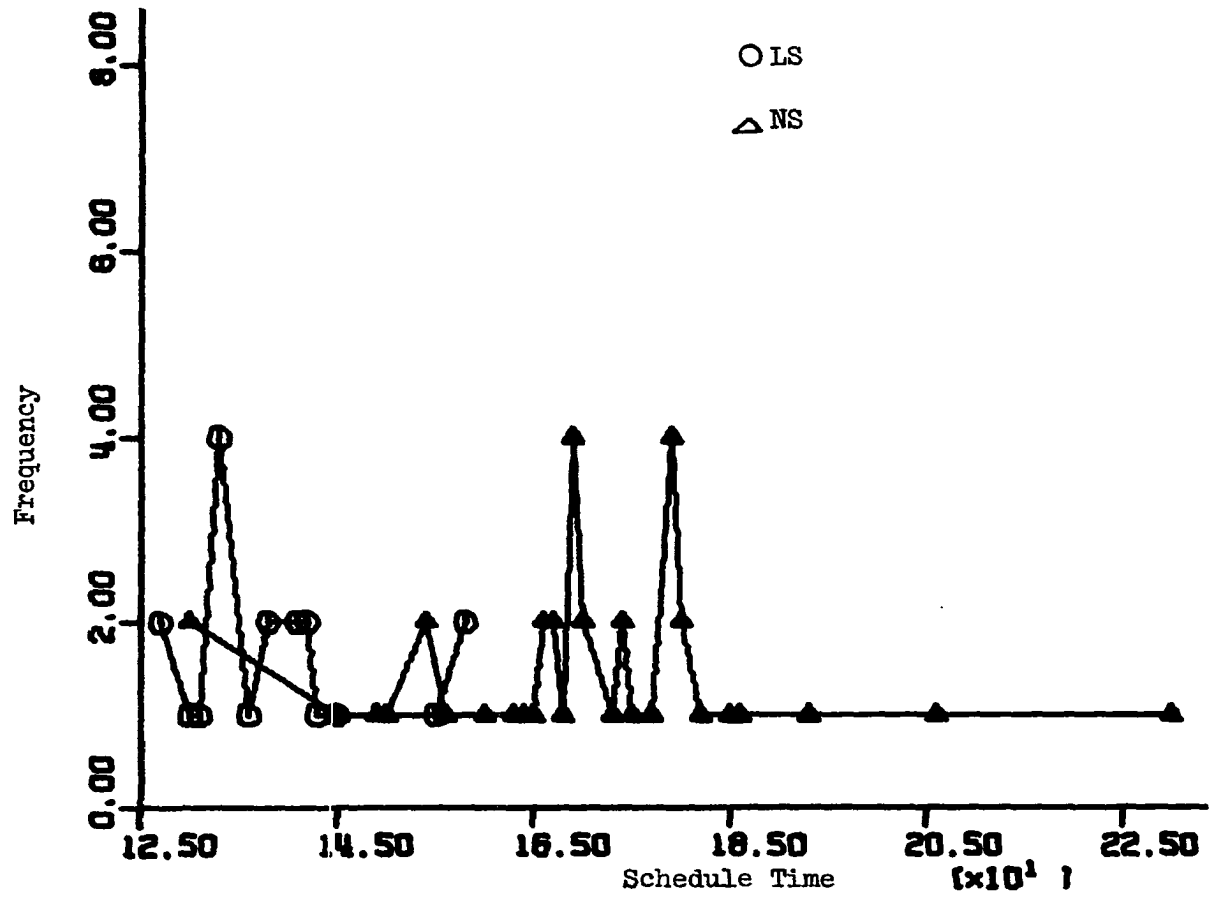


Figure 4.10. Sample problem 15 x 4 (SR 5)

minimum starting time. Referring to the algorithm developed in section B, let P be the set of all scheduleable operations. Corresponding to each operation in P , find $\max_i = \max$ (maximum machine time and maximum job time), $i = 1, 2, \dots, P$. Now find a subset $Q = \min (\max_1, \max_2, \dots, \max_P)$. Select one of the operations in Q at random.

Due to a shortage of computer funds, this technique was not actually tested. But in view of BT3 discussed in Chapter 5, we can conclude that BT1 should be always better than non-shifting sampling and perhaps superior to left-shifting in respect to the range of the distribution. However, this technique will take much longer CPU time than left-shifting.

BT2: In this technique, after selecting a process by BT1, the left-shifting criterion is incorporated. BT2 can be expressed as $BT2 = BT1 + LS$. BT2 must be at least as good as BT1 in any feasible solution because it is a better subset of BT1.

Six problems were tested and different parameters have been entered in Table 4.9. In all the problems, BT2 has been found to be better than LS with respect to minimum schedule time.

In all the problems, the biasing technique BT2 has the shorter range. It shows that if a single feasible schedule is to be drawn, this technique will provide a better schedule with a higher probability.

But if we look at the CPU time, left-shifting is always better. In most of the cases, the CPU time per schedule in the biasing technique is more than twice as much as that in left-shifting. This is perhaps due to the fact that in the biasing technique, for each operation,

Table 4.9. Comparison between NS, LS and BT2 (Numbers in the table correspond to stopping rule 4)

Sample Problems	MIN			R		N		T	
	NS	LS	BT2	LS	BT2	LS	BT2	LS	BT2
9 x 4	68	62	58	37	23	80	80	0.14	0.31
9 x 6	152	116	112	65	42	120	120	0.32	0.62
15 x 4	130	127	126	31	27	40	40	0.52	1.14
6 x 7	561	497	447	242	186	150	100	0.25	0.58
6 x 10	873	663	636	359	294	200	100	0.47	1.19
6 x 15	1131	847	827	389	321	150	100	0.97	1.61

we are to apply max-min principle over the whole set of schedulable operations to check for the minimum starting time.

Depending on the size and the structure of the problem, left-shifting may sometimes be found superior to BT2 with respect to minimum schedule time. In fact as we shall see in Chapter 5, in a small problem, left-shifting provided a better minimum than BT2. The better results demonstrated by BT2 in most of the cases should probably be apprehended because schedules are drawn from a better set (BT1) than that in left-shifting.

If the CPU time is very critical, left-shifting is always to be applied at the expense of probable better results. If the scheduler is more concerned with the better schedule at the expense of higher CPU time, BT2 is a superior choice.

Perhaps the best choice will be to apply a third alternative which combines both techniques LS and BT2. For each operation, the program will randomly select either LS or BT2 and process the operation by that technique. To make a compromise between CPU time and the better schedule, we suggest this combination technique. It is expected that in all cases this will be better than the worse of the two and close to the better one. With respect to CPU time, this will be better than BT2 and worse than LS. According to the judgement of the scheduler, if he prefers a particular technique, he can assign more weight to it and apply the combination technique.

In fact, our program has been designed to incorporate all three techniques. With the following simple changes on the first data card of

the input stream, any of the three techniques can be applied to a problem.

	Column 19	Column 20
BT:	-	1
LS:		1
Combination:		0

Figure 4.11 shows the relative advantage of BT2 over LS. Figure 4.12 depicts how the combination technique relates with LS and BT2 for the problem 15 x 4. The table 4.10 shows the improvement of CPU time by combination technique in problem 6 x 10.

Table 4.10. Combination technique in 6 x 10 problem

	LS	BT	Combination
Minimum schedule time	663	636	639
CPU time/schedule	0.47	1.19	0.8

D. Sample Size and Stopping Rules

Referring to Tables 4.2 through 4.8, we see that different stopping rules need different numbers of schedules to meet their criterion. Usually stopping rule 1 requires fewer numbers of schedules, especially when a higher $(\ell_{i+1} - \ell_i)$ is specified while stopping rule 2 is relatively slow and the sampling procedure is terminated after drawing more schedules. However, the latter produces the best results.

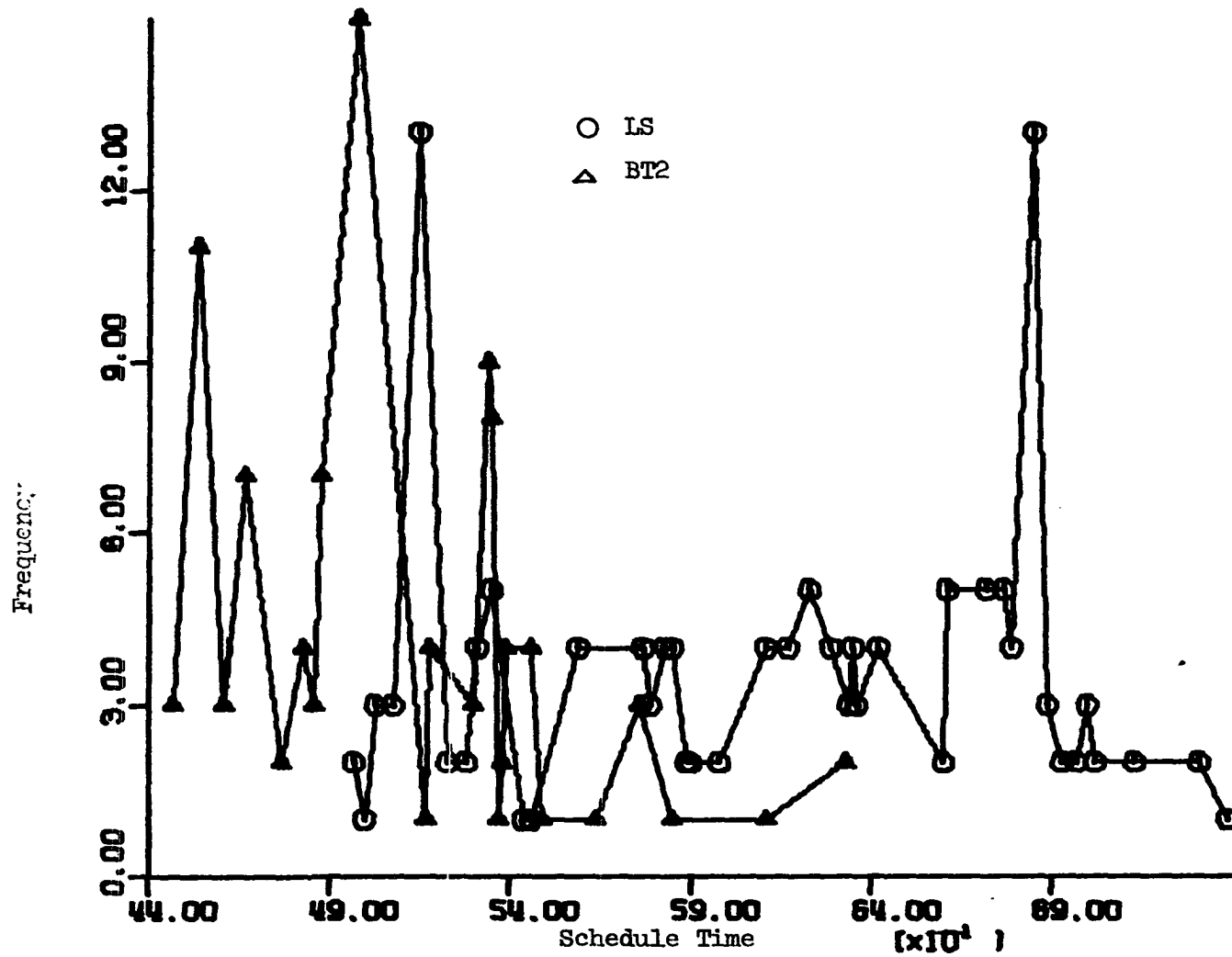


Figure 4.11. Comparison between LS and BT2 (6 x 7)

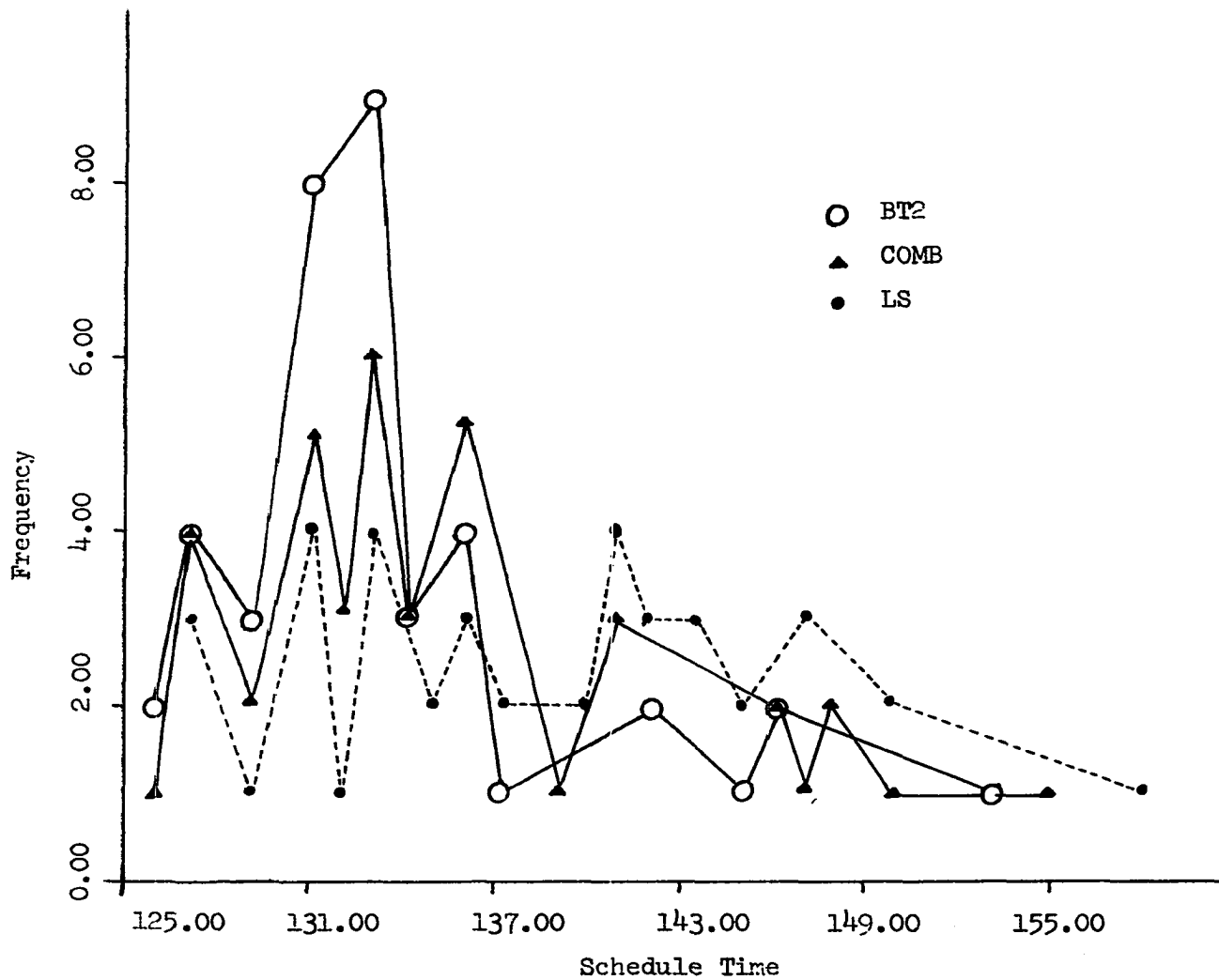


Figure 4.12. Comparison among BT2, COMB, and LS (15 x 4)

If we are interested in a better minimum at the expense of more CPU time, rule 2 should be used.

When CPU time is a critical factor, the scheduler can assign some "reasonable" lower bound for the schedule time and apply stopping rule 5. The lower bound may be assigned on the basis of the lower limit (LL) or any estimated minimum. There is a higher probability that this will give the desired solution within a reasonable amount of time.

For the usual situation, stopping rules 1, 3 and 4 can be applied. Of these three, stopping rule 4 is preferred in view of the CPU time and the better result obtained.

As regards to the sample size, let us observe the relative improvement of the minimum schedule with the increase in the number of schedules (Table 4.11). The numbers in the table are the minimum schedules (MIN) with respect to left-shifting.

From Table 4.11, observe that in most of the cases, 500 schedules gave a better minimum than that obtained from 100 or 150 schedules. However, the improvement may not always justify its worth when CPU time is taken into consideration. All the stopping rules usually need two or more samples to be drawn before the criterion is met. Stopping rule 2 needs more samples as we observed. Stopping rule 5 is sometimes faster if a reasonable desired limit is specified. With this information, we suggest the following rough guide rules for the sample size to be specified in the program (Table 4.12).

Table 4.11. Relative improvement of MIN with number of schedules

Problems	Number of Schedules Drawn				
	50	100	150	200	500
9 x 4	62	60	58	-	58
9 x 6	123	116	114	112	112
15 x 4	127	126	-	-	126
6 x 7	503	497	497	491	486
6 x 10	696	683	663	663	652
6 x 15	876	847	847	843	837

Table 4.12. Stopping rules and sample size (single machine)

Stopping Rules	Approximate Sample Size
1 (with higher $l_{i+1} - l_i$)	40 - 60
1 (with lower $l_{i+1} - l_i$)	30 - 40
2	25 - 35
3	35 - 45
4	35 - 45
5 (with no "bound")	30 - 40
5 (with "bound")	40 - 50

E. CPU Time

Between different stopping rules, CPU time/schedule does not differ significantly. This difference is not always consistent.

As we noticed previously regarding CPU time per schedule, there is not much difference between NS and LS, but there is a significant difference between LS and BT. CPU time per schedule is dependent on the number of operations in the problem. It seems that the algorithm is equally critical on the number of jobs and the number of machines. However, no generalization can be made unless more sample problems are tested. Figures 4.13 and 4.14 show the growth in computing time with the problem size.

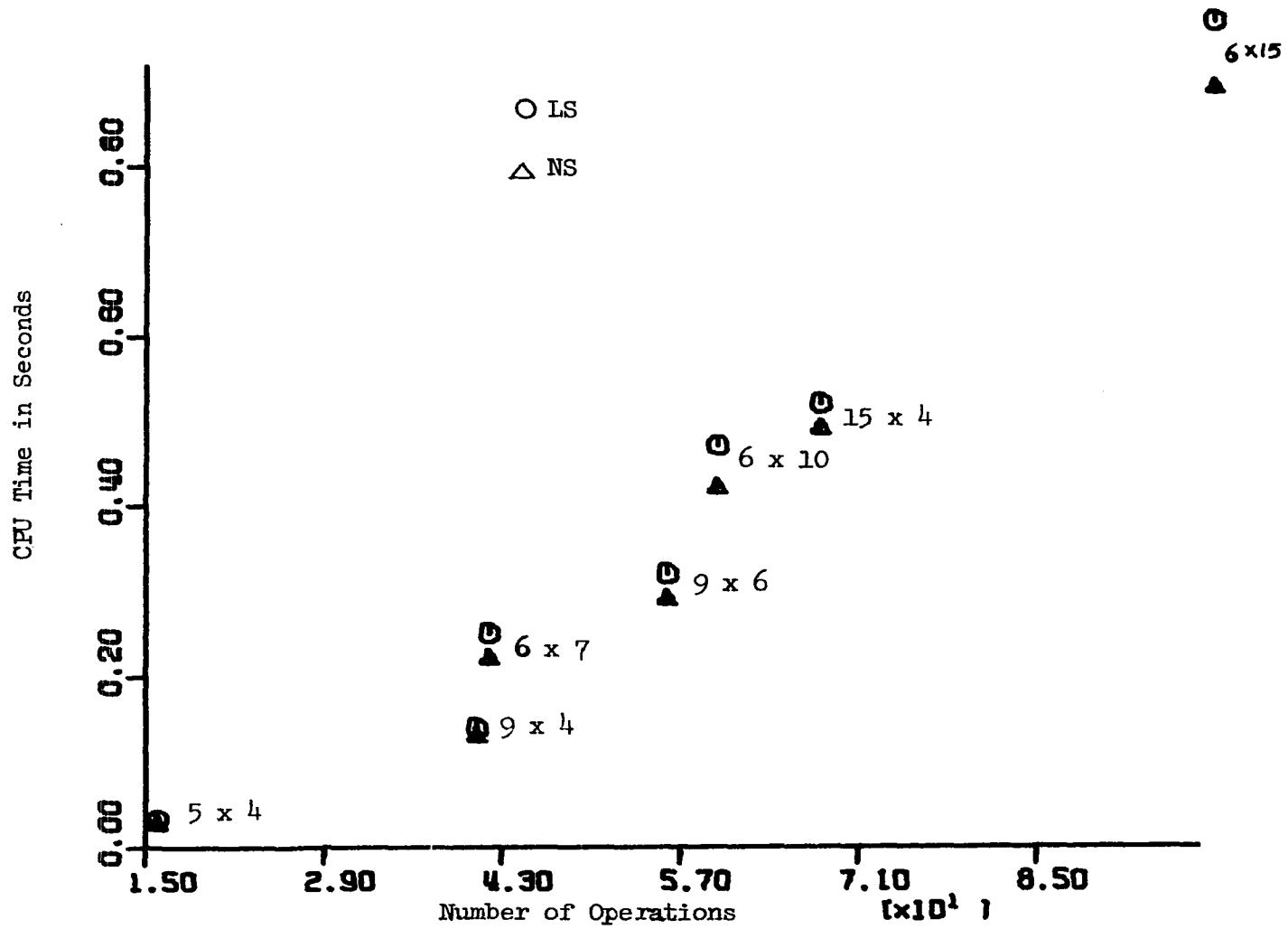


Figure 4.13. CPU time (NS and LS)

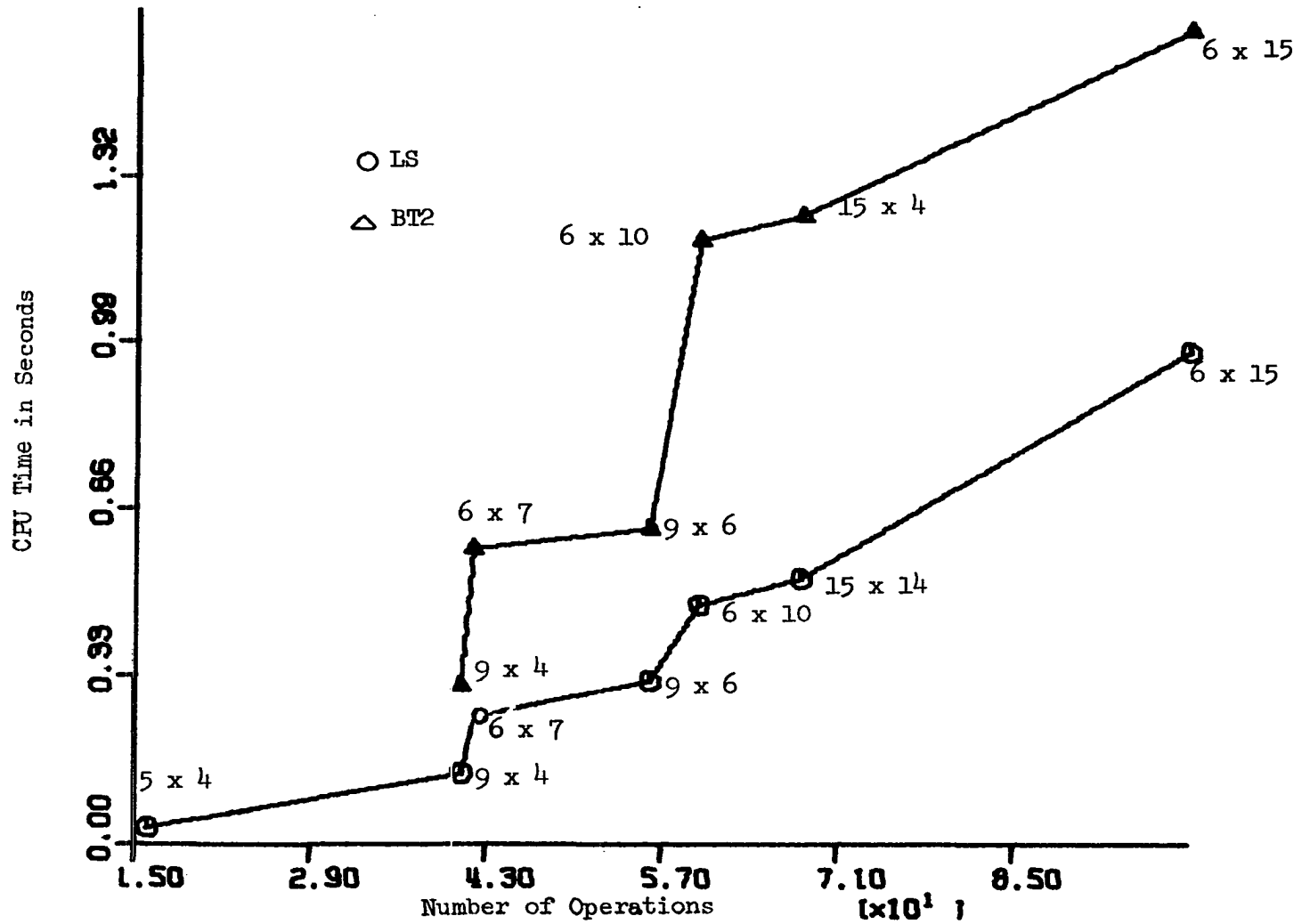


Figure 4.14. CPU Time (LS and BT2)

V. MULTIPLE MACHINE ANALYSIS

A. Introduction

This chapter will deal with the systems where multiple machines can exist in different work centers. This will be the exact representation of the algorithm developed in Chapter 2.

As in Chapter 4, differences between the solutions obtained by left-shifting and non-left-shifting will be studied here with respect to minimum schedule time, sample size needed, CPU time, distribution of schedule time, etc.

A short discussion on single machine versus multiple machines will also be presented.

In addition to the different biasing techniques used in Chapter 4, we will examine the effect of other biasing techniques and explore the possibility of some others.

The stopping rules will be discussed in the context of multiple-machines facilities.

The short notation used in Chapter 4 will also be valid in this chapter.

With respect to the multiple-machines facilities, it is worthwhile to note the following two observations regarding the number of machines and operations. Unlike the single machine facilities, the number of machines and facilities are not the same in "strict" multiple machine facilities. In the latter case, a problem of size 6×7 should not be understood as referring to a system having 6 jobs and 7 machines; it should refer to a system

having 6 jobs and 7 facilities with more than 7 machines. In our discussion, we will specify a multiple-machines facilities system by $n \times f \times m$, where n , f , and m refer, respectively, to the number of jobs, facilities and machines.

Regarding the number of operations in multiple-machines facilities, as mentioned in Chapter 2, some operations will be made "inactive" as we proceed with the algorithm and will never be processed. Therefore, in this case, the total number of possible operations in a problem will be more than the number of "active" operations that are actually processed during the different iterations.

In a later discussion in this chapter on single machine versus multiple machines, we shall see that in a multiple-machines facility, the difference between the minimum obtainable schedule and the lower limit (LL) depends on the number of machines within each facility. For this reason in our subsequent discussion on left-shifting and non-left-shifting, we will not use lower limit (LL) to determine a bound (Bd) to be used in stopping rule function. Thus, stopping rule 6 will be used instead of stopping rule 5 which can be applied in the context of an estimated minimum to be discussed in Chapter 6.

B. Left-Shifting and Non-Left-Shifting

The sample problems in Table 5.1 have been considered to study the different parameters of interest. These problems were also examined in Chapter 4 for single-machine facilities. So the technological ordering remains the same as before for each problem. The solution parameters and variables for the sample problems have been entered in Tables 5.2 through 5.7. The lower limit (LL) that has been mentioned

Table 5.1. Sample problems

Sample Problem Jobs x Facilities x Machines	Total Number of Operations	Number of Active Operations
5 x 4 x 10	39	16
9 x 4 x 10	104	41
15 x 4 x 10	168	68
6 x 7 x 14	104	46
6 x 10 x 19	114	60
6 x 15 x 26	151	99

in each sample problem has been defined as $\max_j (\sum_i P(i,j))$ for all i , where $P(i,j)$ is the processing time of the job i on the most efficient machine of the facility j . Tables 5.2 through 5.7 indicate the improvement in the solution by left-shifting in multiple-machines facilities. As in Chapter 4, this technique is superior to non-shifting procedure with respect to minimum schedule time, range of the schedule time distribution, etc.

From figures 5.1 through 5.3, we observe that the multiple-machines facilities also a single parameter MPS (most probable schedule) does not provide a very good indication about the nature of the distribution. In this respect, we refer to our discussion in Chapter 4.

Table 5.2. Sample problem 5 x 4 x 10 (LL = 24)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$(\ell_{i+1} - \ell_i) < 1$	35	28	73	50	38	32	37		80	80	.105	.107
$ D_i^- - D_{i+1}^- \leq .5$ and $ D_i^+ - D_{i+1}^+ \leq .5$	33	28	75	50	42	32	38		120	120	.105	.108
$ D_i^- - D_{i+1}^- \leq .5$	33	28	75	50	42	32	38		120	120	.105	.107
$(\text{MIN}_i - \text{MIN}_{i+1}) = 0$	35	28	73	50	38	32	37		80	80	.105	.107
$(\text{MIN}_i - \text{MIN}_{i+1}) = 0$ and $(\text{MAX}_{i+1} - \text{MAX}_i) = 0$	35	28	73	50	38	32	37		80	80	.105	.108

Table 5.3. Sample problem 9 x 4 x 10 (LL = 36)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$(\ell_{i+1} - \ell_i) < 1$	55	44	103	82	48	38	77	62	100	100	.59	.52
$ D_i^- - D_{i+1}^- \leq .5$ and $ D_i^+ - D_{i+1}^+ \leq .5$	51	44	112	83	61	39	76	64	150	150	.50	.54
$ D_i^- - D_{i+1}^- \leq .5$	55	44	103	82	48	38	77	62	100	100	.49	.52
$(\text{MIN}_i - \text{MIN}_{i+1}) = 0$	55	44	103	82	48	38	77	62	100	100	.49	.52
$(\text{MIN}_i - \text{MIN}_{i+1}) = 0$ and $(\text{MAX}_{i+1} - \text{MAX}_i) = 0$	55	44	103	82	48	38	77	62	100	100	.50	.53

Table 5.4. Sample problem 15 x 4 x 10 (LL = 38)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$(\ell_{i+1} - \ell_i) < 1$	93	76	181	138	88	62	113	93	100	80	1.15	1.4
$ D_i^- - D_{i+1}^- \leq .5$ and	93	76	185	143	92	67	131	94	150	100	1.16	1.46
$ D_i^+ - D_{i+1}^+ \leq .5$	93	76	181	138	88	62	113	93	100	80	1.14	1.43
$(MIN_i - MIN_{i+1}) = 0$	93	76	181	138	88	62	113	93	100	80	1.14	1.4
$(MIN_i - MIN_{i+1}) = 0$ and	93	76	181	143	88	62	113	94	100	120	1.15	1.43
$(MAX_{i+1} - MAX_i)$												

Table 5.5. Sample problem 6 x 7 x 14 (LL = 379)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$(\ell_{i+1} - \ell_i) < 1$	497	421	1133	692	636	281	791	467 498	80	80	.50	.54
$ D_i^- - D_{i+1}^- \leq .5$ and	493	415	1167	735	674	320	763	498	120	120	.52	.56
$ D_i^+ - D_{i+1}^+ \leq .5$												
$ D_i^- - D_{i+1}^- \leq .5$	497	421	1133	692	674	284	791	467 498	80	80	.50	.54
$(MIN_i - MIN_{i+1}) = 0$	497	421	1133	692	674	281	791	467 498	80	80	.50	.54
$(MIN_i - MIN_{i+1}) = 0$ and	497	421	1133	692	674	281	791	467 498	80	80	.51	.55
$(MAX_{i+1} - MAX_i) = 0$												

Table 5.6. Sample problem 6 x 10 x 19 (LL = 507)

Stopping Rules	MIN		MAX		R		MPS		N		T	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$(\ell_{i+1} - \ell_i) < 1$	694	569	1427	1024	733	455	1121	698 748 819 927	80	60	.91	.97
$ D_i^- - D_{i+1}^- < .5$ and	679	556	1521	1107	842	551	1137	748	100	80	.92	.915
$ D_i^+ - D_{i+1}^+ \leq .5$												
$ D_i^- - D_{i+1}^- \leq .5$	679	556	1521	1107	842	551	1137	748	100	80	.914	.973
$(\text{MIN}_i - \text{MIN}_{i+1}) = 0$	694	569	1427	1024	733	455	1121	698 748 819 927	80	60	.91	.97
$(\text{MIN}_i - \text{MIN}_{i+1}) = 0$ and	694	556	1427	1107	733	551	1121	748	80	80	.914	.973
$(\text{MAX}_{i+1} - \text{MAX}_i) = 0$												

Table 5.7. Sample problem 6 x 15 x 26 (LL = 750)

Stopping Rules	MIN		MAX		R		MPS		N		R	
	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS	NS	LS
$(\lambda_{i+1} - \lambda_i) < 1$	865	791	1322	1017	457	226	1127 1231 1301	821 834	60	60	1.61	1.72
$ D_i^- - D_{i+1}^- \leq .5$ and	849	789	1412	1049	563	760	1127 1304	813 834	100	100	1.63	1.75
$ D_i^+ - D_{i+1}^+ \leq .5$ $ D_i^- - D_{i+1}^- \leq .5$	849	789	1377	1033	528	244	1127	821 834	80	80	1.62	1.73
$(MIN_i - MIN_{i+1}) = 0$	865	791	1322	1017	457	226	1127 1232 1301	821 834	60	60	1.61	1.72
$(MIN_i - MIN_{i+1}) = 0$ and	849	789	1412	1049	563	260	1127 1304	823 834	100	100	1.62	1.73
$(MAX_{i+1} - MAX_i) = 0$												

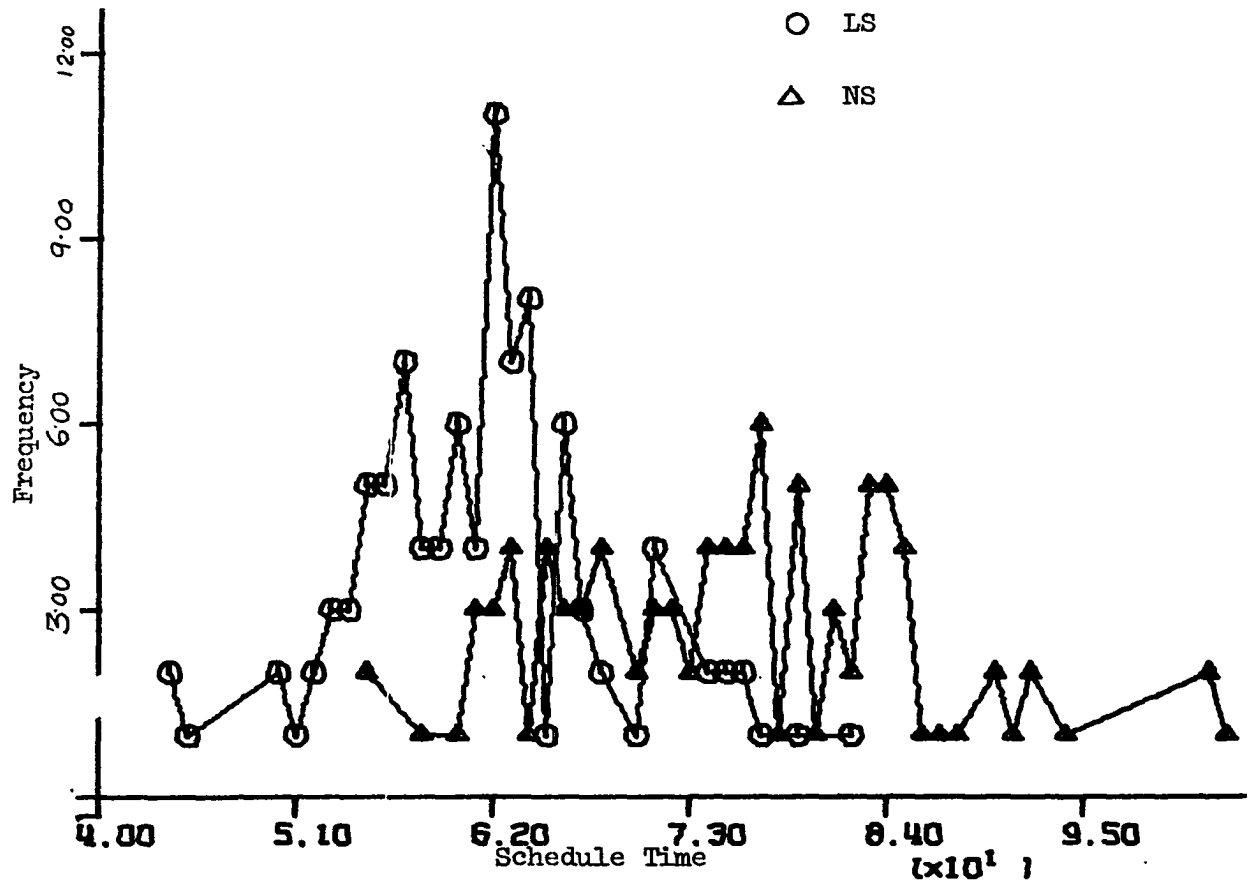


Figure 5.1. Sample problem (9 x 4 x 10)

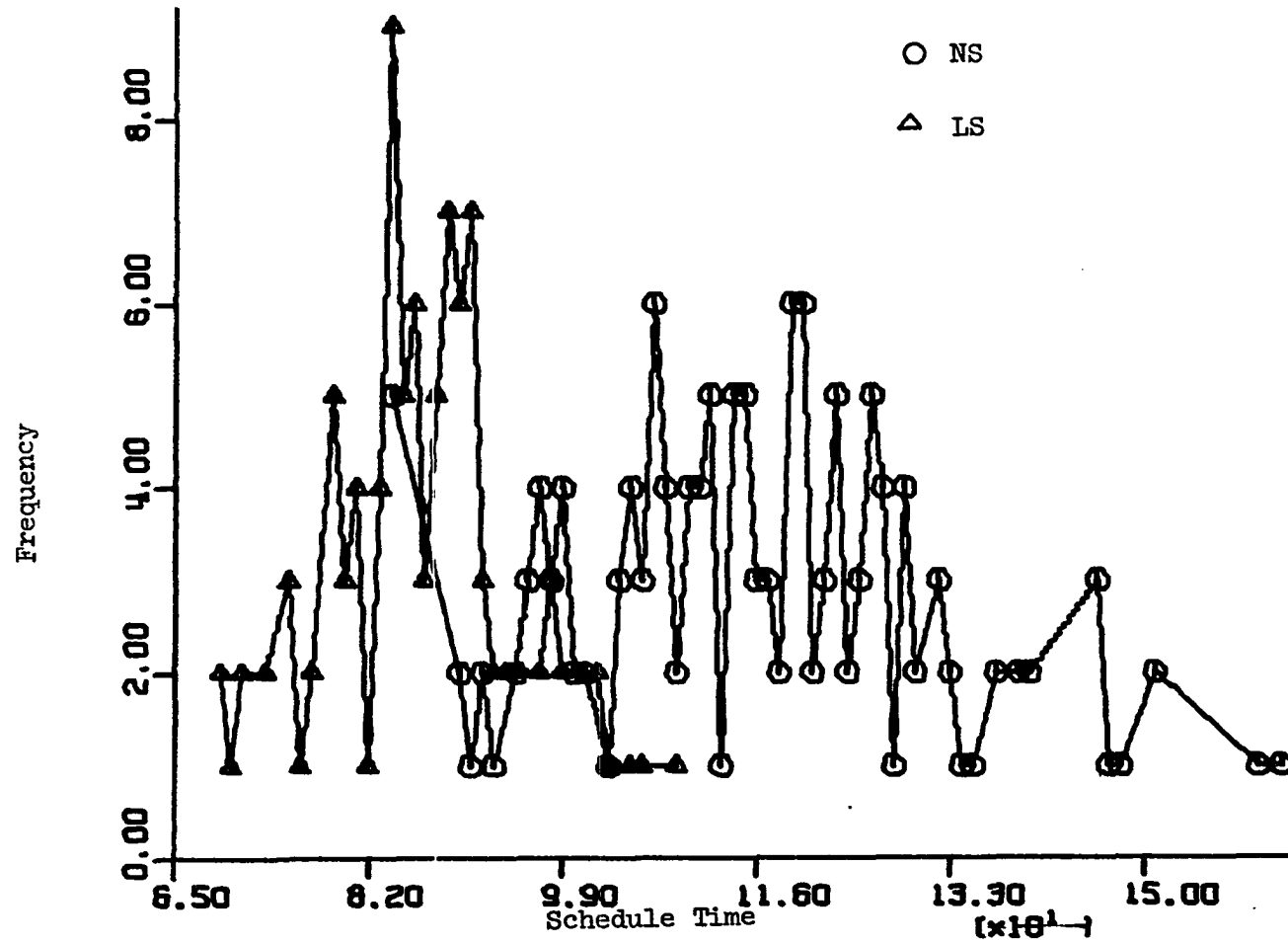


Figure 5.2. Sample problem (15 x 4 x 10; identical)

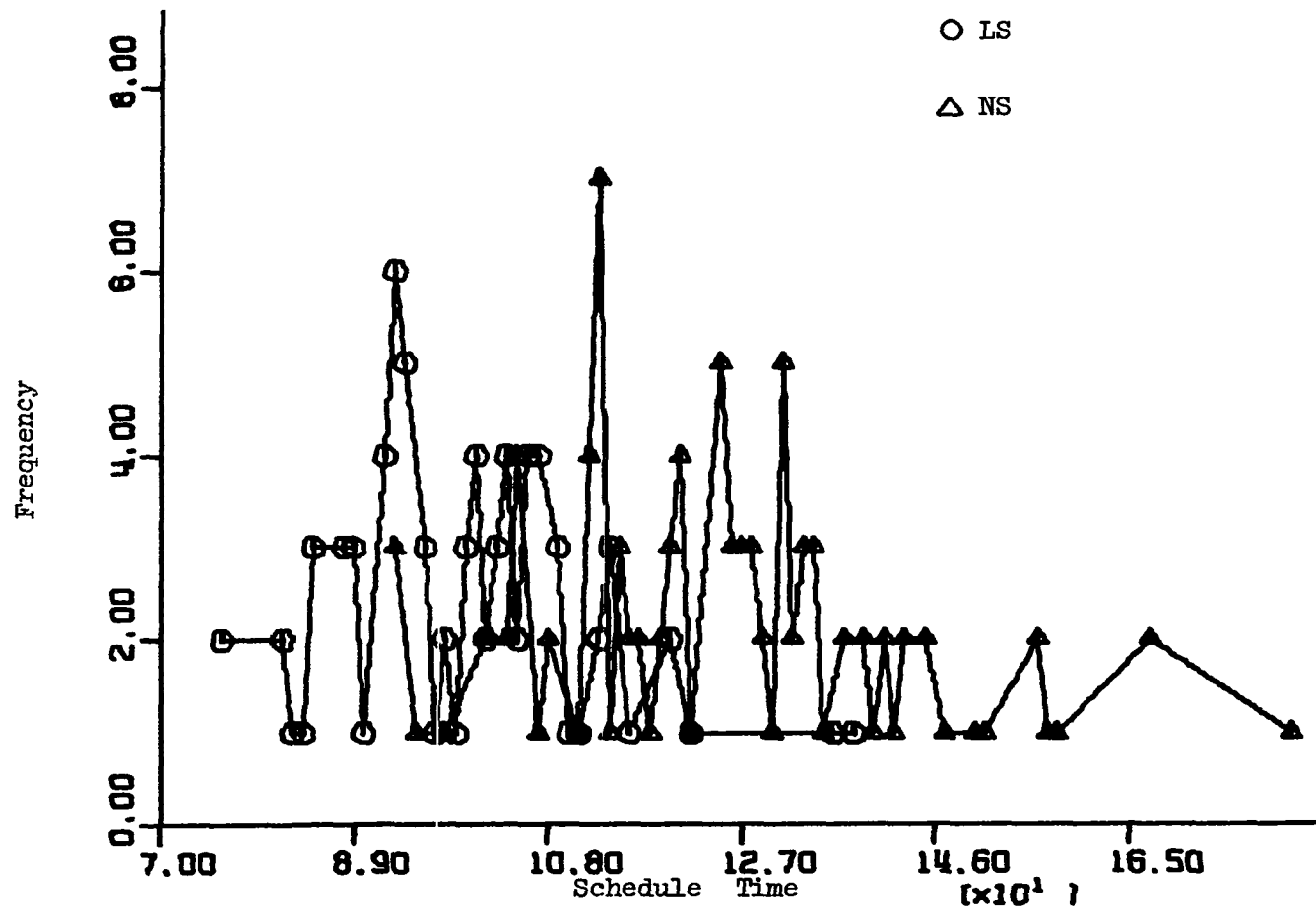


Figure 5.3. Sample problem (15 x 4 x 10; nonidentical)

C. Biasing Techniques

Biasing techniques (BT1 and BT2) and their combinations with left-shifting and non-left-shifting have been discussed in Chapter 4. Due to shortage of computer money, all these techniques could not be examined with multiple-machines facilities. A (3 x 3 x 7) problem was tested by BT2, LS and their combination. Distributions of the schedule times have been shown in figure 5.4. We observe that with respect to minimum schedule time, left-shifting is superior to BT2, but the latter has a smaller range. However, no firm conclusion can be drawn from the solution of a single problem. In fact, from experience with BT4 to be discussed shortly, BT2 should be, in most of the cases, superior to left-shifting because schedules are drawn from a better subset.

Let us now try to explore some other techniques which are unique to multiple-machines facilities.

1. BT3

This technique refers to the sampling procedure which includes two stages. At the first stage, a scheduleable operation is selected at random from a facility. Final selection is made at the second stage on the basis of the minimum starting times corresponding to the machines of that facility. Referring to the algorithm developed in Chapter 2, let the randomly selected operation refer to job j and facility f which has m machines of the same type (they may have different efficiencies). Corresponding to the operation and its counterparts, find

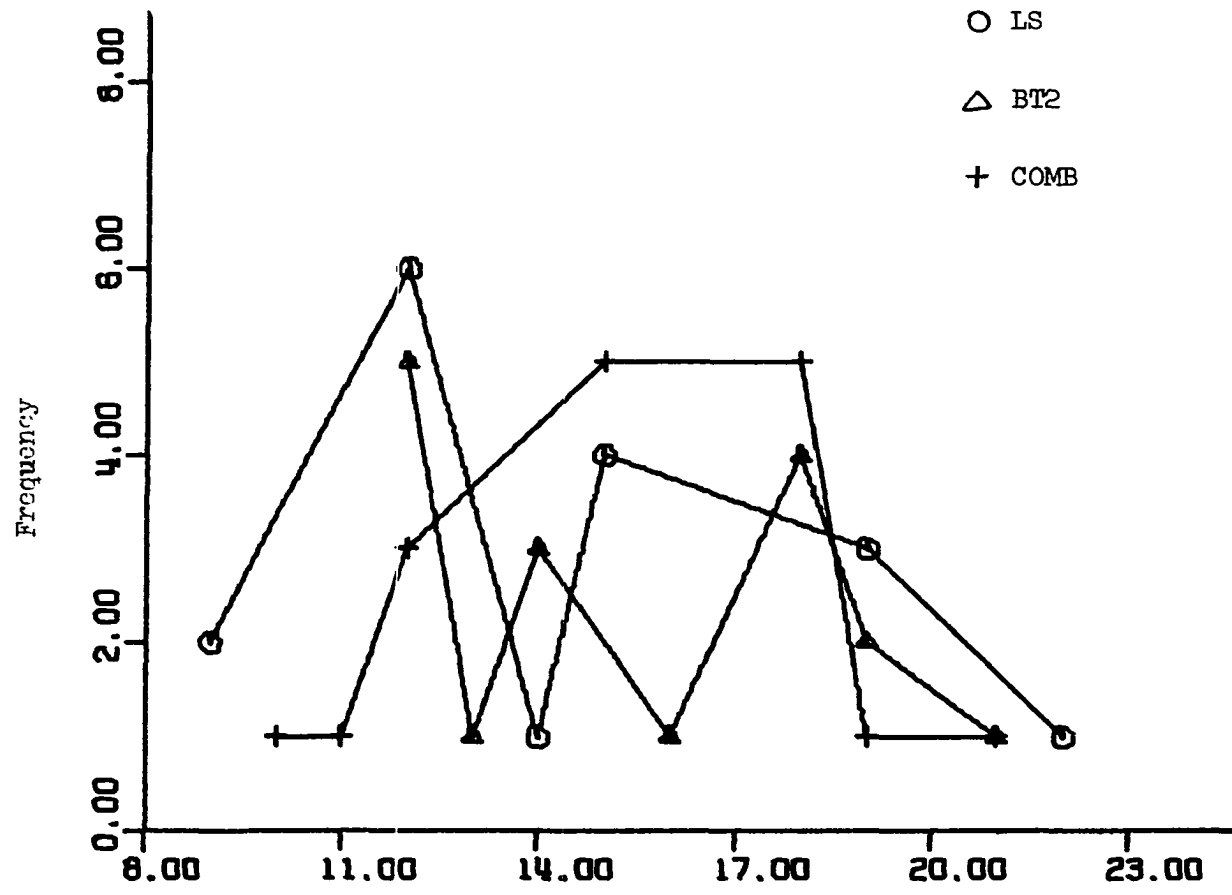


Figure 5.4. Comparison among LS, BT2 and COMB (3 x 3 x 7)

$$\text{max}_i = \max (\text{maximum machine time for machine } i \text{ and maximum job time for } j) \quad i = 1, 2, \dots, m$$

Now find a subset $Q = \min (\text{max}_1, \text{max}_2, \dots, \text{max}_m)$; select one of the operations in Q at random. In BT1, instead of sampling at random, we applied the max-min criterion over the facilities in which scheduleable operations are found. But in BT3, a scheduleable operation was randomly selected, but the final selection for actual processing was made after applying the max-min criterion over the machines of a single facility. We tested only one problem by this technique. Figures 5.5 and 5.6 show the relative merits of NS, LS, and BT3. A combination of BT3 with NS and LS can also be examined.

Both in BT1 and BT3, instead of selecting the operation based on the minimum starting time, biasing can be applied on the minimum completion time. In applying the max-min principle, we have to consider the processing time of the operation in the respective machine. As for example, for BT3

$$\text{max}_i = \max (\text{maximum machine time for machine } i \text{ and maximum job time for } j) + C^4_i \quad i = 1, 2, \dots, m$$

where C^4_i is the processing time of the operation in machine i .

2. BT4

In this technique, after selecting a process by BT3, the left-shifting criterion is incorporated. BT4 can be expressed as $\text{BT4} = \text{BT3} + \text{LS}$. BT4 must be at least as good as BT3 in any feasible solution because it is a better subset of BT3.

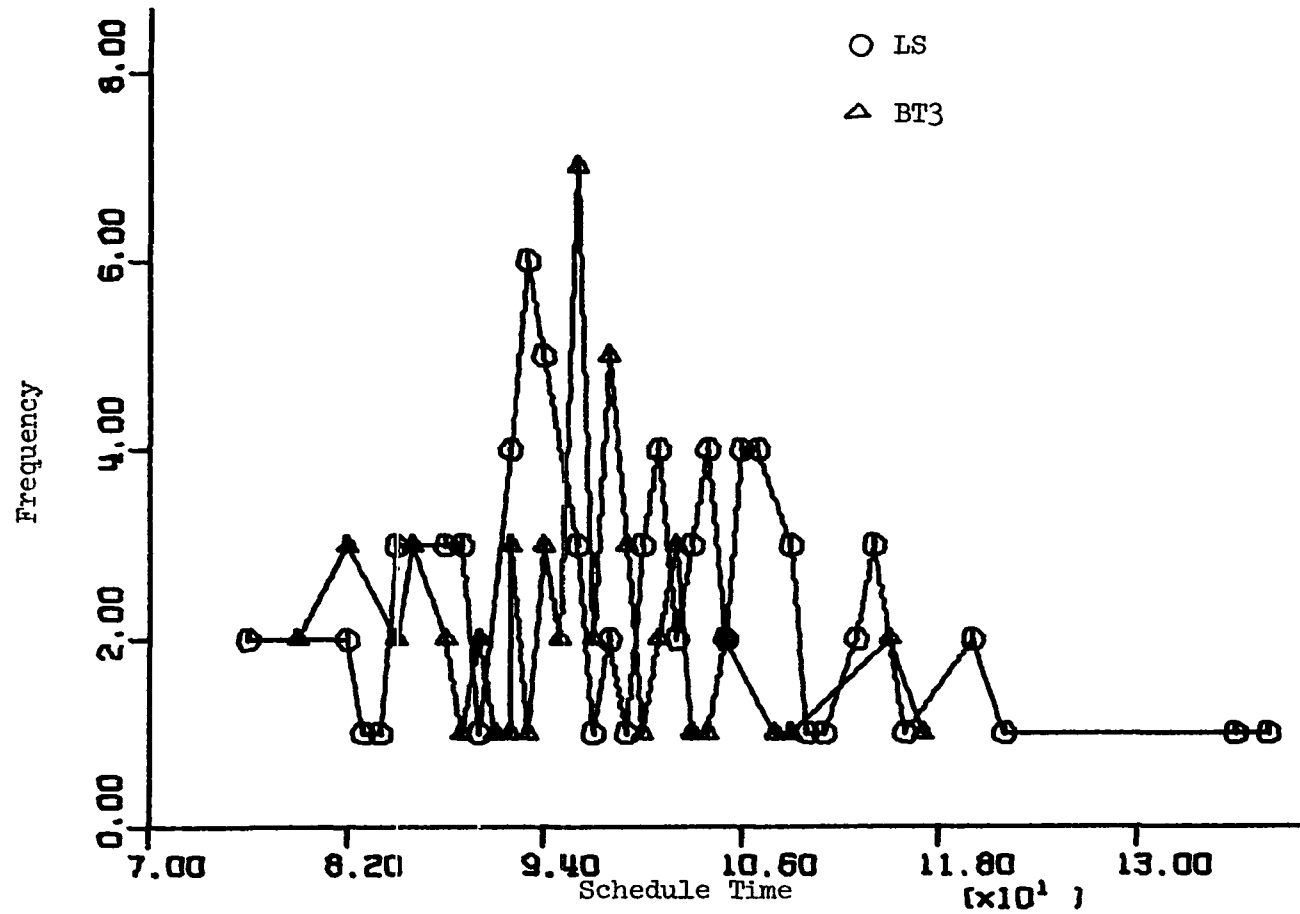


Figure 5.5. Comparison between LS and BT3 (15 x 4 x 10; non-identical)

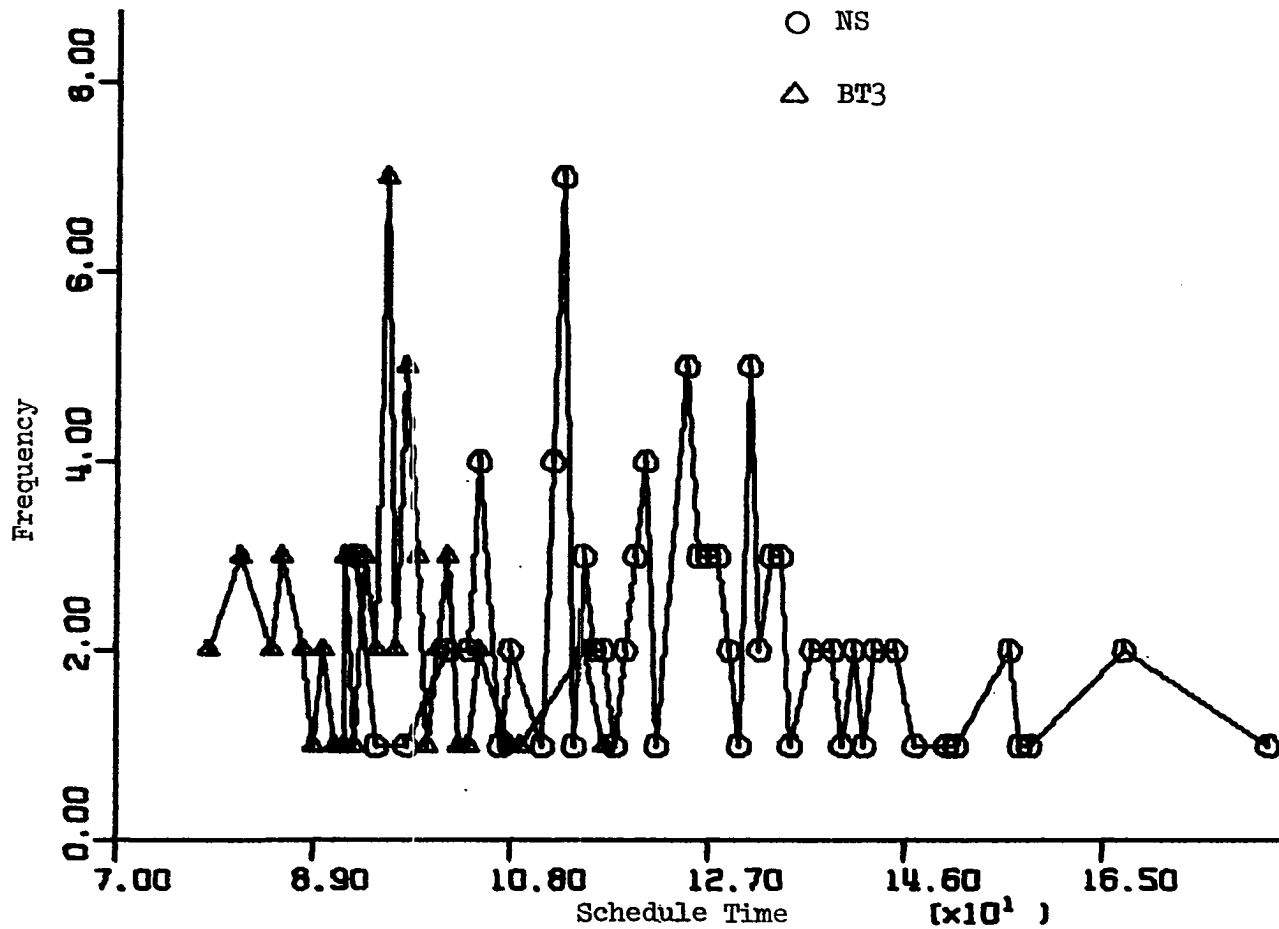


Figure 5.6. Comparison between NS and BT3 (15 x 4 x 10; non-identical)

Five problems were tested and different parameters have been entered in Table 5.8 where all the figures refer to stopping rule 4. In all the problems, we observe that BT4 is superior to LS except computer time/schedule. As the range of the distribution is smaller in BT4, we can expect that specifying a smaller sample size, we can reduce the CPU time without affecting the solution considerably. Perhaps this will lower the total CPU time in BT4 almost equal to that in left-shifting. Another alternative will be the combination of BT4 and LS. We did not actually try this combination. Figures 5.7, 5.8 and 5.9 show the comparison between LS and BT4 in their distributions for two sample problems.

3. BT2 and BT4

We notice that both BT2 and BT4 incorporate left-shifting principle in their procedures. In each case, schedules are drawn from a better subset of the set from which schedules are drawn in left-shifting. BT2 applies the max-min principle over the set of all scheduleable jobs to determine which job has the minimum starting time. This technique is analogous to first come--first serve technique in dynamic scheduling. First come--first serve rules have been found to be superior to many dispatching rules. BT2 tries to reduce the waiting time for each job.

On the other hand, BT4, which is unique for multiple-machines facilities, applies the max-min principle over the different machines of the same facility with respect to a single job. For multiple-machines facilities, the scheduler might be only interested to see

Table 5.8. Sample problems showing difference between LS and BT4

Sample Problem	Total Operations	MIN		MAX		R		MPS		BT4	
		LS	BT4	LS	BT4	LS	BT4	LS	BT4	LS	BT4
5x4x10 (Identical) (Active operations: 16)	39	25	25	43	25	18	10	32	26	.107	.197
9x4x10 (Active operations: 41) (Different from 9x4x10 used in 5.2)	104	51	43	87	75	36	32	66	52	0.52	.99
15x4x10 (Active operations: 68)	168	76	76	138	108	62	32	93	89	1.40	2.95
15x4x10 (Identical) (Active operations: 68)	168	69	68	109	95	40	27	84	74	1.4	2.95
6x10x19 (Active operations: 60)	114	569	541	1024	804	455	263	698 748 819 927	627	0.97	1.92

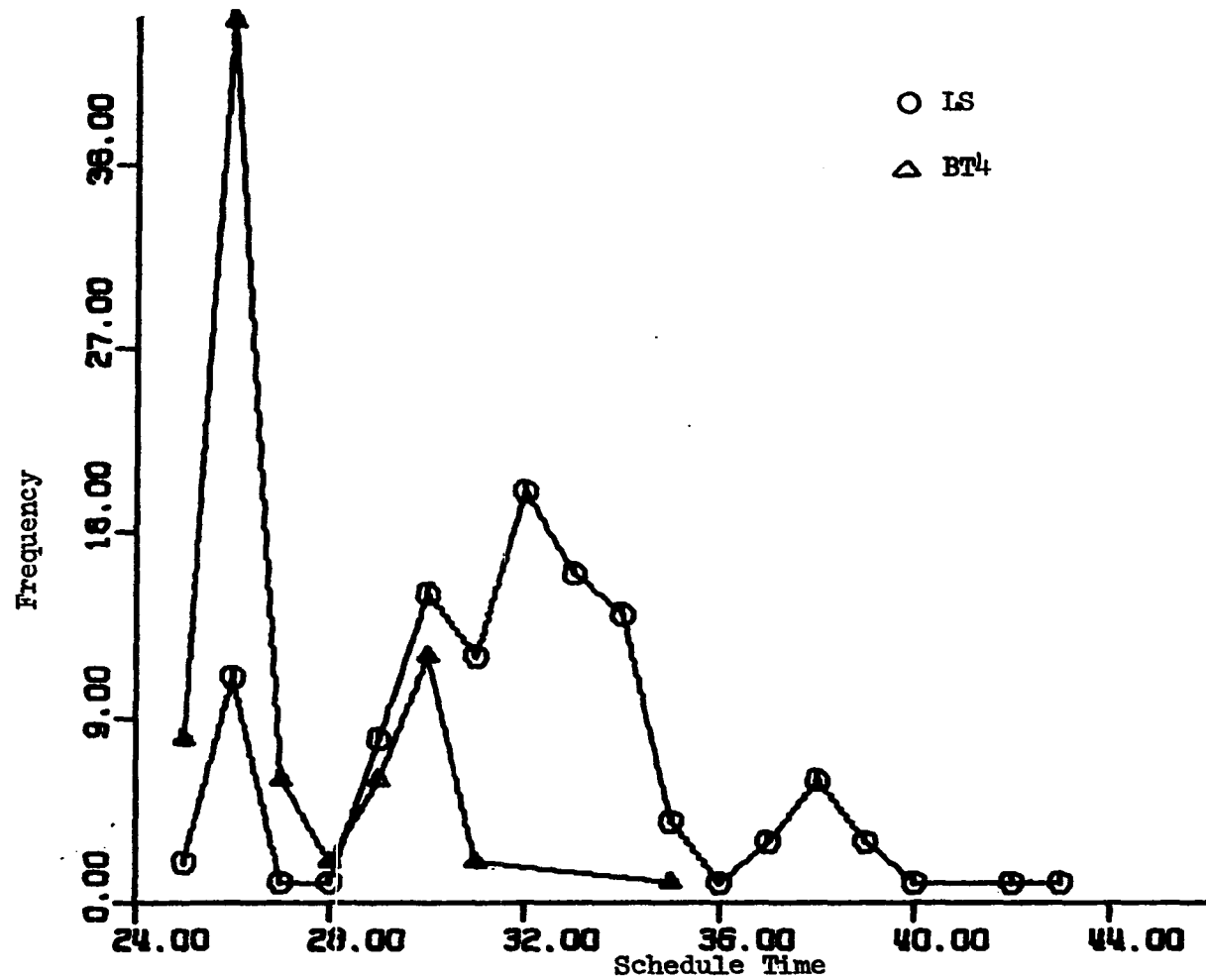


Figure 5.7. Comparison between LS and BT4 (5 x 4 x 10)

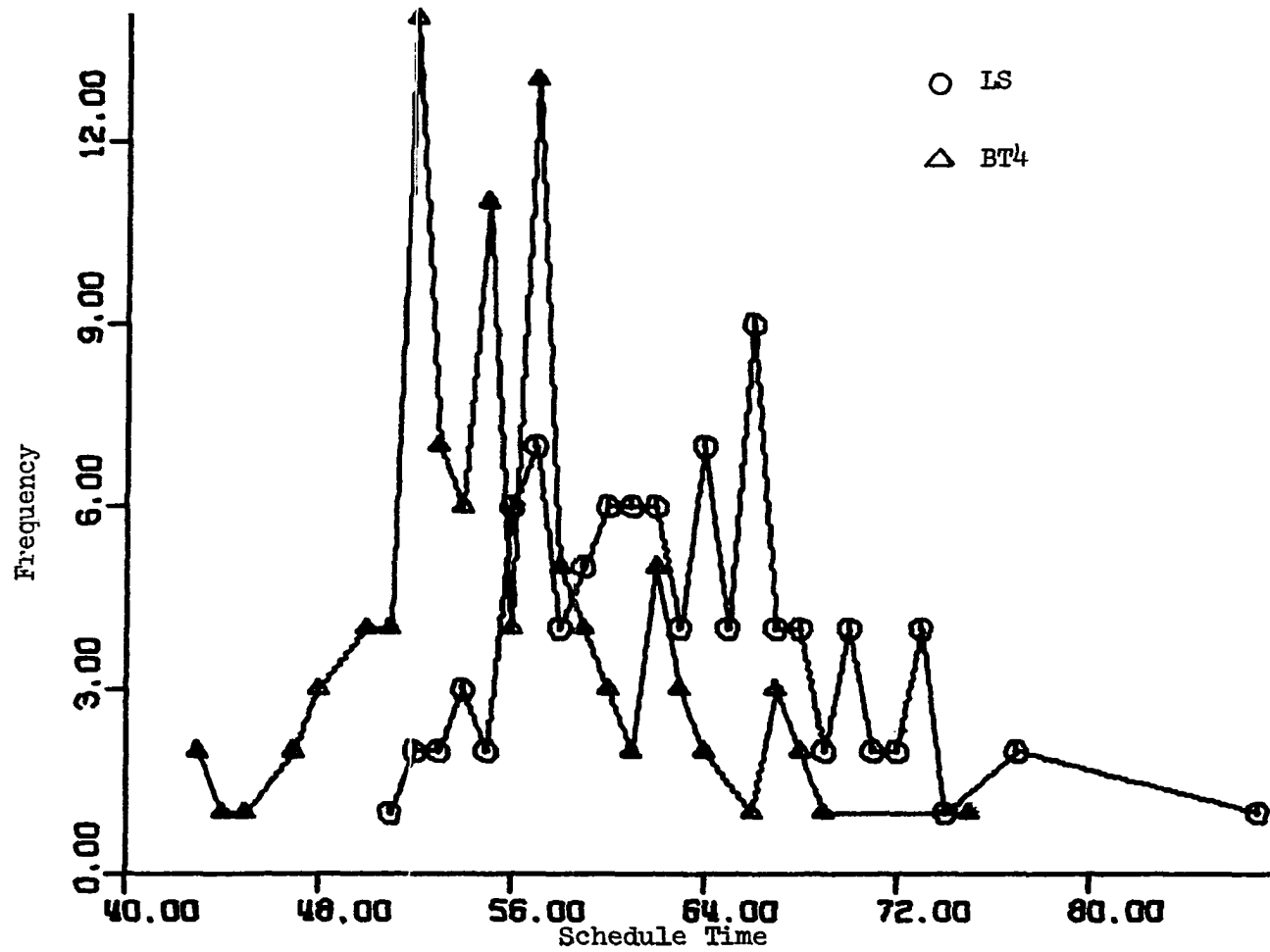


Figure 5.8. Comparison between LS and BT4 (9 x 4 x 10)

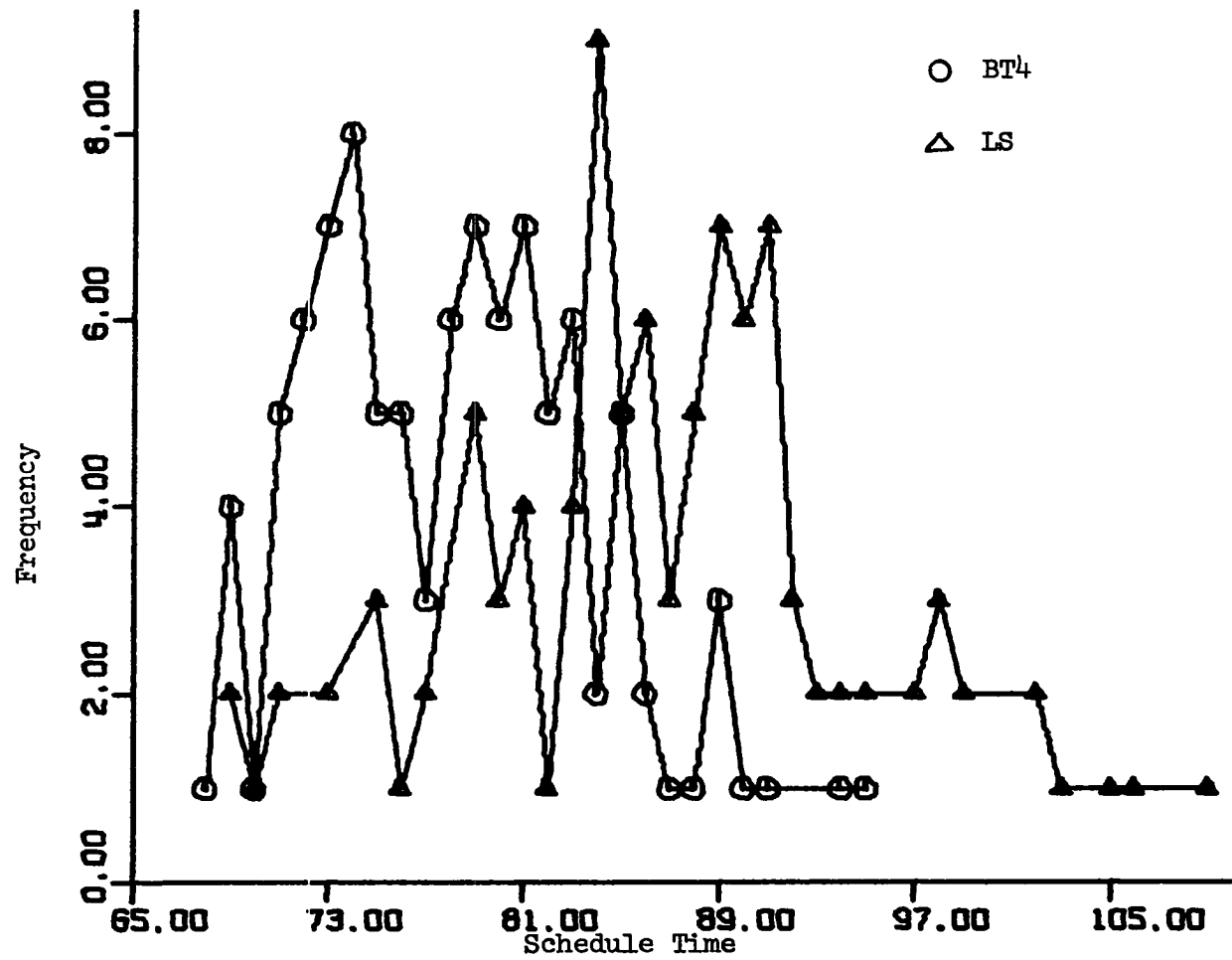


Figure 5.9. Comparison between LS and BT4 (15 x 4 x 10; identical)

that if a job is waiting to be served in a facility, which of the machines of the facility can start the job earliest. This is what is accomplished by BT⁴.

We did not actually compare BT² and BT⁴ with any sample problem. With respect to computer time, BT⁴ should have an advantage over BT² because BT² applies the max-min principle over the whole set of scheduleable operations while BT⁴ applies this principle over a single facility. A combination of BT² and BT⁴ might also be an interesting technique to look at.

4. Multiple left-shifting (MLS)

Referring to the left-shifting principle or BT⁴, suppose a particular operation is randomly selected. Instead of processing the operation by left-shifting technique or applying the max-min principle of BT⁴, we apply the left-shifting principle over this operation and its counterparts and then choose the operation which can start earliest. In this technique, the computer logic will be more complex, but it will ensure solutions to be at least as good as that either by left-shifting or by BT⁴ because solutions by this technique (MLS) will always constitute a better subset of both the sets of solutions by LS and BT⁴. Perhaps the best biasing technique (BT⁶) will be to choose the operation by BT¹ and then apply multiple left-shifting (MLS) within that facility. This will be undoubtedly better than BT² because left-shifting is applied over all the counterparts. Here two biasing techniques are applied in series. First, it considers all the jobs to determine

which job can start fast (BT1) and then put the operation in the machine which can start processing first (MLS).

No computer program has been written for MLS. Further research will be needed in exploring the different aspects of above biasing techniques and development of better techniques in order to increase the efficiency of the Monte Carlo sampling procedure used in this dissertation.

We summarize in figure 5.10 the different biasing techniques arranged downward in the order of their increasing efficiency.

D. Single Machines Versus Multiple Machines

In multiple-machines facilities, different machines of the same facility can remain simultaneously engaged for different jobs. So as the number of machines increases in different facilities, the lower limit (LL) of the schedule times will be less dependent on $\sum P(i,j)$ and it may be defined as mentioned earlier as $\max_j (\sum P(i,j))$ for all i , where $P(i,j)$ is the processing time of the job i on the most efficient machine of the facility j . Consequently, in multiple-machines facilities, by increasing the number of machines, it is always possible to have a feasible situation having the schedule time equal to the lower limit. But considering only this particular aspect, one should not be motivated to shift from single-machine facilities to multiple-machines facilities. Unless the machine utilization factor is taken into consideration, even by reducing the span of the schedule time by shifting to multiple-machines facilities, the overall cost may not justify the change. Let us consider two sample problems (5 x 4) and (15 x 4) analyzed in Chapter 4 and their

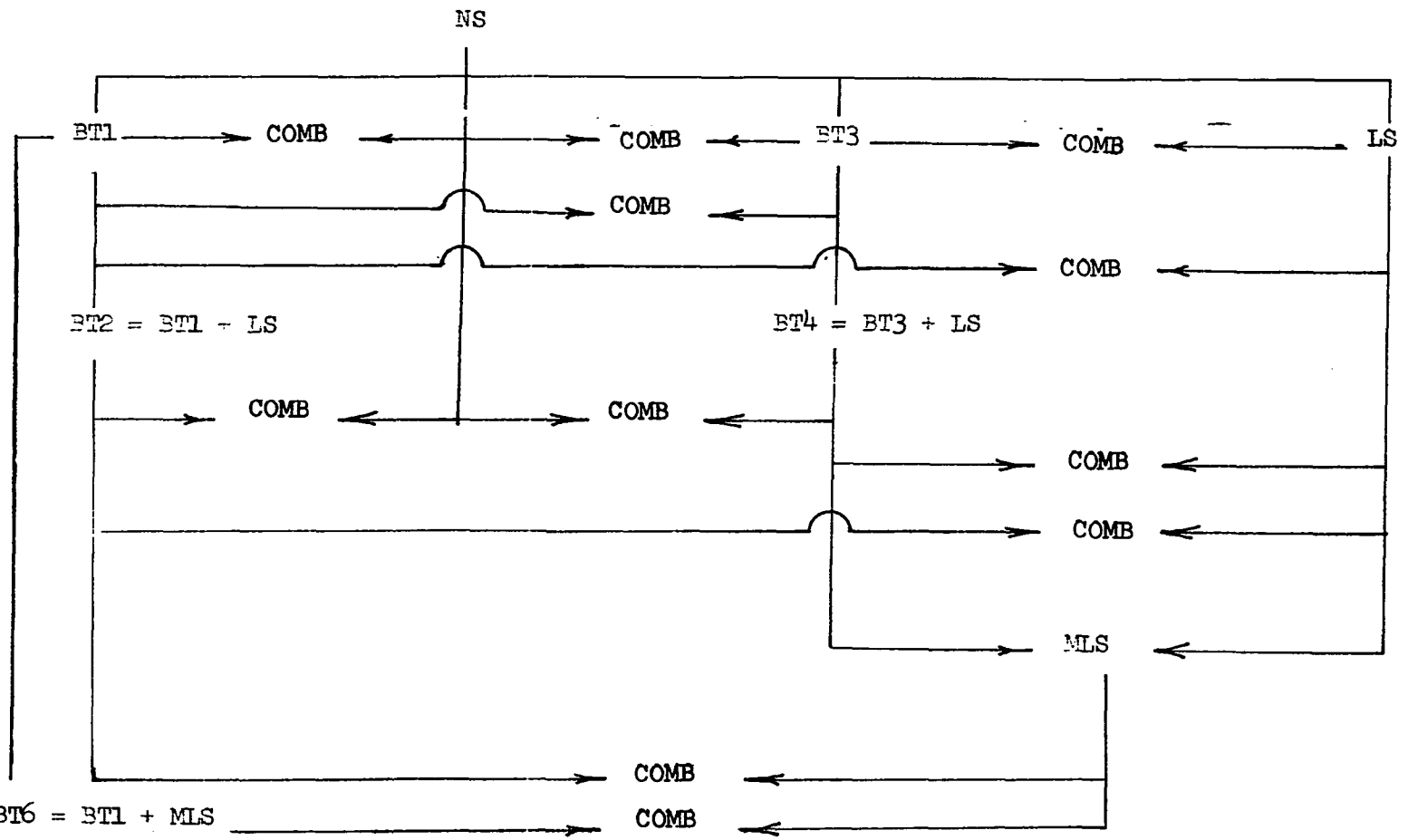


Figure 5.10. Biasing techniques in multiple-machines facilities

multiple versions (5 x 4 x 10) and (15 x 4 x 10) analyzed at the beginning of this chapter to see the increase in the idle time of the facilities when single-machine facilities are arbitrarily changed to multiple-machine facilities. Figures 5.11 and 5.12 display the Gantt charts of the two problems in single-machine facilities, and figures 5.13 and 5.14 give the corresponding Gantt charts of their multiple versions. For the sake of discussion let us consider the schedule time to constitute a cycle and we will determine the portion of the cycle time each machine remains idle. From the Gantt charts, idle time for each machine is calculated and entered in Table 5.9.

As mentioned in Tables 5.2 and 5.4, the lower limits for the two problems (5 x 4 x 10) and (15 x 4 x 10) are 25 and 38, respectively. From Table 5.9, we can make the following observations.

(1) Switching from single machine facilities to multiple machines facilities moved the schedule times towards their lower limits in both problems, but the shift is more considerable in the problem having more jobs (15 x 4 x 10).

(2) In both the problems, idle time/cycle time increases in each original machine, and additional idle times occurred due to additional machines in each facility.

(3) Except for A1, A3 and B2, idle time/cycle time is considerably less in the (15 x 4 x 10) problem.

(4) An approximate idea regarding facility utilization can be obtained considering the idle times of the machines within each facility. Considering cost/idle time as the same for each machine (which is in

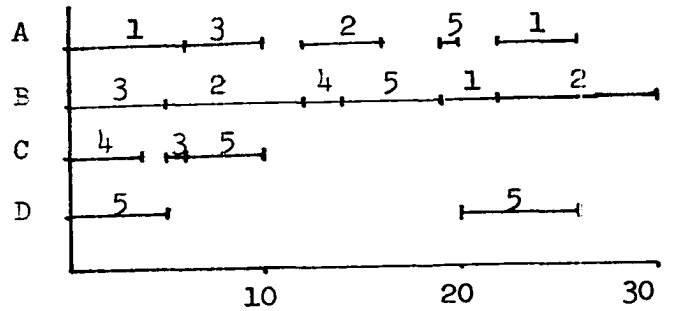


Figure 5.11. Gantt chart for problem 5 x 4 (single)

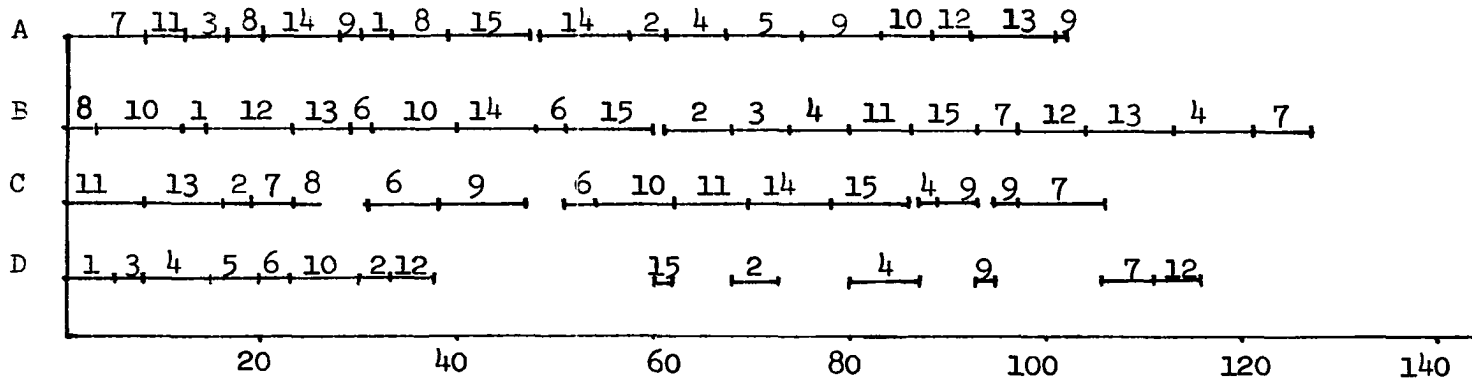


Figure 5.12. Gantt chart for problem 15 x 4 (single)

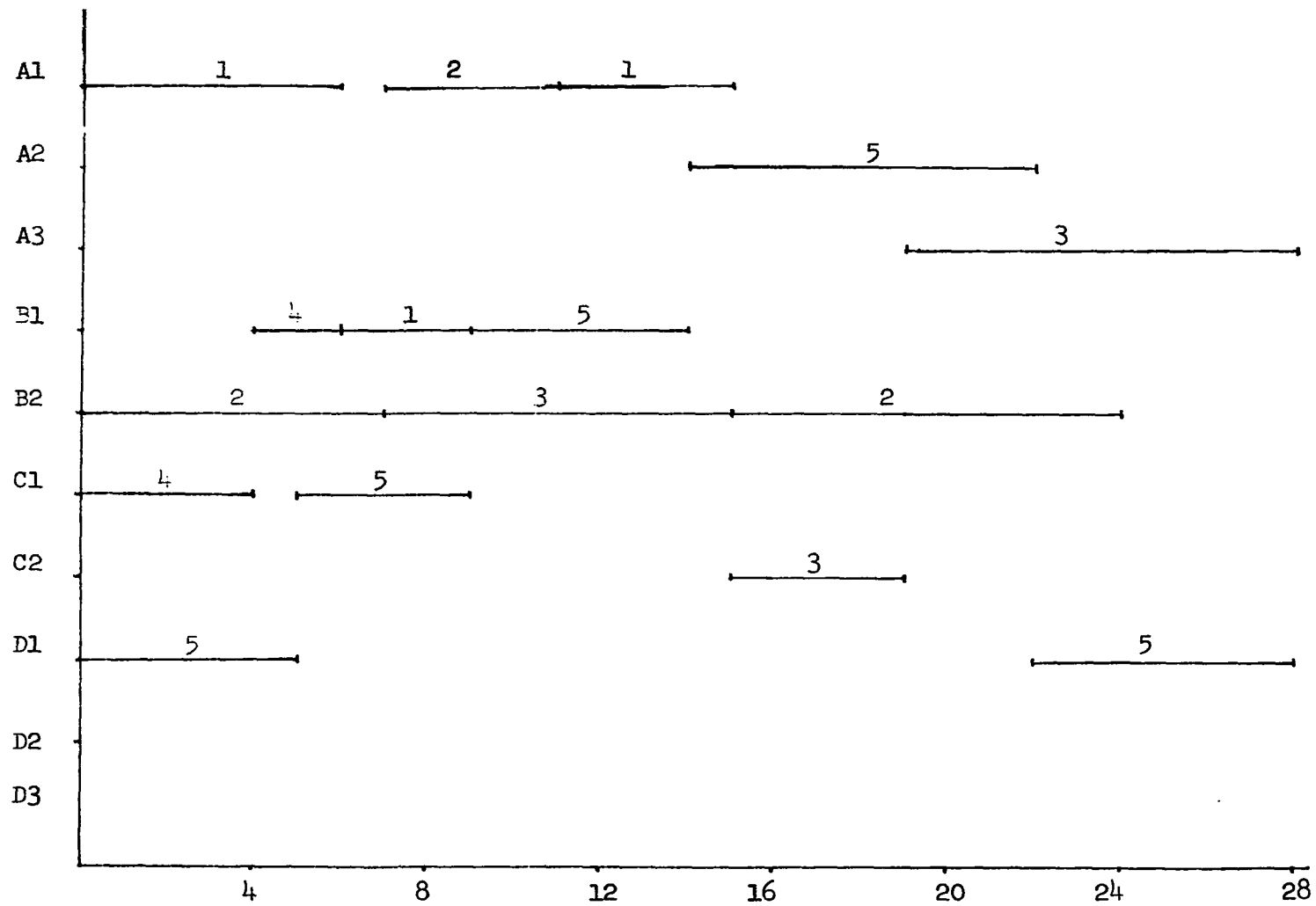


Figure 5.13. Gantt chart for problem $5 \times 4 \times 10$ (multiple)

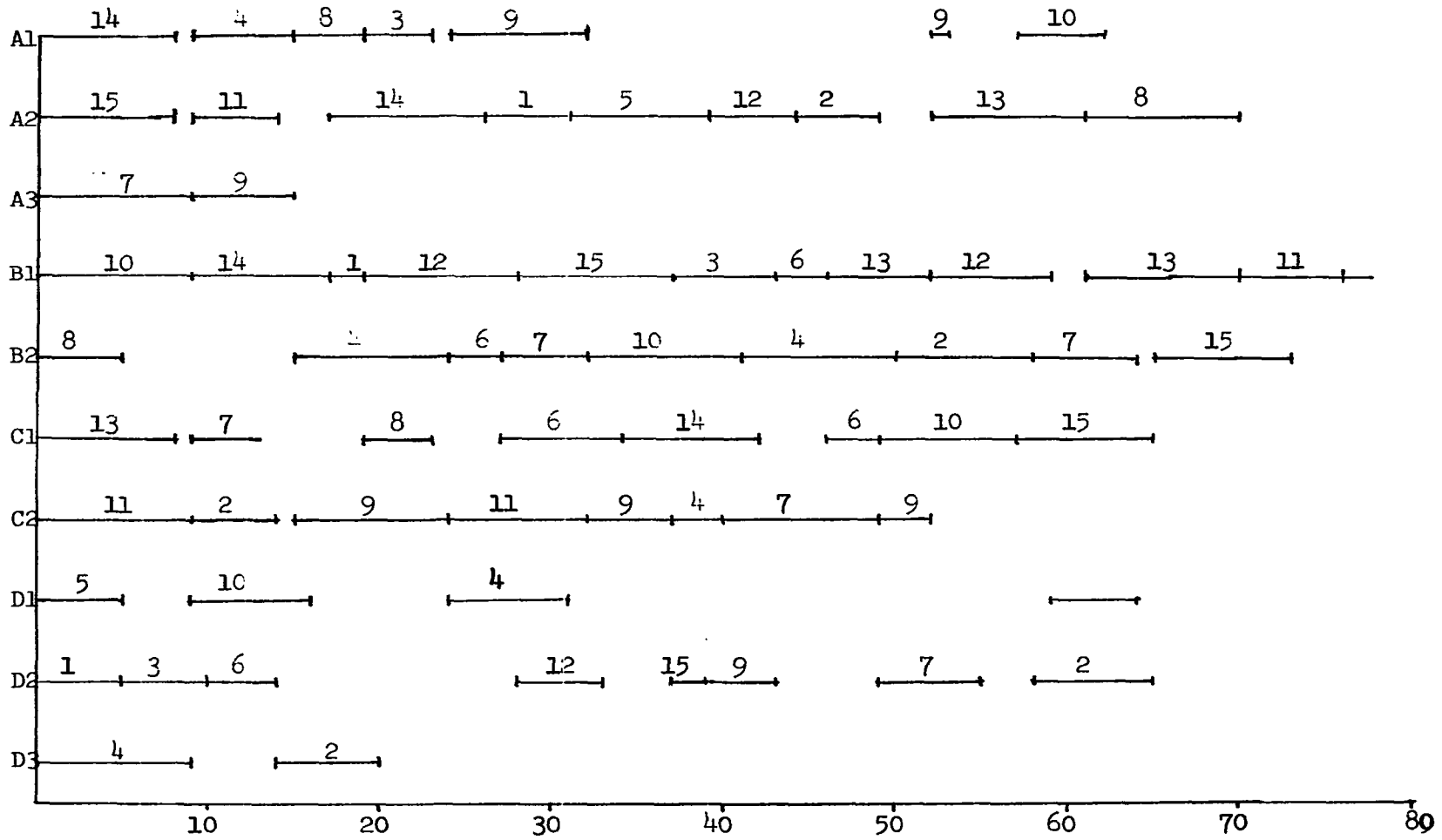


Figure 5.14. Gantt chart for problem 15 x 4 x 10 (multiple)

Table 5.9. Single machine vs multiple machines

Machines	Single-Machine Facilities						Multiple-Machines Facilities					
	<u>Cycle Time</u>		<u>Idle Time</u>		<u>Idle Time/ Cycle Time</u>		<u>Cycle Time</u>		<u>Idle Time</u>		<u>Idle Time/ Cycle Time</u>	
	5x4	15x4	5x4	15x4	5x4	15x4	5x4x10	15x4x10	5x4x10	15x4x10	5x4x10	15x4x10
A1	30	127	11	26	0.37	0.20	28	76	14	40	0.50	0.53
A2							28	76	20	14	0.71	0.18
A3							28	76	17	61	0.61	0.80
B1	30	127	0	1	0.00	0.01	28	76	18	2	0.67	0.03
B2							28	76	4	14	0.14	0.18
C1	30	127	21	32	0.70	0.25	28	76	20	27	0.71	0.36
C2							28	76	24	25	0.86	0.33
D1	30	127	19	69	0.63	0.54	28	76	23	52	0.82	0.59
D2							28	76	22	38	0.79	0.50
D3							28	76	28	60	1.00	0.79

fact an oversimplified assumption), let us define an approximate measure of the facility utilization by

$$u_f = \frac{\sum_{i=1}^m (\text{Cycle time} - \text{Idle time for machine } i)}{m_f \times \text{cycle time}}$$

u_f = utilization factor for facility f

m_f = number of machines in the facility f

Table 5.10 shows the utilization factors for each facility. These are calculated from Table 5.9. F mentioned in Table 5.10 refers to the overall system.

Table 5.10. Utilization factor for facilities

Facilities	Single-Machine Facilities		Multiple-Machines Facilities	
	5 x 4	15 x 4	5 x 4 x 10	15 x 4 x 10
A	63	80	39	50
B	100	99	61	89
C	30	75	21	66
D	37	46	13	34
F	58	75	32	56

The utilization factors in multiple-machines facilities for the problem (15 x 4 x 10) are consistently greater. This is expected because the machines are to process more jobs and thereby have less idle time. In Table 5.10, if we concentrate on the problem (15 x 4 x 10) in multiple-machines facilities, the difference in the utilization factors for the different facilities can be explained if we consider the increase in the number of machines in each facility and the number of operations each has to perform. The processing times were drawn from a random number table and, therefore, there should not be any bias to any particular facility with respect to processing times. From Gantt chart (figure 5.12) the number of operations performed in each facility was calculated. These are entered in Table 5.11

Table 5.11. Number of operations and increase in machines

Facilities	Number of Operations	Increase in Machines	U_i 's for 15 x 4 x 10 (from Table 5.10)
A	18	2	50
B	20	1	89
C	16	1	66
D	14	2	34

From Table 5.11, we can conclude that the utilization factor for facility B is highest because of the maximum number of operations in this facility and the fewer number of machines. By similar reasoning, we can justify the worst utilization of the facility D. The relative advantage of the facility over the facility A can be explained by the likewise argument.

On the basis of the above discussion on single machine versus multiple machines, we make the following remarks.

The very first step that occurs when shifting from single-machine facilities to multiple-machines facilities is to find which area has the potential need for change. In intermittent industries, usually every job needs some common operations and, therefore, the possible needs for change to multiple-machines facilities should more likely arise from general purpose machines rather than special purpose machines. From past experience management should realize mostly which jobs are arriving and the area of greater accumulation of in-process inventory.

After management is convinced about the potential need for change in some particular area, the next question which arises is how many additional machines are required. From past records of job arrival, management should be able to roughly calculate the expected number of different jobs in the shop at a particular time. On the basis of this set of jobs, computer simulation may be carried on to determine the "optimum" number of machines needed at a particular facility. A cost function must be defined and varying the number

of machines, a trade-off point should be determined where dollars saved by reducing the schedule time and in-process inventory can at least justify the increased expenses due to cost, idle time, maintenance, etc., of the additional machine units.

E. Sample Size and Stopping Rules

Referring to Tables 5.2 through 5.7, we notice that like single machine facilities, different stopping rules need a different number of schedules to meet their criteria. In the first three sample problems, stopping rule 2 did not show any improvement in the minimum schedule though it took more CPU time. This may be due to structure of the problem itself. Actually, the algorithm converged earlier for these problems and higher number of schedules also could not provide any better result as shown in Table 5.12.

As mentioned earlier, the difference between the minimum schedule time and its lower limit (LL) depends on the number of machines in each facility. Usually this difference increases relatively with the increase of jobs in the system. Because at present we do not have an exact relationship between this difference and the structure of the problem, we suggest stopping rule 6 instead of stopping rule 5. Since a "reasonable" lower bound for multiple-machines facilities is not available, stopping rule 5 cannot be applied even if CPU time is a critical factor.

For the usual situation, as in single machine facilities, stopping rules 1, 3 and 4 can be applied. Of these three, stopping rule 4 is preferred in view of the CPU time and better result obtained.

Table 5.12. Relative improvement of MIN with number of schedules (LS)

Sample Problems	Number of Schedules Drawn					
	40	60	80	150	250	500
5 x 4 x 10	28	-	28	-	28	26
9 x 4 x 10 (used in 5.3)	55	-	51	44	-	44
15 x 4 x 10	84	76	76	-	76	76
6 x 7 x 14	435	-	421	415	-	412
6 x 10 x 19	569	569	-	556	552	541
6 x 15 x 26	813	791	789	789		781

Considering the sample size, as in Chapter 4, let us observe the relative improvement of the minimum schedule with the increase in the number of schedules (Table 5.12). The numbers in Table 5.12 are the minimum schedules (MIN) with respect to left-shifting. From Table 5.12, we observe that in most of the cases 500 schedules gave a better minimum than that from 80 or 150 schedules, but the improvement may not justify its worth where CPU time is taken into consideration.

Before we specify the sample size for each stopping rule, let us consider the four sample problems from Table 5.12 to show the relative improvement of "MIN" when BT4 is applied. Table 5.13 shows this improvement.

Table 5.13. Relative improvement of MIN by BT4

Sample Problems	Number of Schedules Drawn				
	40	50	80	100	200
5 x 4 x 10 (Identical)	25	-	25	25	-
9 x 4 x 10 (used in 5.3)	44	43	-	43	43
15 x 4 x 10 (Identical)	-	68	-	68	68
6 x 10 x 19	551	-	541	-	541

Considering Tables 5.12 and 5.13, it is very obvious that different simple sizes should be specified for LS and BT4. In fact, in BT4, smaller sample size should be used because, in this technique, convergence of the solution is faster and the quality is also better. This will compensate to a great extent the larger CPU time/schedule in BT4.

All the stopping rules need two or more samples to be drawn before the criterion is met. Stopping rule 2 needs more samples. With this above information, in Table 5.14, we suggest a rough guide set of rules for the sample size to be specified in the program.

Table 5.14. Stopping rules and sample size (multiple machines)

Stopping Rules	Sample Size	
	LS	BT4
1	40-50	20-30
2	30-40	15-20
3	30-40	15-25
4	40-50	20-30
6	30-40	15-25

F. CPU Time

As in single-machine facilities, there is not much difference in CPU time/schedule between left-shifting and non-left-shifting.

The difference between the different stopping rules with respect to CPU time/schedule is not very significant. However, on the average, stopping rule 2 takes relatively more CPU time. Stopping rules 3 and 6 take almost the same time. Stopping rules 1 and 4 usually require the least time on the average.

As mentioned earlier, CPU time per schedule in BT4 is almost double that in left-shifting.

In multiple-machines facilities, CPU time per schedule is less than that in corresponding single-machine facilities. This is due to the fact that the "active" operations are less than the total possible operations in multiple-machines facilities.

Figures 5.15 through 5.17 show the increase in CPU time/
schedule with the increase in problem size in different techniques.

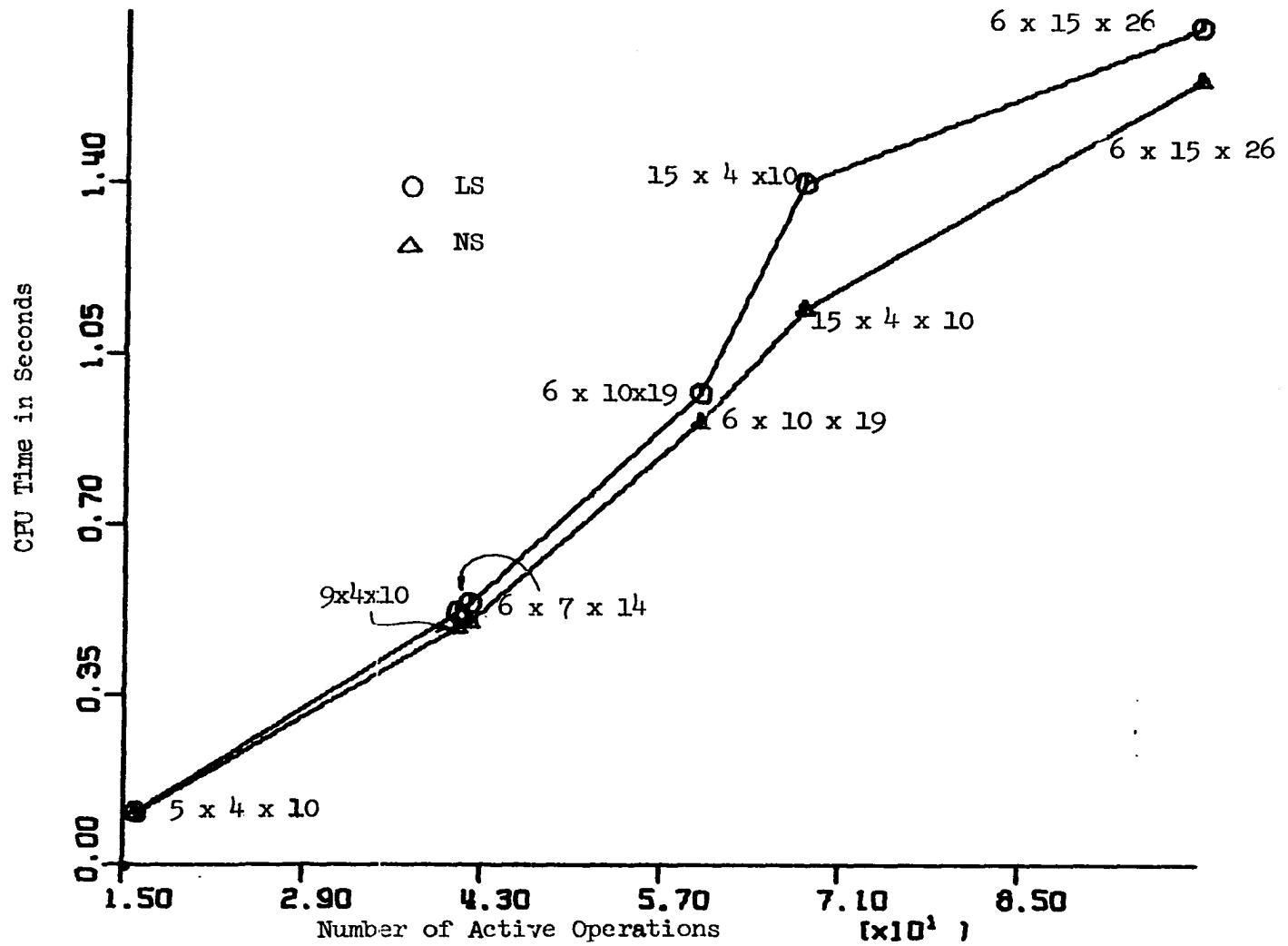


Figure 5.15. CPU time (LS and NS)

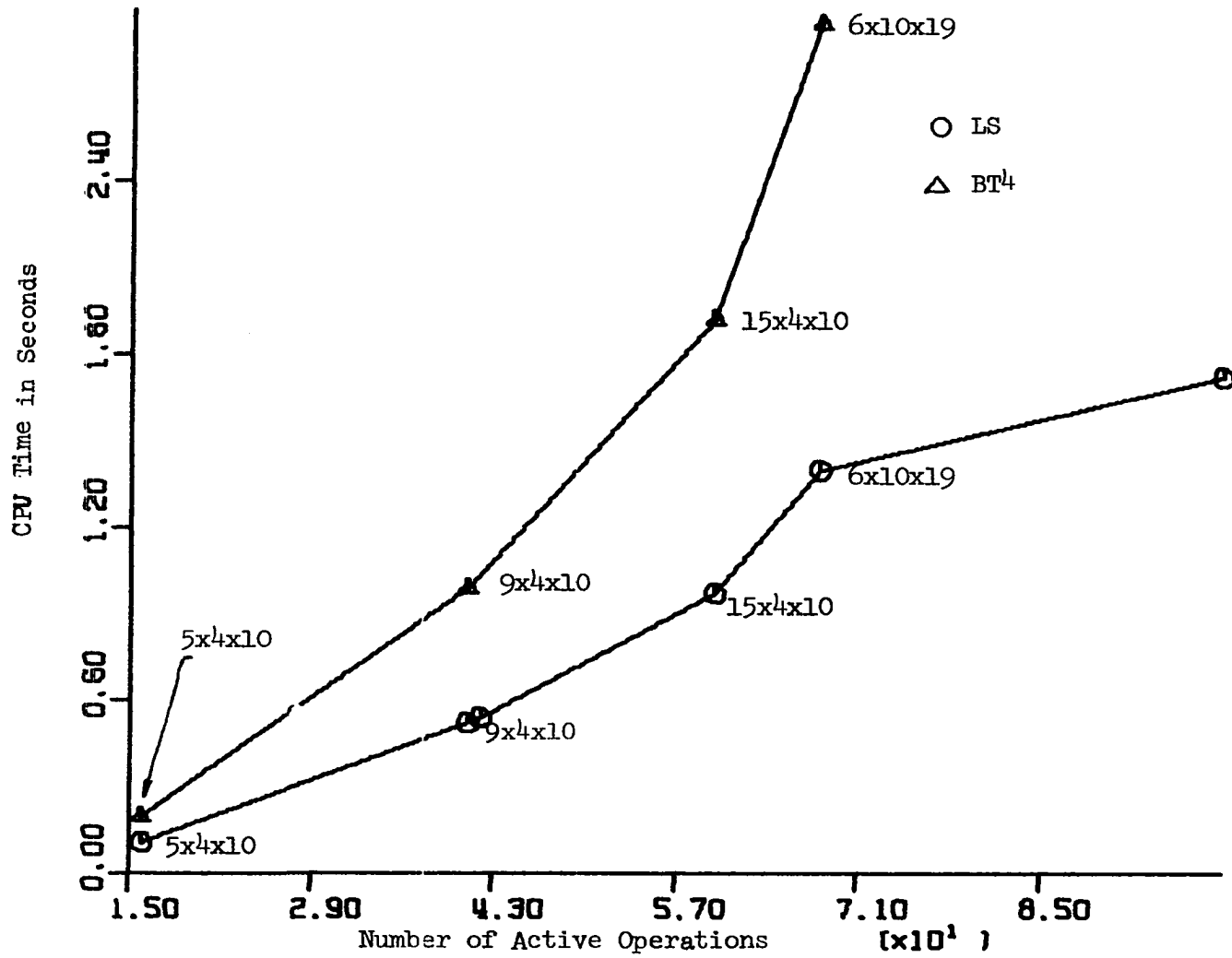


Figure 5.16. CPU time (LS and BT⁴)

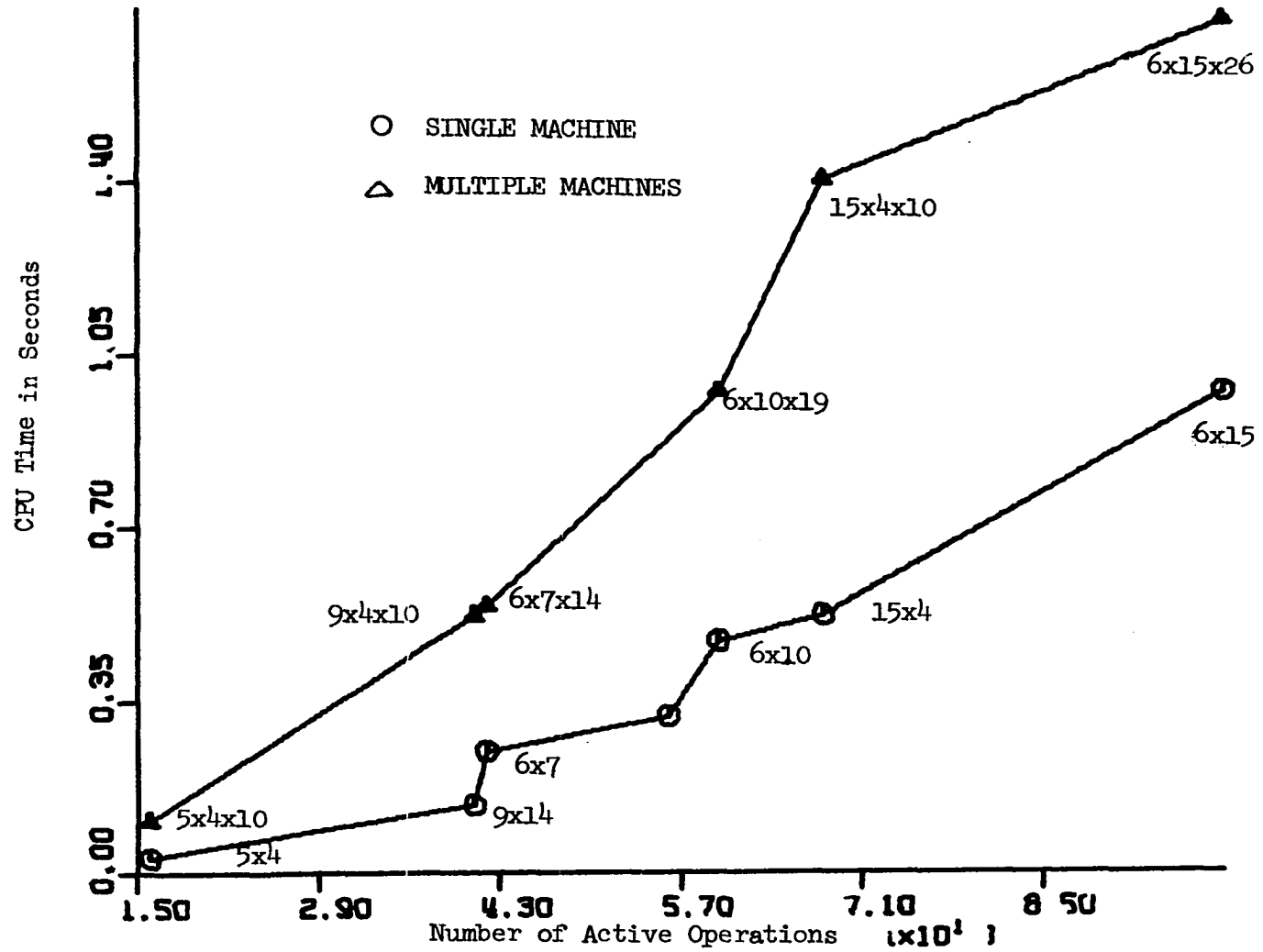


Figure 5.17. CPU Time (Single and multiple machines; LS only)

VI. MINIMUM BOUND ESTIMATION

A. Introduction

For a large combinatorial problem, it is very difficult to obtain the true optimal solution. Different techniques, such as the Monte Carlo sampling procedure used in this dissertation, provide near optimal solutions in most cases. For practical purposes, the near optimal solutions may do just as well. However, in this case, the scheduler might be interested in knowing the different information regarding the minimum schedule obtained as to its "closeness" to optimal or some "estimated optimal" schedule.

The different stopping rules used in this dissertation provide some justifications to the scheduler to believe that the best schedule so obtained should be close to the "best" obtainable schedule by the algorithm. Further information regarding the minimum schedule can be provided if an estimate for the lower bound is obtained. If x_{min} is such an estimate and x is the value of the schedule so far obtained, then, as we indicated in Chapter 3, the probability of further improvement and the difference $(x - x_{min})$ will provide still other dimensions for decision making. Further, the estimated minimum, x_{min} , can be used as a basis for the bound B_d to be used in stopping rule 5 discussed in Chapter 3.

This chapter will concentrate on estimating the minimum schedules (x_{min}) for different problems and interpreting other associated parameters with respect to those problems. Extreme value theory, which has its wide application in a variety of meteorological and engineering problems, is the

basic mechanism for estimating the minimum schedule. Gumbel (1958) provided a detailed description of the underlying theory along with the three types of asymptotes associated with it.

McRoberts (1971) first demonstrated the usefulness of extreme value theory in plant layout problems. In this application, the third asymptotic, which is the three parameter Weibull distribution, was used as the basis for estimating the minimum bound value. The third asymptote for the smallest values arises when the underlying population distribution is bounded from below. In this dissertation, the procedure as outlined in the above paper will be followed for estimating the minimum bound value.

There are two basic hypotheses established in applying the theory to the combinatorial optimization problems (Bae, 1972):

(1) The near optimal values resulting from some powerful algorithmic treatment are equivalent to the smallest values of random samples of a large size.

(2) The Weibull distribution "adequately" describes the behavior of these smallest values.

The second hypothesis takes advantage of the fact that a Weibull distribution provides some flexibility and absorbs possible inaccuracy in describing the behavior of the random variable near the bounding value (Bae, 1972). Empirical studies of the application of the Weibull to the combinatorial problems have verified its applicability (McRoberts, 1966). The distribution is a mathematical function describing the behavior of the lowest (or highest) values taken from samples independently drawn from a parent population. The function itself is independent of the parent

distribution function and has the distinct advantage of having as one of its parameters, the boundary value of interest (McRoberts, 1971).

The Weibull function describing the behavior of the extreme-value statistics is in the cumulative form (Gumbel, 1958),

$$F(X) = 1 - \exp \left\{ - \left[\frac{(X - e)}{(V - e)} \right]^K \right\} \quad (1)$$

where

X = criterion or the smallest sample value;

$F(X)$ = the probability that the x_{min} is equal to or less than x

e = location parameter or the bounding value, i.e., a constant equal to the lowest value of x_{min} ;

V = a constant parameter indicating the value of the variable such that the probability that x_{min} is equal to or less than V is approximately 0.63 referred to as the characteristic smallest value in extreme-value theory;

K = a constant parameter indicating the shape of the distribution. The distribution will be positively skewed, symmetrical, or negatively skewed, depending on whether K is less than, equal to, or greater than 3.28.

The different parameters of the Weibull distribution have been shown in figure 6.1

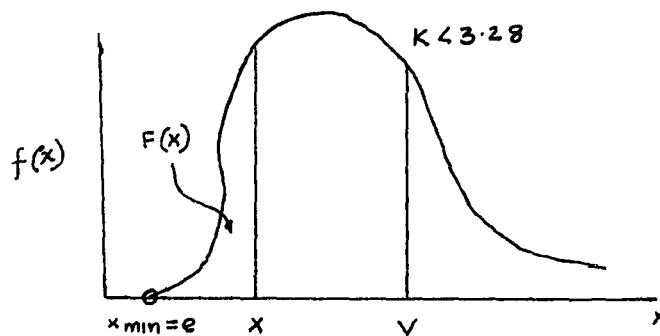


Figure 6.1. Parameters in Weibull distribution

When the double logarithmic transformation is performed on (1), the result is

$$\text{Ln Ln } [1 - F(X)]^{-1} = K \text{ Ln } (X - e) - K \text{ Ln}(V - e) \quad (2)$$

Since this is in linear form, the plot of $\text{Ln}(X - e)$ against $\text{LnLn}[1 - F(X)]^{-1}$ will be a straight line with slope $1/K$ if e is properly selected. In estimating the parameters with sample values by the method of sum of squares, it is necessary to vary the bound " e " until the sum of squares of deviations in the linear regression becomes minimum. The characteristic value V is estimated with the intercept of the vertical axis computed during the regression. In this dissertation, for each value of e , the corresponding parameters K and V were found by using TARSIER program.

B. Analysis

The figures 6.2 and 6.3 show the logarithmic plots of data for estimating Weibull parameters for two problems 6×15 (LS) and $6 \times 10 \times 19$ (LS). Better estimates would be found if we utilized more values of e (instead of three) or used the TARSIER program for all three parameters.

Figures 6.4 and 6.5 show the cumulative sample distribution for the problem 6×15 (LS and NS).

Table 6.1 shows the different parameters with respect to Weibull distributions corresponding to five problems. For each problem the parameters have been estimated for NS and LS and only in two cases for BT4.

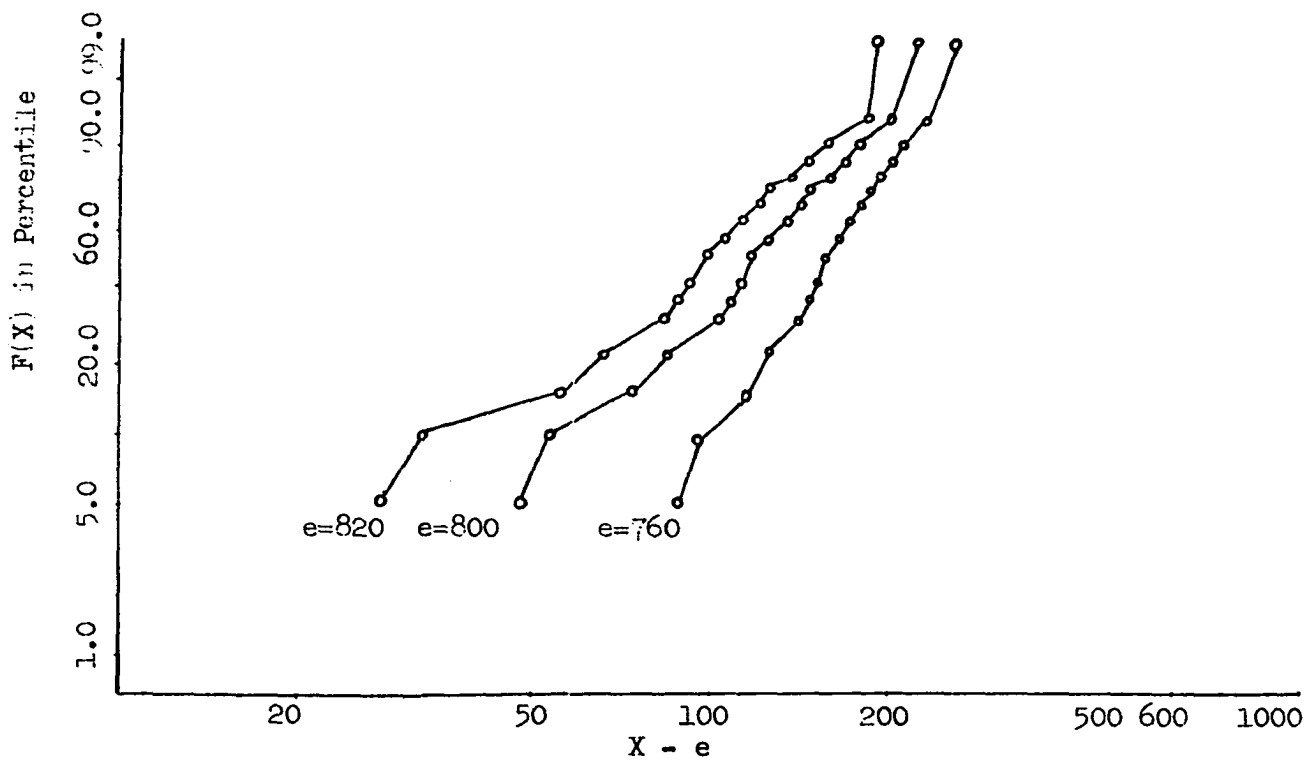


Figure 6.2. Logarithmic plot on Weibull paper (6 x 15, LS)

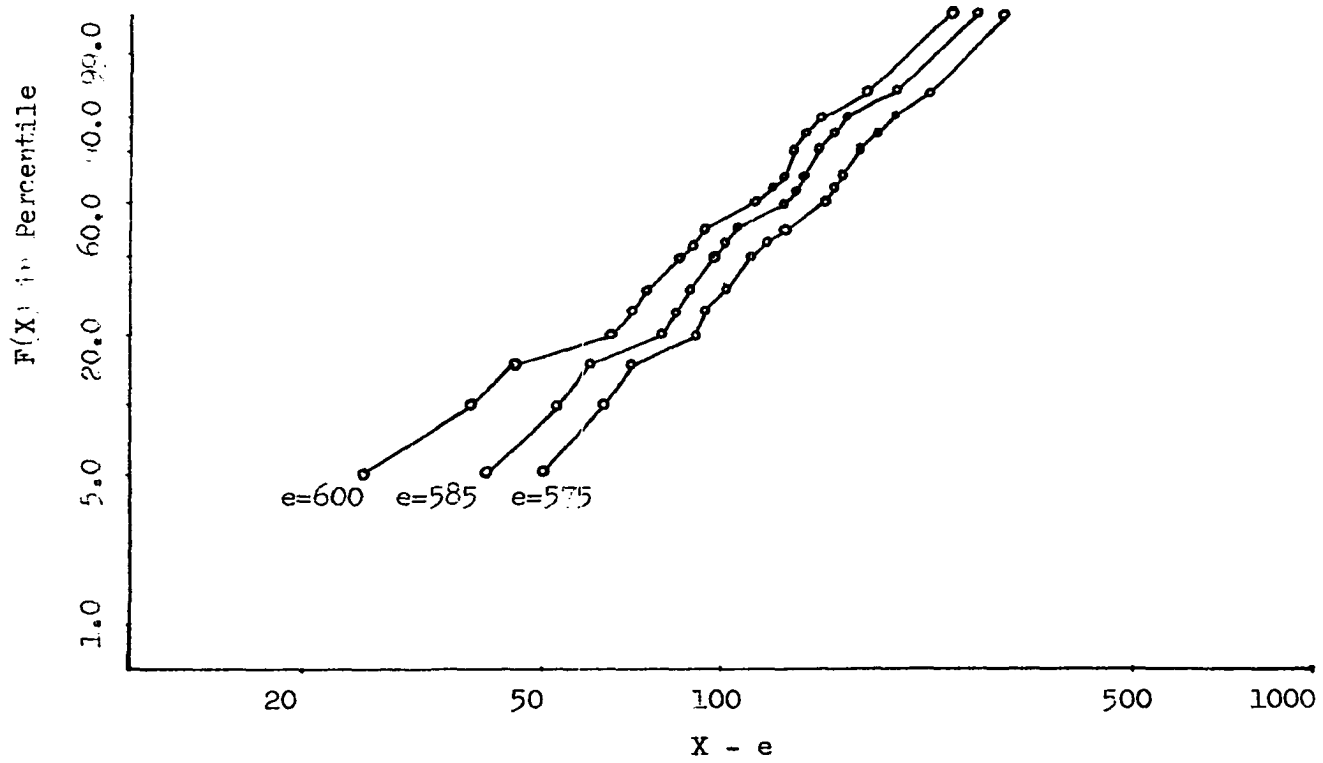


Figure 6.3. Logarithmic plot on Weibull paper (6 x 10 x 19, LS)

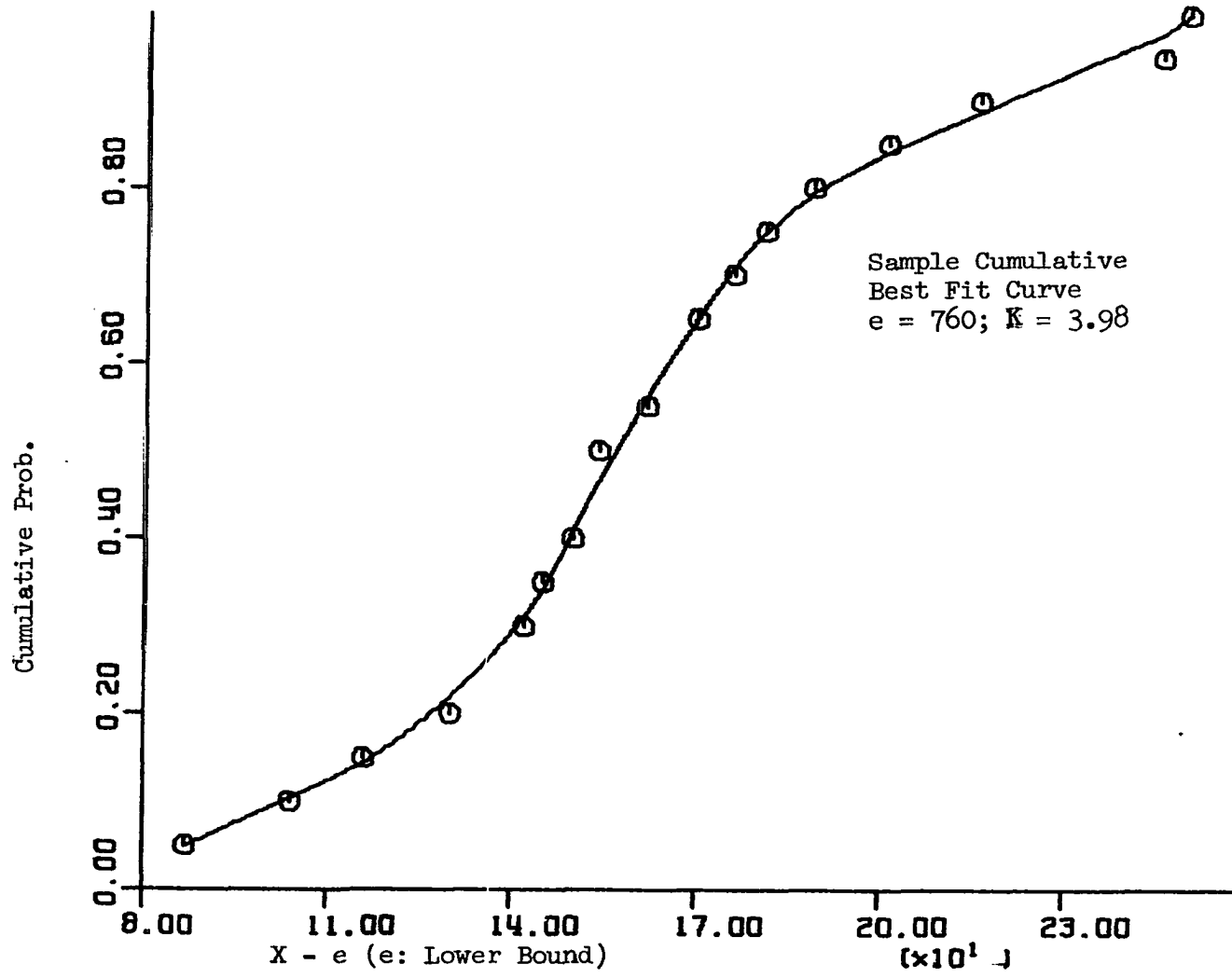


Figure 6.4. Cumulative sample distribution (6 x 15, IS)

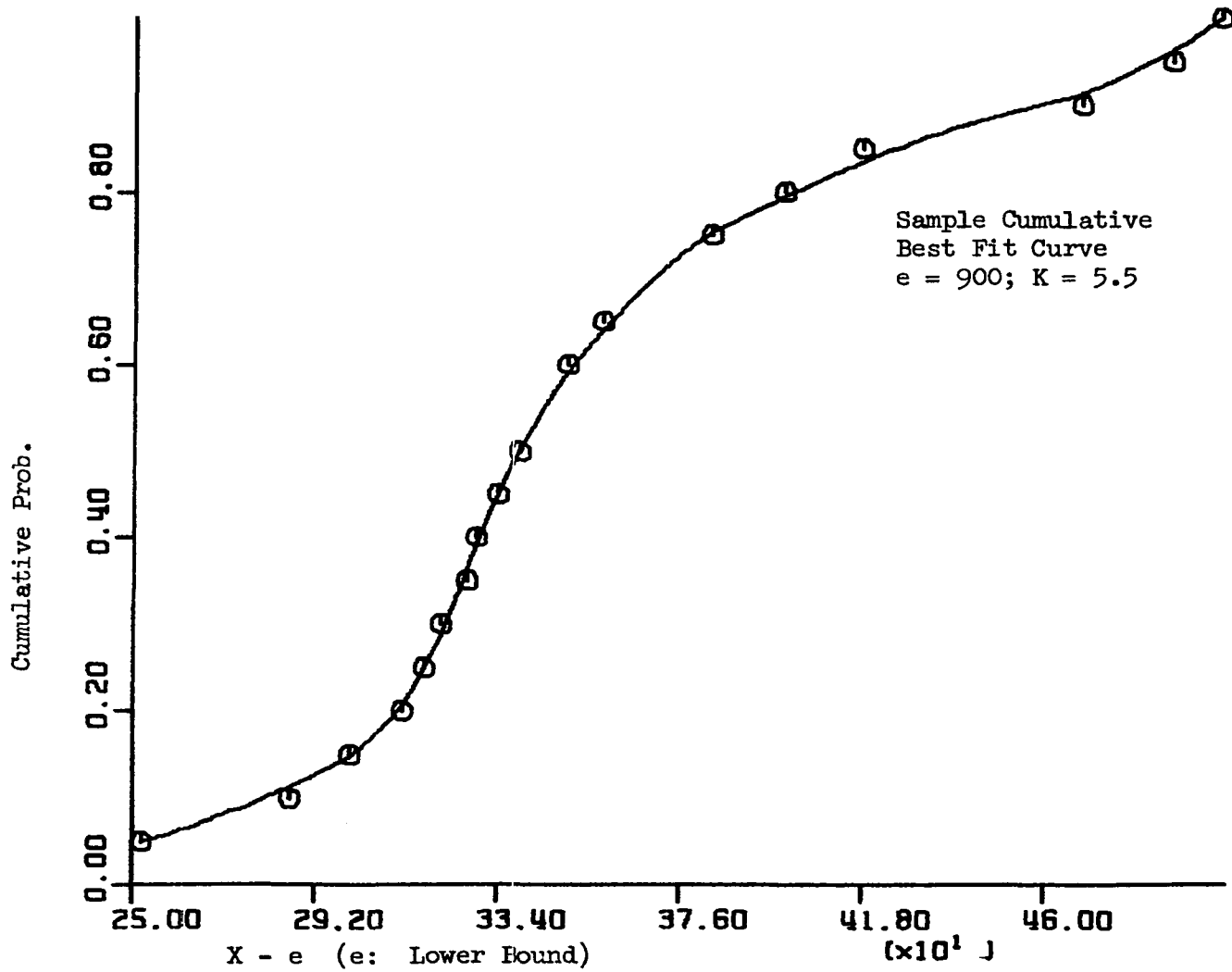


Figure 6.5. Cumulative sample distribution (6 x 15, NS)

Table 6.1. Sample study using the Weibull distribution

Problems	Parameters							
	MIN(X)	LL	e=Xmin	F(X)	K	R	V (for extreme values)	
6x15	NS	1152	750	900	0.09	5.5	294	1280
	LS	847	750	760	0.059	3.98	163	936
15x4	NS	130	126	118	0.0086	4.3	37	152
	LS	127	126	-	-	-	-	-
9x4x10	NS	55	36	49	0.006	4.92	21	66
	LS	51	36	43	0.05	4.8	11	58
	BT4	43	36	38	0.045	3.52	10	50
15x4x10	NS	92	38	79	0.015	5.4	39	107
	LS	76	38	71	0.005	3.7	24	92
	BT4	76	38	69	0.054	3.5	13	85
6x10x19	NS	682	507	612	0.007	4.2	400	840
	LS	625	507	575	0.062	2.8	241	708

In Table 6.1 we observe that in all cases we get a better estimate by LS than NS with respect to different parameters. The estimate by BT seems to be the best of all the methods. This is consistent with our intuition. The value of K is largest for NS because this is relatively a poor sampling procedure. The probability of improvement $F(X)$ is smaller in some cases for NS because the distribution has a longer left tail for a higher value of K . In Table 6.1, corresponding to the problem 15×4 , no parameter was estimated for LS because among the schedules drawn for estimation there was one schedule having the schedule time equal to the lower limit. Realistically, estimated lower bound cannot be less than the lower limit (LL). Corresponding to the same problem for NS, the estimated lower bound was found to be below the lower limit. Therefore, the probability of improvement $F(x)$ was found on the basis of lower limit and not the estimated value.

A large K value accompanied by a long left tail is an indication of weakened sampling method. Where K is as large as 5 or 6, the estimated lower bound may lie well below the best one found. If this happens, a search for a better schedule should be carried out. The effect of the slope of the distribution is illustrated in figure 6.6. The variate $(X - e)$ is standardized to give a better comparison in the distributions having different characteristic values.

If more samples are taken as extreme values, the shape and location parameters can be better estimated. But because of the computer time used in generating samples, a decision must be made to obtain or not obtain a better estimation at the expense of an increase in computation time. Let

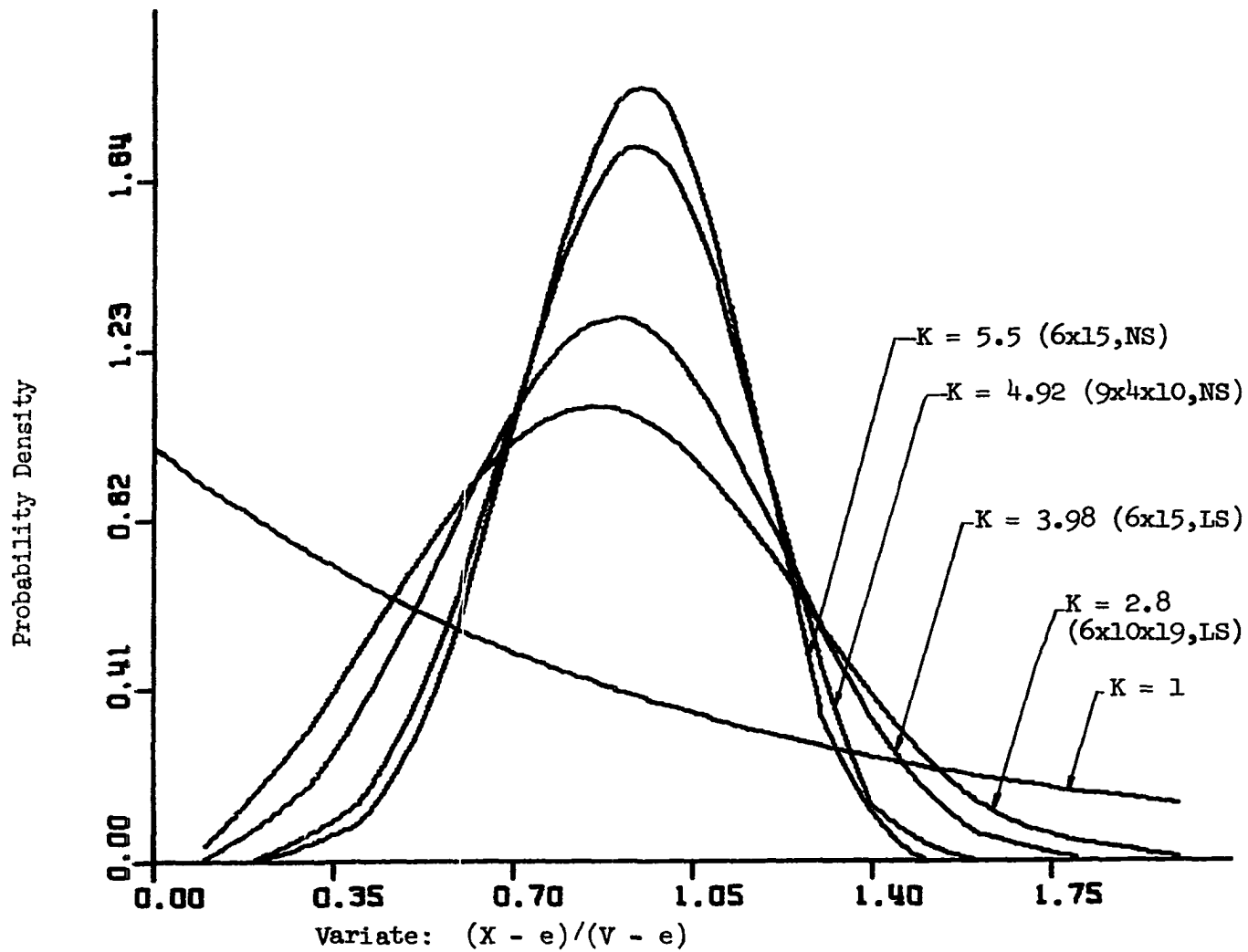


Figure 6.6. Effect of shape parameter on "skewness" of sample distributions

N be the total number of samples drawn and n the sample size in which an extreme point is selected. As we increase the value of n , we should expect a better estimate. But if $\frac{N}{n}$ decreases beyond a certain value, the number of extreme points may be so small that a reasonable analysis is not possible. For a fixed value of n if we increase the value of N , dispersion of the extreme values will be more and estimation may thus be affected. It seems that there should be a particular value of $\frac{N}{n}$ for which a better estimation of the parameters is ensured. However, very small values of N will give a poor estimate.

In Table 6.2, the values of the different parameters have been calculated for various values of N and n with respect to a 5×4 problem. The numbers correspond to a particular value of e . We see that as we increase the value of n , we get a better estimate with regard to the skewness of the distribution. However, from our results we could not come up with any generalized conclusion. Increasing the efficiency of the sampling procedure to get a good estimate by fixing an "optimum" $\frac{N}{n}$ is one of the critical and important aspects of the study of Weibull distribution. With different values of $\frac{N}{n}$, if the TARSIER program is used for different problems in the estimation of the Weibull parameters, some idea may be obtained regarding a good value of $\frac{N}{n}$.

If a good estimate is obtained, the scheduler may use this as a basis for finding the bound B_d to be used in stopping rule 5 discussed in Chapter 3.

Table 6.2. Parameters for different number of samples (N) and sample size (n)

n	N	200	400	500	600
5		V = 29	V = 29	V = 29	V = 29
		K = 3.53	K = 3.01	K = 2.6	K = 3.34
		F(X) = .02048	F(X) = .03597	F(X) = .05586	F(X) = .02517
		SS = 0.1764	SS = .2618	SS = .2021	SS = .17
10		V = 27	V = 27	V = 27	V = 27
		K = 3.1	K = 3.00	K = 3.00	K = 3.19
		F(X) = .11008	F(X) = .1175	F(X) = .1175	F(X) = .10379
		SS = .22	SS = .153	SS = .143	SS = .21
20		V = 26	V = 26	V = 26	V = 26
		K = 2.8	K = 2.75	K = 2.75	K = 3.18
		F(X) = .27481	F(X) = .27957	F(X) = .27957	F(X) = .24076
		SS = .247	SS = .25	SS = .253	SS = .284
25			V = 26	V = 26	V = 26
			K = 3.92	K = 3.85	K = 3.95
			F(X) = .18457	F(X) = .18935	F(X) = .18256
			SS = .104	SS = 0.11	SS = 0.12

VII. TRUCK ROUTING ALGORITHM

A. Introduction

The classical transportation model represents shipping schedules that minimize the cost of shipping products from origins to destinations. Due to the very nature of its formulation, this model when applied to the trucking industry faces some basic problems. In particular, various types of information such as how many loads to be taken each day, how to return from each delivery, how to minimize the cost of the trips when the truck is travelling empty are usually not included in the basic transportation model. This necessitates some modification of the model before it can be a useful tool for a trucking firm. The purpose of the modification is to minimize the total travelling time of a group of trucks to satisfy a given set of demands and to determine the "exact" route each truck should follow on each day.

The basic situation to which we will apply a modified form of the scheduling algorithm, explained in Chapter II, is one where a trucking company needs to ship one bulk product such as oil, gasoline, or cement from m different origins or warehouses to n destinations. Our objective is to produce a schedule which will give the "exact" route each truck is to follow so that the total travelling time of all the trucks is minimized. Each route will begin by leaving the trucking terminal to go to some origin and end by returning to the terminal from some destination.

B. Mathematical Model

Kammin (1976) developed a mathematical model of the above situation for arriving at an optimal solution. Before we discuss a short version of this mathematical model, let us look into the following basic assumptions under which it is constructed (Kammin, 1976):

(i) The trucking firm has one terminal to maintain and refuel its trucks. Each truck must start from this terminal in the morning and return to this terminal at the end of the day.

(ii) Only one type of product is shipped. Therefore, a truck is ready to be reloaded as soon as it is emptied.

(iii) The origins and destinations are close enough to each other so that more than one delivery can be made in one day.

(iv) The trucks will be routed so that each complete route can be accomplished in one shift of eight to twelve hours.

(v) The total number of loads of the product available at the origins is equal to the total number of loads required at the destinations.

(vi) Minimizing the time required to satisfy a set of demands is equivalent to minimizing the cost of satisfying the demands.

Let us now discuss the objective function and different constraints of the mathematical model for the modified transportation problem.

1. The objective function

$$\text{Min} \quad \sum_{k=1}^r \sum_{j=1}^n \sum_{i=1}^m c_{ij} x_{ijk} + \sum_{j=1}^n \sum_{i=1}^m d_{ij} y_{ijk} + \sum_{i=1}^m c_i w_{ik} + \sum_{j=1}^n d_j v_{jk} \quad (1)$$

where

x_{ijk} = the number of loads delivered from origin i to destination j on the k th day,

c_{ij} = the time it takes to deliver one load of product from origin i to destination j ,

y_{ijk} = the number of empty trips from destination j to origin i on the k th day,

d_{ij} = the time it takes to travel from destination j to origin i ,

w_{ik} = the number of trips from the terminal to origin i on the k th day,

c_i = the time it takes to travel from the terminal to origin i ,

v_{jk} = the number of trips from destination j back to the terminal on the k th day,

d_j = the time it takes to travel from destination j back to the terminal.

2. Supply and demand constraints

$$\sum_{k=1}^r \sum_{j=1}^n x_{ijk} = a_i, \quad i=1, \dots, m, \quad (2)$$

$$\sum_{k=1}^r \sum_{i=1}^m x_{ijk} = b_j, \quad j=1, \dots, n, \quad (3)$$

where:

a_i = the number of loads available at origin i ,

b_j = the number of loads required at destination j .

3. Routing constraints

It is necessary that the solution for one day, i.e., one specific k , should be a route that can be actually driven by a truck. The truck must start at the terminal, alternate between origins and destinations, and return to the terminal from some destination at the end of the day. A set of constraints from (4) to (7) insures that these requirements are met.

$$\sum_{i=1}^m w_{ik} = 1, \quad k = 1, \dots, r \quad (4)$$

$$\sum_{j=1}^n v_{jk} = 1, \quad k = 1, \dots, r \quad (5)$$

$$\sum_{j=1}^n x_{ijk} = \sum_{j=1}^n y_{ijk} + w_{ik} \quad \begin{array}{l} i=1, \dots, m \\ k=1, \dots, r \end{array} \quad (6)$$

$$\sum_{i=1}^m i_{jk} = \sum_{i=1}^m y_{ijk} + v_{jk} \quad \begin{array}{l} j=1, \dots, n \\ k=1, \dots, r \end{array} \quad (7)$$

The sets of constraints (6) and (7) specify that for any given day and one specific origin or destination, the number of arrivals equals

the number of departures of the truck.

4. Balance constraints

It is considered in the model that the hours travelled per day are limited and the time driven each day is kept fairly consistent by the following constraints:

$$\sum_{j=1}^n \sum_{i=1}^m x_{ijk} \leq \left[\frac{\sum_{i=1}^m a_i}{r} \right] + 1, \quad k=1, \dots, r \quad (8)$$

5. Subtour constraints

One more type of constraint is required for the above mathematical model to produce a workable solution. A subtour is a complete loop in a route that is not connected to the rest of the route. The set of constraints (6) and (7) can also be satisfied by such a loop. So these subtours must be prevented since a truck cannot follow a route that contains a subtour. Instead of discussing the different subtour constraints of the model in this dissertation, we simply refer to Kammin (1976).

C. Motivation for Monte Carlo

Again referring to Kammin (1976), the following three problems provide us some information regarding the number of constraints and variables for the above mathematical model.

<u>Problem 1.</u>	<u>Problem 2.</u>	<u>Problem 3.</u>
(2 x 2 x 10 x 20)	(3 x 3 x 10 x 20)	(4 x 3 x 10 x 20)
2 origins	3 origins	4 origins
2 destinations	3 destinations	3 destinations
20 loads	20 loads	20 loads
10 trucks	10 trucks	10 trucks

Table 7.1 shows how the number of variables and constraints increase with problem size. In fact, these numbers depend on the number of origins, number of destinations and number of trucks. With a slight increase in the problem size, subtour constraints increase at a very high rate. Though the mathematical model would give an optimal solution, depending on the problem size, the number of variables and constraints may be so high that it will not be economical with respect to the resources needed for computer time. This aspect of the mathematical model motivated the use of the Monte Carlo technique which can generate many schedules within a very short time. It is expected that this technique will provide us a near optimal solution which can justify its worth against an optimal solution which requires much more computer time. Furthermore, the Monte Carlo technique, as we shall see later, is more flexible and easier to apply under different situations. In the rest of this chapter, we will concentrate on developing a Monte Carlo algorithm which will be suitable for a trucking industry as mentioned earlier and illustrate the algorithm with different examples.

Table 7.1. Number of variables and constraints

Problem	Parameters of the model	Subtour excluded	Due to subtour	Total
1	Variables	120	40	160
(2 x 2 x 10 x 20)	Constraints	65	80	145
2	Variables	240	90	330
(3 x 3 x 10 x 20)	Constraints	96	360	456
3	Variables	280	120	400
(4 x 3 x 10 x 20)	Constraints	107	720	827

D. Development of Algorithm

Let us consider a simple problem having two origins O1 and O2, two destinations D1 and D2, one terminal T, two trucks and three loads. Let us also assume that truck 1 will carry 1 load and truck 2 will carry the other two. The following data matrix gives the other relevant information.

Table 7.2. Data matrix for a sample problem (2 x 2 x 2 x 3)

	T	O1	O2	D1	D2	a_i^*
T		3	4	3	6	
O1	(Time/cost matrix)			2	6	2
O2				7	4	1
** b_j				1	2	3

* a_i = number of loads available at origin i , ($i = 1,2$)

** b_j = number of loads required at destination j , ($j = 1,2$)

The purpose of the algorithm will be to define a route for each truck so that the total travelling time becomes "minimum" under the above limitations. Each route starts at the terminal (T) and ends at the terminal.

Before we attempt to modify the algorithm of Chapter 2 to solve the above transportation problem, we have to define the problem in the context

of a multi-machines multiple facilities system. Here, origin refers to a facility in which two origins O_1 and O_2 refer to the two machines. Similar interpretation holds for destination and terminal. For the sake of illustration of the algorithm, origins, destinations and terminal will refer to facilities 1, 2, and 3, respectively. Trucks refer to the respective jobs to be processed. The travelling time for a truck from a previous facility refers to the processing time with respect to the present facility. As an example, if truck 1 takes two hours to travel from the terminal to origin 1, then in terms of the scheduling algorithm, it will be stated as follows: processing time for job 1 in machine 1 of facility 1 is two hours.

In the transportation problem, more than one truck can travel simultaneously from one facility to another. In a scheduling algorithm, it means that more than one job can be simultaneously processed in a single machine. This aspect along with the different limitations in the transportation problem discussed at the beginning of this section necessitates the modifications of the algorithm developed in Chapter 2.

At this stage, the notation that will be used in the algorithm to specify different operations can be better explained if we define the transportation problem as a linear graph. Referring to the problem stated earlier and considering for the time being infinite capacity and demand at each origin and destination respectively, we can define the route of each truck independent of the other by two linear graphs as shown in figures 7.1 and 7.2.

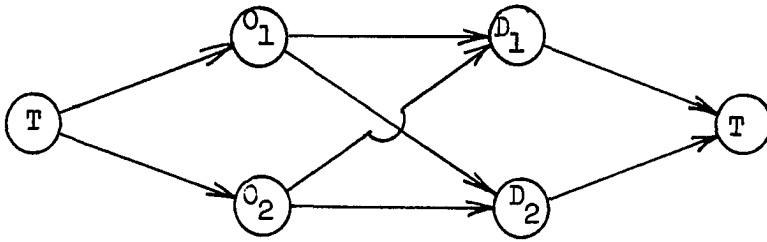


Figure 7.1. Possible routes for truck 1

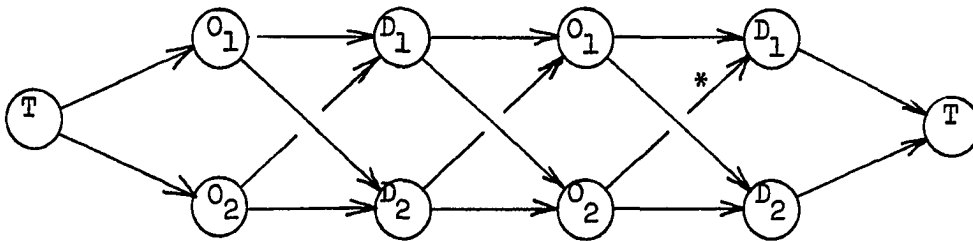


Figure 7.2. Possible routes for truck 2

Pictorial representations of the precedence relations of technological and scheduling orderings have been completely defined by the network diagrams. We will denote each node of these linear graphs by 5 integers $(ijk\ell m)$, where

- i : facility (origin, destination or terminal)
- j : machine within the facility (O1 or O2 if the facility is the origin, etc.)
- k : job to be processed (whether truck 1 or truck 2)
- ℓ : ℓ th time job k is in facility i (highest value of ℓ is the number of loads a truck can carry)
- m : machine within the previous facility the truck is coming from

Thus, for example, in figure 7.2, let us refer to the operation with an asterisk (*) on it. Here

Facility: destination $\Rightarrow i = 2$

Machine: destination 1 $\Rightarrow j = 1$

Job: truck 2 $\Rightarrow k = 2$

Second time truck 2 is in facility 2 (destination), so $l = 2$

Machine of the previous facility the truck is coming from is 02, so $m = 2$

So the operation will be denoted by (21222). It is to be noted that each operation in the algorithm developed in Chapter 2 was denoted by 4 integers instead of 5.

The algorithm to be developed here differs from that in Chapter 2 not only in the logics, but also in structure. Here we have to add one more column CLO, the figures therein indicate the number of loads available at each origin or required at each destination. At the beginning of each schedule the entries in CLO should be initialized. The other columns have the identical meaning as that in Chapter 2. It should be noted that the concept of left shifting is meaningless in a transportation problem because more than one truck can come simultaneously to any facility and the operation will always start at the max job time.

Let us now discuss the different steps needed in the computer program for generating a feasible route for each truck. As before, this program can handle 500 operations and 1500 schedules.

Step 1. Resolving or initializing: Resolve the data and make the proper entry to CLO and to each of the arrays from IC1 to IC7. If any schedule is already found, reinitialize all the entries.

Step 2. Randomization in selecting a process: Check IC3 to see whether there is any zero entry in any row. If no zero entry is found, it implies that a feasible route has been obtained for each truck and step 6 follows. Otherwise, count the number of zeroes and select one of the processes at random. IFIND is the row selected by randomization.

Step 3. Processing the operation: Process the operation at the present maximum job time. Put this value in IC6. To this value, add the entry in IC4 and put it in IC7. Find the maximum entry (MAXM = IMAX) in IC5 corresponding to the machine. Make $MAXM = MAXM + 1$ and put it in column 5. If MAXM is not equal to the entry in CLO, turn the switches from 0 (or 1) to -1 corresponding to IFIND and its counterparts and then proceed to step 5. Otherwise, go to step 4.

Step 4. Blocking the machine: Multiply the entry in CLO by -1 and turn all the switches in IC3 corresponding to the machine to -1. It means no more new entries are possible in any column for this machine.

Step 5. Operations to follow: Check IC2. If there is no entry corresponding to IFIND, go to step 2. Otherwise, corresponding to each operation, check the entry CLO. If it is negative, do nothing and go to step 2. Otherwise, turn the switch in IC3 from 1 to 0 for the corresponding operation and go to step 2.

Step 6. Schedule time: Add all the entries in IC7 corresponding to the terminal. This is the total travelling time C7 for all the trucks. If C7 is less than or equal to the best previous C7, write out this tableau.

Step 7. Stopping rule: Update NPROB (number of feasible solutions) by $NPROB = NPROB + 1$. If NPROB is not a multiple of the sample number specified, go to step 1. Otherwise, call the desired stopping rule and check the criterion. If it is not met, go to step 1. Otherwise, proceed to the next step.

Step 8. Printout: Print all the travelling times found so far in descending order. The program is terminated at this point.

Before we conclude this section, let us summarize some of the differences between the algorithm developed here and the one in Chapter 2.

(i) Each operation in this section is denoted by five integers, while in Chapter 2, it is denoted by four integers.

(ii) In order to keep track of the availabilities of the origins and requirements at destinations, an extra column CLO is needed in transportation algorithm.

(iii) In Chapter 2, the operation is processed either according to the left-shifting principle or at max (maximum job time and maximum machine time). But in the transportation algorithm, the operation is always processed at the maximum job time.

(iv) Instead of selecting one operation at random in CL2 in Chapter 2, in the transportation algorithm all the operations are taken into consideration, checked with the respective entries in CLO and adjusted accordingly. If one operation is selected at random, then some routes may not be completed at all.

(v) In step 6 of the algorithm in Chapter 2, the highest entry in IC7 is the schedule time, while the sum of the entries in IC7 corresponding to terminal is the total travelling time of all the trucks.

E. Iteration Procedure

Referring to the discussion in the previous section, let us now proceed to apply the algorithm to the problem stated at the beginning of section D.

In Table 7.3, the problem has been defined completely in algorithmic notations, showing the technological orderings between the operations. The processing times have been entered in column 4.

To start with, we see that certain operations are scheduleable; i.e., there are 0 entries in CL3 corresponding to these operations.

Iteration 1: From the scheduleable operations 11111, 11211, 12111, 12211, let us randomly select 11111. Truck 1 was at the terminal and so the maximum job time $MJM = 0$. So, starting time is 0; the completion time is $0 + C_4 = 0 + 3 = 3$. The Index corresponding to IFIND in IC5 is increased by 1. So $MAXMJ = 1$.

The entry in CLO corresponding to the machine (origin 1) is 2. So, $MAXMJ \neq C_0$ (entry in CLO) which implies we turn the switches in IC3 corresponding only to 11111 and its counterpart 12111 from 0 to -1.

In column 2 corresponding to IFIND, there are two operations, 21111, 22111. The entries in CLO for destinations 1 and 2 are positive and so the switches in IC3 corresponding to 21111 and 22111

Table 7.3. Iteration 0

CL0	CL1	CL2	CL3	CL4	CL5	CL6	CL7	
2	11111	→ Origin 1, Truck 1	21111, 22111	0	3	0	0	0
	11211	→ Origin 1, Truck 2	21211, 22211	0	3	0	0	0
	11221		21221, 22221	1	2	0	0	0
	11222		21221, 22221	1	6	0	0	0
1	12111	→ Origin 2, Truck 1	21112, 22112	0	4	0	0	0
	12211	→ Origin 2, Truck 2	21212, 22212	0	4	0	0	0
	12221		21222, 22222	1	7	0	0	0
	12222		21222, 22222	1	4	0	0	0
1	21111	→ Destination 1, Truck 1	31111	1	2	0	0	0
	21112		31111	1	7	0	0	0
	21211	→ Destination 1, Truck 2	11221, 12221	1	2	0	0	0
	21212		11221, 12221	1	7	0	0	0
	21221		31211	1	2	0	0	0
	21222		31211	1	7	0	0	0
2	22111	→ Destination 2, Truck 1	31112	1	6	0	0	0
	22112		31112	1	4	0	0	0
	22211	→ Destination 2, Truck 2	11222, 12222	1	6	0	0	0
	22212		11222, 12222	1	4	0	0	0
	22221		31212	1	6	0	0	0
	22222		31212	1	4	0	0	0
99	31111	Terminal, Truck 1		1	3	0	0	0
	31112			1	6	0	0	0
	31211	Terminal, Truck 2		1	3	0	0	0
	31212			1	6	0	0	0

are reset from 1 to 0, showing that the operations are now schedule-able. All these changes are shown in Table 7.4

Iteration 2: Now we have the possibility of scheduling one of the operations 11211, 12211, 21111 and 22111. Let us suppose we randomly select 21111. Truck 1 has already travelled 3 units of time, so $MJM = 3$ which means the starting time for this operation is $C_6 = 3$ and the completion time $C_7 = C_6 + C_4 = 3 + 2 = 5$. $MAXMJ = MAXMJ + 1 = 0 + 1 = 1$ which is equal to the entry in CLO for destination 1. So, corresponding to the destination 1, multiply the entry in CLO by -1 and turn all the switches in IC3 to -1. This means no more new entries are possible in any of the columns for destination 1. Turn also the switch in IC3 from 0 to -1 for 22111, the counterpart of 21111. In column 2 corresponding to IFIND is 31111. The entry in CLO for terminal is positive so the switch in IC3 corresponding to 31111 is reset from 1 to 0, showing that it is scheduleable.

Table 7.5 reflects these changes.

Iteration 3: Among the scheduleable operations 12211 and 31111, we randomly select 12211. Truck 2 was at the terminal and the maximum job time $MJM = 0$. So the starting time is $C_6 = 20$ and the completion time is $C_7 = C_6 + C_4 = 0 + 4 = 4$. $MAXMJ = MAXMJ + 1 = 0 + 1 = 1$ which is equal to the entry in CLO for origin 2. So corresponding to this origin, multiply the entry in CLO by -1 and turn all the switches in IC3 to -1. Turn also the switch in IC3 from 0 to -1 for 11211, counterpart of 12211.

In column 2 corresponding to IFIND, there are two operations 21212,

Table 7.4. Iteration 1

CL0	CL1	CL2	CL3	CL4	CL5	CL6	CL7
2	11111 → Origin 1, Truck 1	211111, 22111	-1	3	1	0	3
	11211 } → Origin 1, Truck 2	21211, 22211	0	3	0	0	0
	11221 } → Origin 1, Truck 2	21221, 22221	1	2	0	0	0
	11222 } → Origin 1, Truck 2	21221, 22221	1	6	0	0	0
1	12111 → Origin 2, Truck 1	21112, 22112	-1	4	0	0	0
	12211 } → Origin 2, Truck 2	21212, 22212	0	4	0	0	0
	12221 } → Origin 2, Truck 2	21222, 22222	1	7	0	0	0
	12222 } → Origin 2, Truck 2	21222, 22222	1	4	0	0	0
1	21111 } → Destination 1, Truck 1	31111	0	2	0	0	0
	21112 } → Destination 1, Truck 1	31111	1	7	0	0	0
	21211 } → Destination 1, Truck 2	11221, 12221	1	2	0	0	0
	21212 } → Destination 1, Truck 2	11221, 12221	1	7	0	0	0
	21221 } → Destination 1, Truck 2	31211	1	2	0	0	0
	21222 } → Destination 1, Truck 2	31211	1	7	0	0	0
2	22111 } → Destination 2, Truck 1	31112	0	6	0	0	0
	22112 } → Destination 2, Truck 1	31112	1	4	0	0	0
	22211 } → Destination 2, Truck 2	11222, 12222	1	6	0	0	0
	22212 } → Destination 2, Truck 2	11222, 12222	1	4	0	0	0
	22221 } → Destination 2, Truck 2	31212	1	6	0	0	0
	22222 } → Destination 2, Truck 2	31212	1	4	0	0	0
99	31111 } → Terminal, Truck 1		1	3	0	0	0
	31112 } → Terminal, Truck 1		1	6	0	0	0
	31211 } → Terminal, Truck 2		1	6	0	0	0
	31212 } → Terminal, Truck 2		1	6	0	0	0

Table 7.5. Iteration 2

CLO	CL1	CL2	CL3	CL4	CL5	CL6	CL7	
2	11111	→ Origin 1, Truck 1	21111, 22111	-1	3	1	0	3
	11211	→ Origin 1, Truck 2	21211, 22211	0	3	0	0	0
	11221		21221, 22221	1	2	0	0	0
	11222		21221, 22221	1	6	0	0	0
1	12111	→ Origin 2, Truck 1	21112, 22112	-1	4	0	0	0
	12211	→ Origin 2, Truck 2	21212, 22212	0	4	0	0	0
	12221		21222, 22222	1	7	0	0	0
	12222		21222, 22222	1	4	0	0	0
-1	21111	→ Destination 1, Truck 1	31111	-1	2	1	3	5
	21112		31111	-1	7	0	0	0
	21211	→ Destination 1, Truck 2	11221, 12221	-1	2	0	0	0
	21212		11221, 12221	-1	7	0	0	0
	21221		31211	-1	2	0	0	0
	21222		31211	-1	7	0	0	0
2	22111	→ Destination 2, Truck 1	31112	-1	6	0	0	0
	22112		31112	1	4	0	0	0
	22211	→ Destination 2, Truck 2	11222, 12222	1	6	0	0	0
	22212		11222, 12222	1	4	0	0	0
	22221		31212	1	6	0	0	0
	22222		31212	1	4	0	0	0
99	31111	→ Terminal, Truck 1		0	3	0	0	0
	31112			1	6	0	0	0
	31211	→ Terminal, Truck 2		1	3	0	0	0
	31212			1	6	0	0	0

22212. The entry in CLO for destination 1 is negative and that for destination 2 is positive. This means we cannot do anything with operation 21212 and we simply turn the switch in IC3 corresponding to 22212 from 1 to 0, showing that it is schedulable. This iteration is shown in Table 7.6.

Iteration 4: Among the two schedulable operations 22212 and 31111, let us randomly select 22212. Truck 2 has already travelled 4 units of time. So $MJM = 4$ which means the starting time of the operation is $C_6 = 4$ and the completion time $C_7 = C_6 + C_4 = 4 + 4 = 8$.

$$MAXMJ = MAXMJ + 1 = 1$$

The entry in CLO corresponding to this machine (or destination 2) is 2. So $MAXMJ \neq C_0$ (entry in CLO) which implies the switches in IC3 corresponding only to 22212 and its counterpart 21212 should be made -1.

In column 2 corresponding to IFIND, there are two operations, 11222, 12222. The entry in CLO for origin 1 is positive and that for origin 2 is negative. As in iteration 3, we cannot proceed towards the operation 12222 and we simply turn the switch in IC3 corresponding to 11222 from 1 to 0, showing that it is now schedulable. This iteration is shown in Table 7.7.

Iteration 5: Among the schedulable operations 11222 and 31111, let us randomly select 11222. Truck 2 already travelled 8 units of time. So $MJM = 8$ which means the starting time of the operation is $C_6 = 8$ and the completion time $C_7 = C_6 + C_4 = 8 + 6 = 14$. $MAXMJ = MAXMJ + 1 = 1 + 1 = 2$ which is equal to the entry in CLO for origin 1. So,

Table 7.6. Iteration 3

CL0	CL1	CL2	CL3	CL4	CL5	CL6	CL7
2	11111 } → Origin 1, Truck 1	21111, 22111	-1	3	1	0	3
	11211 } → Origin 1, Truck 2	21211, 22211	-1	3	0	0	0
	11221 } → Origin 1, Truck 2	21221, 22221	1	2	0	0	0
	11222 }	21221, 22221	1	6	0	0	0
-1	12111 } → Origin 2, Truck 1	21112, 22112	-1	4	0	0	0
	12111 } → Origin 2, Truck 2	21212, 22212	-1	4	1	0	4
	12221 } → Origin 2, Truck 2	21222, 22222	-1	7	0	0	0
	12222 }	21222, 22222	-1	4	0	0	0
-1	21111 } Destination, Truck 1	31111	-1	2	1	3	5
	21112 }	31111	-1	7	0	0	0
	21211 } Destination 1, Truck 2	11221, 12221	-1	2	0	0	0
	21212 }	11221, 12221	-1	7	0	0	0
	21221 }	31211	-1	2	0	0	0
	21222 }	31211	-1	7	0	0	0
2	22111 } Destination 2, Truck 1	31112	-1	6	0	0	0
	22112 }	31112	1	4	0	0	0
	22211 } Destination 2, Truck 2	11222, 12222	1	6	0	0	0
	22212 }	11222, 12222	0	4	0	0	0
	22221 }	31212	1	6	0	0	0
	22222 }	31212	1	4	0	0	0
99	31111 } → Terminal, Truck 1		0	3	0	0	0
	31112 }		1	6	0	0	0
	31211 } → Terminal, Truck 2		1	3	0	0	0
	31212 }		1	6	0	0	0

Table 7.7. Iteration 4

CLO	CL1	CL2	CL3	CL4	CL5	CL6	CL7
2	11111 } → Origin 1, Truck 1	21111, 22111	-1	3	1	0	3
	11211 } → Origin 1, Truck 2]1]11,]]]11	-1	3	0	0	0
	11221 } → Origin 1, Truck 2	21221, 22221	1	2	0	0	0
	11222 }	21221, 22221	0	6	0	0	0
-1	12111 } → Origin 2, Truck 1	21112, 22112	-1	4	0	0	0
	12211 } → Origin 2, Truck 2	21212, 22212	-1	4	1	0	4
	12221 } → Origin 2, Truck 2	21222, 22222	-1	7	0	0	0
	12222 }	21222, 22222	-1	4	0	0	0
-1	21111 } → Destination 1,	31111	-1	2	1	3	5
	21112 } Truck 1	31111	-1	7	0	0	0
	21211 } → Destination,	11221, 12221	-1	2	0	0	0
	21212 } Truck 2	11221, 12221	-1	7	0	0	0
	21221 }	31211	-1	2	0	0	0
	21222 }	31211	-1	7	0	0	0
2	22111 } Destination 2,	31112	-1	6	0	0	0
	22112 } → Truck 1	31112	1	4	0	0	0
	22211 } → Destination 2,	11222, 12222	1	6	0	0	0
	22212 } Truck 2	11222, 12222	-1	4	1	4	8
	22221 }	31212	1	6	0	0	0
	22222 }	31212	1	4	0	0	0
99	31111 } → Terminal, Truck 1		0	3	0	0	0
	31112 }		1	6	0	0	0
	31211 } → Terminal, Truck 2		1	3	0	0	0
	31212 }		1	6	0	0	0

corresponding to this origin, multiply the entry in CLO by -1 and turn all the switches in IC3 to -1. The switch in IC3 for 12222, counterpart of 11222 is already -1.

In column 2 corresponding to IFIND, there are two operations 21221, 22221. The entry in CLO for destination 1 is negative and that for destination 2 is positive. This means that operation 21221 can never be explored and we simply turn the switch in IC3 corresponding to 22221 from 1 to 0, showing that it is scheduleable. This iteration is shown in Table 7.8.

Iteration 6: Among the two scheduleable operations 22221 and 31111, we randomly select 31111. Truck 1 has already travelled 5 units of time. So $MJM = 5$ which means $C_6 = 5$ and $C_7 = C_6 + C_4 = 5 + 3 = 8$.

$MAXMJ = MAXMJ + 1 = 0 + 1 = 1$ which is not equal to the entry in CLO for the terminal which implies we turn the switch in IC3 corresponding only to 31111. This iteration ends here because there is no entry in column 2. Table 7.9 displays this iteration.

Iteration 7: We select the only remaining scheduleable operation 22221. Truck 2 has already travelled 14 units of time. So $MJM = 14$, which means $C_6 = 14$ and $C_7 = C_6 + C_4 = 14 + 6 = 20$.

$MAXMJ = MAXMJ + 1 = 1 + 1 = 2$ which is equal to the entry in CLO for destination 2. So corresponding to this destination, multiply the entry in CLO by -1 and turn all the switches in IC3 to -1. The switch in IC3 for 21221, the counterpart of 22221 is already -1.

In column 2 corresponding to IFIND is 31212. The entry in CLO

Table 7.8. Iteration 5

CL0	CL1	CL2	CL3	CL4	CL5	CL6	CL7
-2	11111 } → Origin 1, Truck 1	21111, 22111	-1	3	1	0	3
	11211 } → Origin 1, Truck 2	21211, 22211	-1	3	0	0	0
	11221 } → Origin 1, Truck 2	21221, 22221	-1	2	0	0	0
	11222 }	21221, 22221	-1	6	2	8	14
-1	12111 } → Origin 2, Truck 1	21112, 22112	-1	4	0	0	0
	12211 } → Origin 2, Truck 2	21212, 22212	-1	4	1	0	4
	12221 } → Origin 2, Truck 2	21222, 22222	-1	7	0	0	0
	12222 }	21222, 22222	-1	4	0	0	0
-1	21111 } → Destination 1,	31111	-1	2	1	3	5
	21112 } Truck 1	31111	-1	7	0	0	0
	21211 } → Destination 1,	11221, 12221	-1	2	0	0	0
	21212 } → Truck 2	11221, 12221	-1	7	0	0	0
	21221 } → Truck 2	31211	-1	2	0	0	0
	21222 }	31211	-1	7	0	0	0
2	22111 } → Destination 2,	31112	-1	6	0	0	0
	22112 } Truck 1	31112	1	4	0	0	0
	22211 } → Destination 2,	11222, 12222	1	6	0	0	0
	22212 } → Truck 2	11222, 12222	-1	4	1	4	8
	22221 } → Truck 2	31212	0	6	0	0	0
	22222 }	31212	1	4	0	0	0
99	31111 } → Terminal, Truck 1		0	3	0	0	0
	31112 }		1	6	0	0	0
	31211 } → Terminal, Truck 2		1	3	0	0	0
	31212 }		1	6	0	0	0

Table 7.9. Iteration 6

CLO	CL1	CL2	CL3	CL4	CL5	CL6	CL7	
-2	11111	→ Origin 1, Truck 1	21111, 22111	-1	3	1	0	3
	11211	→ Origin 1, Truck 2	21211, 22211	-1	3	0	0	0
	11221		21221, 22221	-1	2	0	0	0
	11222		21221, 22221	-1	6	2		14
-1	12111	→ Origin 2, Truck 1	21112, 22112	-1	4	0	0	0
	12211	→ Origin 2, Truck 2	21212, 22212	-1	4	1	0	4
	12221		21222, 22222	-1	7	0	0	0
	12222		21222	-1	4	0	0	0
-1	21111	→ Destination 1, Truck 1	31111	-1	2	1	3	5
	21112		31111	-1	7	0	0	0
	21211	→ Destination 1, Truck 2	11221, 12221	-1	2	0	0	0
	21212		11221, 12221	-1	7	0	0	0
	21221		31211	-1	2	0	0	0
	21222		31211	1	7	0	0	0
2	22111	→ Destination 2, Truck 1	31112	-1	6	0	0	0
	22112		31112	1	4	0	0	0
	22211	→ Destination 2, Truck 2	11222, 12222	1	6	0	0	0
	22212		11222, 12222	-1	4	1		8
	22221		31212	0	6	0	0	0
	22222		31212	1	4	0	0	0
99	31111	→ Terminal, Truck 1		-1	3	1	5	8
	31112			1	6	0	0	0
	31211	→ Terminal, Truck 2		1	3	0	0	0
	31212			1	6	0	0	0

for the terminal is positive so the switch in IC3 corresponding to 31212 is reset from 1 to 0, showing that it is scheduleable.

This iteration is shown in Table 7.10.

Iteration 8: The only scheduleable operation now is 31212. Truck 2 has already travelled 20 units of time. So $MJM = 20$ which means

$$C_6 = 20 \text{ and } C_7 = C_6 + C_4 = 20 + 6 = 26.$$

$MAXMJ = MAXMJ + 1 = 1 + 1 = 2$ which is not equal to the entry in CLO for the terminal which implies that we turn the switch in IC3 corresponding only to 31212. This iteration ends there because there is no entry in column 2. Changes have been shown in Table 7.11.

At this point we check IC3, and we find no zero entry which implies that we have obtained a feasible route for each truck.

Check IC7 and add the entries in this array corresponding to the terminal. The figure so obtained is the total travelling time. In this problem this is $26 + 8 = 34$ units of time.

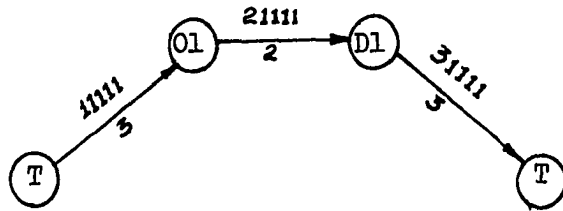
The operations constituting the route travelled by truck 1 are 11111, 21111 and 31111 and those corresponding to the route for truck 2 are 11222, 12211, 22212, 22221 and 31212. The routes for the two trucks are shown in figures 7.3 and 7.4. The number below the arrow indicates the travelling time while the number above the arrow shows the specific operation.

Table 7.10. Iteration 7

CL0	CL1	CL2	CL3	CL4	CL5	CL6	CL7	
-2	11111	→ Origin 1, Truck 1	21111, 22111	-1	3	1	0	3
	11211	→ Origin 1, Truck 2	21211, 22211	-1	3	0	0	0
	11221		21221, 22221	-1	2	0	0	0
	11222		21221, 22221	-1	6	2	8	14
-1	12111	→ Origin 2, Truck 1	21112, 22112	-1	4	0	0	0
	12211	→ Origin 2, Truck 2	21212, 22212	-1	4	1	0	4
	12221		21222, 22222	-1	7	0	0	0
	12222		21222, 22222	-1	4	0	0	0
-1	21111	→ Destination 1, Truck 1	31111	-1	2	1	3	5
	21112		31111	-1	7	0	0	0
	21211	→ Destination 1, Truck 2	11221, 12221	-1	2	0	0	0
	21212		11221, 12221	-1	7	0	0	0
	21221		31211	-1	2	0	0	0
	21222		31211	-1	7	0	0	0
-2	22111	→ Destination 2, Truck 1	31112	-1	6	0	0	0
	22112		31112	-1	4	0	0	0
	22211	→ Destination 2, Truck 2	11222, 12222	-1	6	0	0	0
	22212		-1	4	1	4	8	
	22221		-1	6	2	14	20	
	22222		-1	4	0	0	0	
99	31111	→ Terminal, Truck 1		-1	3	1	4	8
	31112			1	6	0	0	0
	31211	→ Terminal, Truck 2		1	3	0	0	0
	31212		0	6	0	0	0	

Table 7.11. Iteration 8

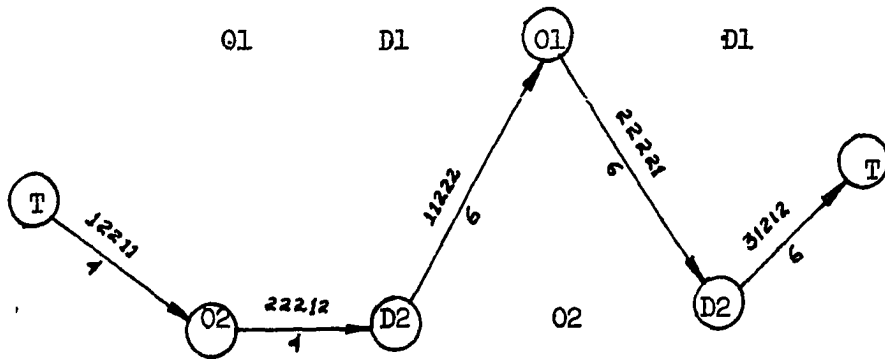
CLO	CL1	CL2	CL3	CL4	CL5	CL6	CL7	
-2	11111	→ Origin 1, Truck 1	21111, 22111	-1	3	1	0	3
	11211	→ Origin 1, Truck 2	21211, 22211	-1	3	0	0	0
	11221		21221, 22221	-1	2	0	0	0
	11222		21221, 22221	-1	6	2	8	14
-1	12111	→ Origin 2, Truck 1	21112, 22112	-1	4	0	0	0
	12211	→ Origin 2, Truc, 2	21212, 22212	-1	4	1	0	4
	12221		12222, 22222	-1	7	0	0	0
	12222		21222, 22222	-1	4	0	0	0
-1	21111	→ Destination 1 Truck 1	31111	-1	2	1	3	5
	21112		31111	-1	7	0	0	0
	21211	→ Destination 1, Truck 2	11221, 12221	-1	2	0	0	0
	21212		11221, 12221	-1	7	0	0	0
	21221		31211	-1	2	0	0	0
	21222		31211	-1	7	0	0	0
-2	22111	→ Destination 2, Truck 1	31112	-1	6	0	0	0
	22112		31112	-1	4	0	0	0
	22211	→ Destination 2 Truck 2	11222, 12222	-1	6	0	0	0
	22212		11222, 12222	-1	4	1	4	8
	22221		31212	-1	6	2	14	20
	22222		31212	-1	4	0	0	0
99	31111	→ Terminal, Truck 1		-1	3	1	5	8
	31112			1	6	0	0	0
	31211	→ Termianl, Truck 2		1	3	0	0	0
	31212			-1	6	2	20	26



O2 D2

Total travel time = 3 + 2 + 3 = 8

Figure 7.3. Route for truck 1



Total travel time = 4 + 4 + 6 + 6 + 6 = 26

Figure 7.4. Route for truck 2

F. Presentation of a Larger Problem

In the previous section, we considered a small problem with 2 origins, 2 destinations, 2 trucks and 3 loads. Let us expand the problem to one having 3 origins, 3 destinations, 10 trucks and 20 loads (3 x 3 x 10 x 20).

Let us assume that truck 1 will carry 4 loads, each of the trucks 7 and 10 will carry 1 load and the rest of the trucks will carry 2 loads each. Other relevant information has been given in the following data matrix.

Table 7.12. Data matrix for the problem (3 x 3 x 10 x 20)

	O_1	O_2	O_3	D_1	D_2	D_3	a_i
T	4	9	4	6	4	3	
O_1	(Time/cost matrix)			2	5	6	8
O_2				8	3	7	6
O_3				6	8	2	6
b_j				5	9	6	20

The problem was run on the computer using the algorithm developed in section D. This problem has 303 possible operations whereas there are 24 operations for the (2 x 2 x 2 x 3) problem explained in the previous section. It shows that for a bigger problem, the input stream

becomes considerably bigger. But this involves only a few changes on each data card.

To halt the sampling process, stopping rule 3 as explained in Chapter 3, section B, was applied. The sample size used was 20 and the total number of schedules generated was 60. CPU time was less than 2 seconds per schedule.

Out of the 60 schedules, the total travelling time for the worst schedule was 307 and that for the best one was 229.

Stopping rule 2 might give us a better possible solution, but at the expense of a higher CPU time. In fact, the best schedule having 229 units of travelling time was obtained in the first 2 samples of size 20. But the stopping rule criterion was not met until a third sample of size 20 was generated. So, for a bigger problem, the sampling strategy should be to specify a smaller sample size and apply stopping rule 2. This may yield a better result in less CPU time.

The routes for each of the 10 trucks corresponding to the schedule having 229 units of travelling time have been displayed in figures 7.5 - 7.14.

The truck routine summary is given in Table 7.13. The information gives an indication of the distribution of the arrivals and departures by each truck at each origin and destination. Although the routes are not known from this table, small changes in the product availability or number of trucks available might permit management to construct another "good" solution by using the table in conjunction with the previous routing diagrams.

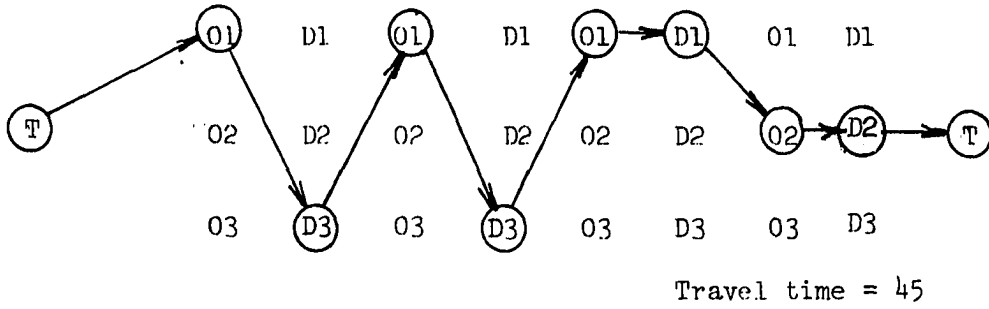


Figure 7.5. Route for truck 1

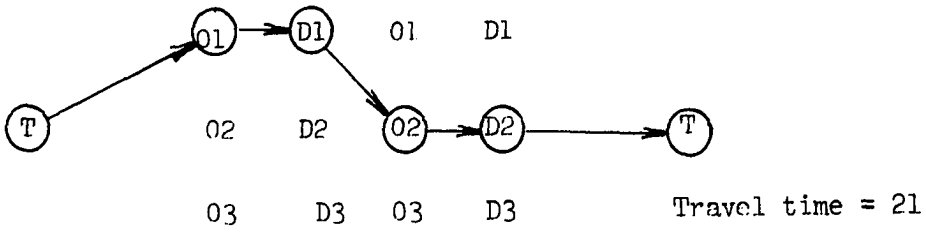


Figure 7.6. Route for truck 2

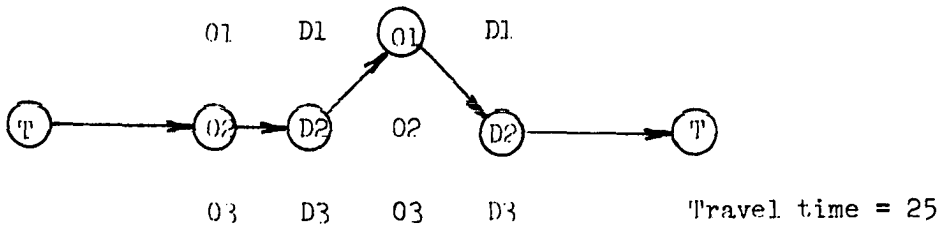


Figure 7.7. Route for truck 3

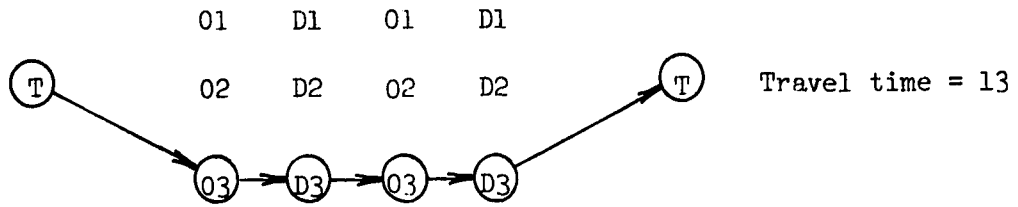


Figure 7.8. Route for truck 4

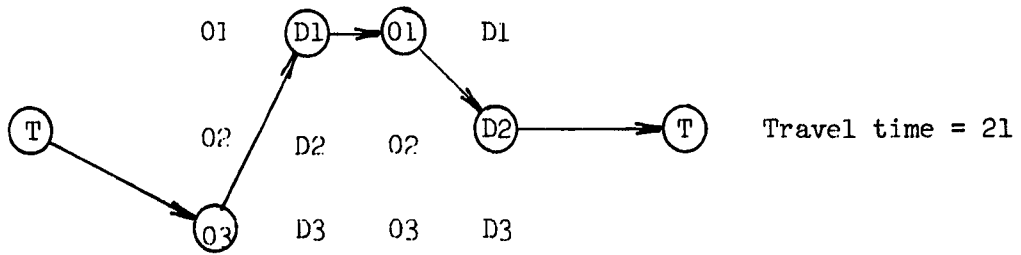


Figure 7.9. Route for truck 5

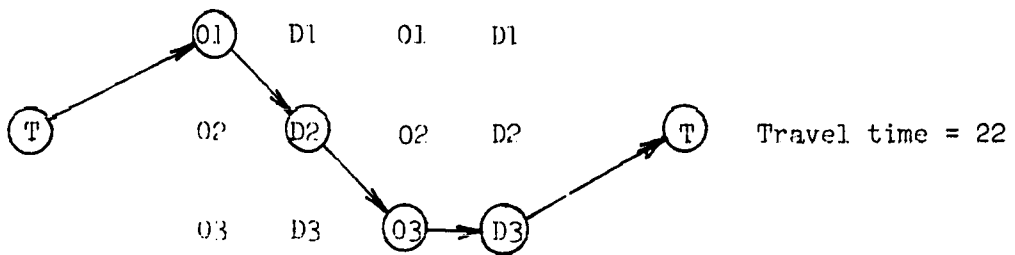


Figure 7.10. Route for truck 6

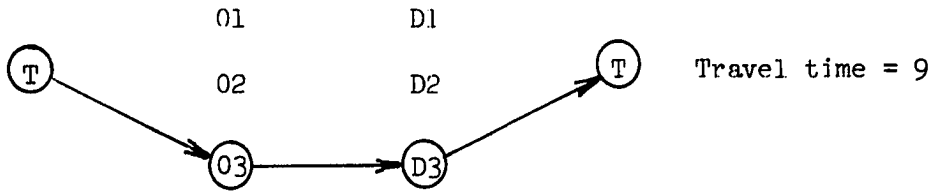


Figure 7.11. Route for truck 7

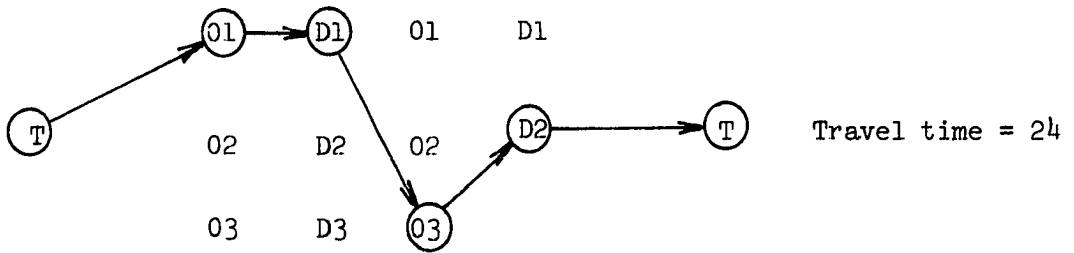


Figure 7.12. Route for truck 8

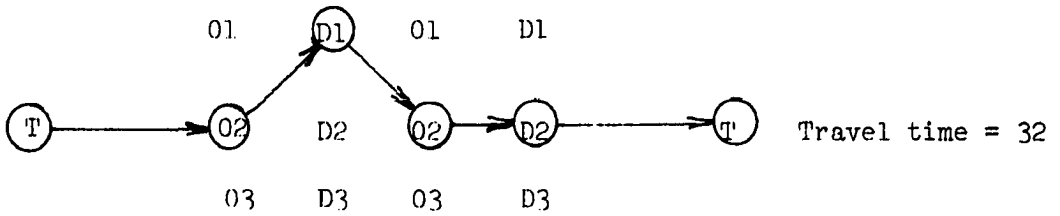
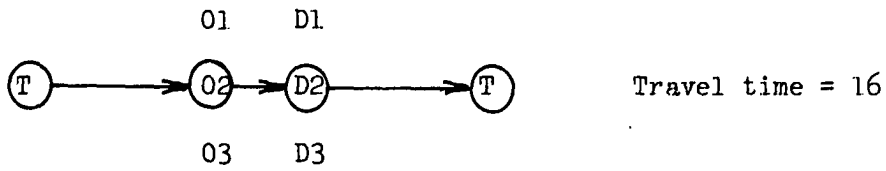


Figure 7.13. Route for truck 9



Total travel time for 10 trucks = 229

Figure 7.14. Route for truck 10

Table 7.13. Truck routing summary

Trucks	O_1	O_2	O_3	D_1	D_2	D_3	L_K^a
1	3	1		1	1	2	4
2	1	1		1	1		2
3	1	1			2		2
4			2			2	2
5	1		1	1	1		2
6	1		1		1	1	2
7			1			1	1
8	1		1	1	1		2
9		2		1	1		2
10		1			1		1

$a_1=8$	$a_2=6$	$a_3=6$	$b_1=5$	$b_2=9$	$b_3=6$	$\sum_{K=1}^{10} L_K^a = 20$
$\begin{pmatrix} 3 \\ \sum_{i=1}^3 a_i = 20 \end{pmatrix}$			$\begin{pmatrix} 3 \\ \sum_{i=1}^3 b_i = 20 \end{pmatrix}$			

L_K^a = number of loads carried by Kth truck

G. Extension of the Transportation Algorithm

1. Multi-terminal system

The algorithm developed in section D is similar to the one given in Chapter 2. Although in sections E and F a single terminal is considered for the illustration of the transportation problem, the algorithm can be applied without any basic modification for a multi-terminal system. The different terminals will be numbered as different machines of the same facility.

2. Multi-product system

This system refers to the situation where more than one type of product is involved. For the sake of discussion, this system can be further subdivided into two classes:

(1) Each truck carries a specific product, and

(2) A truck can carry more than one type of product. Let us discuss each of the above classes by specific examples. For the first class, let us look into a system having three origins, three destinations, one terminal, four trucks and two types of products A and B with the following limitations.

<u>Trucks</u>	<u>Number of loads to carry</u>
1	2(A)
2	3(B)
3	2(A)
4	2(B)

Table 7.14. Supplies and demands

Origins/destinations	Number of loads available/required	
	Product A	Product B
O_1	2	1
O_2	-	2
O_3	2	2
D_1	1	2
D_2	2	-
D_3	1	3

One way to attack the problem would be to add one more integer to the algorithmic notation for an operation to specify the type of product. But this would bring operational complexities in the computer logic. We, therefore, suggest the following alternative approach.

Principle of decomposition: Let us decompose each origin and destination involving more than one type of the product into two origins or destinations. This will give rise to a single-product system of five origins and five destinations as shown in Table 7.15. The system after decomposition is equivalent to two independent systems corresponding to two types of products as shown in Tables 7.16 and 7.17. The two systems with the specific identities to their origins, destinations and trucks can now be solved by a single program with the algorithm developed in section D.

Table 7.15. System after decompositon

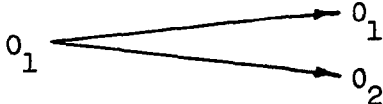


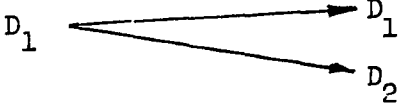
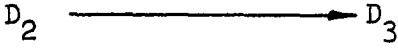

Origins/destinations		Number of loads available/required (System after decomposition)	
Original system	System after decomposition	Product A	Product B
O_1		2	-
		-	1
O_2		-	2
O_3		2	-
		-	2
D_1		1	-
		-	2
D_2		2	-
D_3		1	-
		-	3

Table 7.16. System with product A

Origins	Destinations	Trucks
O_1	D_1	1
O_4	D_3	3
	D_4	

Table 7.17. System with product B

Origins	Destinations	Trucks
O_2	D_2	2
O_3		
O_5	D_5	4

The second class is a multi-product system where trucks can carry more than one type of product. Some modifications are needed in the algorithm of section D in addition to decomposition of origins and destinations.

Let us describe two independent routes of the same truck for two different products by the network diagrams shown in figures 7.15 and 7.16.

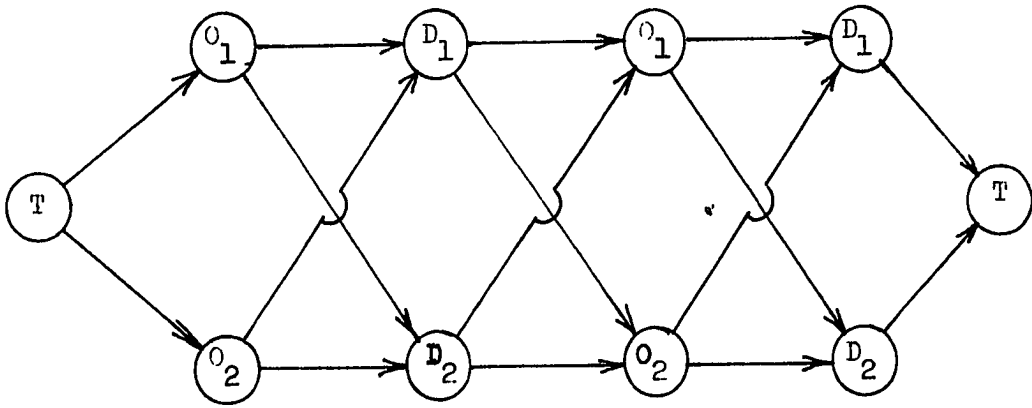


Figure 7.15. Route for product A

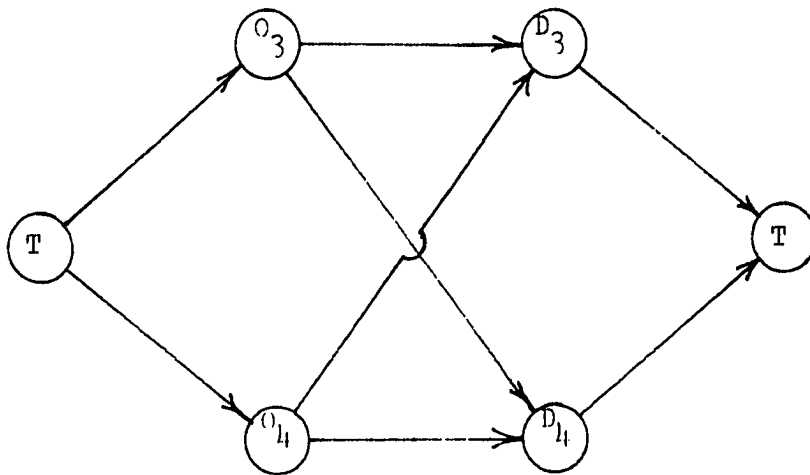


Figure 7.16. Route for product B

The routes of the same truck for the two products being "independent," the truck can reach the terminal without delivering any or some loads of some product, if the algorithm is applied to the problem without any modification. To show this, referring to figures 7.15 and 7.16, let us suppose there are two loads yet to be delivered, one from O_1 to D_1 and the other from O_3 to D_3 . The truck from some destination may come to O_3 , deliver the load to D_3 and go back to the terminal without delivering the load from O_1 to D_1 . The algorithm does not provide anything to prevent this possibility.

In order to ensure that the truck delivers all its assigned loads before it reaches the terminal, let us incorporate the following changes in steps 2 and 3 of the algorithm in section D.

Change the title of the step 3 from "Processing the operation" to "Processing the operation related to origin/destination."

Step 2 will be changed as follows:

Check IC3 except the last group (terminal) whether there is any zero entry in any row. If no zero entry is found, go to step 2A. Otherwise, count the number of zeros and select one of the processes at random. IFIND is the row selected by randomization. Go to step 3.

Step 2A. Processing the operation related to terminal: Count the number of operations having a zero in IC3 in the last group. Let these operations be in set 5. Select one of the operations at random from 5 and call it IFIND. In 5, check the operations corresponding to the same job as IFIND (having first 3 digits in common). Let these operations including IFIND be in a subset w. From among the operations in w, process the one having the maximum starting time (C_6).

Put this value in IC6. To this value, add the entry in IC4 and put it in IC7. Make $MAXM = MAXM + 1$ and put it in IC5. Turn the switches from 0 to -1 in IC3 for the operations in w. If there is no zero entry in IC3 (last group), go to step 6; otherwise, repeat this step.

3. More than one job in a machine

The transportation algorithm developed in section D can also be applied with some modifications for a production scheduling in which more than one job can be processed simultaneously in a single machine. Heat treating ovens, chemical treatment tanks, and the like are some of the examples in which usually more than one job is processed simultaneously. Another example is "Ironworker" which is a general-purpose tool that punches, notches, and cuts. Here men can work on two jobs simultaneously. This general service area has a number of such "utility" tools that can be run by any worker who needs them. It may be desirable to have this type of general purpose machine in the intermittent industry which deals with many different types of products.

VIII. SUMMARY

A. Summary and Conclusions

An algorithm was developed to generate feasible schedules for a multi-machines multiple facilities system. The algorithm was further improved by the concept of left-shifting to provide a better subset of the feasible solutions. In all the sample problems tested, the superiority of the left-shifting techniques over the pure random (non-shifting) procedure was verified with respect to minimum schedule time and different parameters of its distribution.

It must be emphasized that two factors pose potential severe limitations on the utility of Monte Carlo method for solving sequencing problem: the efficiency of the algorithm and rules for halting the sampling processes. Besides the left-shifting procedure, different biasing techniques were explored to improve the efficiency of the random sampling. Some of the biasing techniques introduced in this dissertation have been found even more powerful than the left-shifting procedure. However, the improvement by those biasing techniques over the left-shifting solutions is almost always accompanied by a considerable increase in CPU time/schedule. Total CPU time associated with any biasing technique could be reduced by specifying a smaller sample size in the algorithm without affecting the result considerably. It was, therefore, suggested that depending on the approximate minimum schedule time and CPU time, the scheduler can apply the left-shifting principle, combination techniques or biasing techniques with smaller sample size.

With regard to the rules for halting sampling processes, several distribution-free stopping rules have been suggested in this dissertation. The stopping rule functions were converging in nature; and, as such, the scheduler has some justification to believe that the schedule obtained at the time of stopping should be close enough to the best obtainable schedule by this algorithm. From our experience, the different stopping rules need different number of schedules before the sampling process stops. Therefore, a rough guideline was suggested regarding the sample size to be specified for different stopping rules and different solution techniques. To determine an approximation to the best obtainable schedule, a minimum bound value for the schedule time was estimated by using a three-parameter Weibull distribution which permits the calculation of the probability of further improvement in the solution at any instant.

In a practical situation, the scheduler may be interested in performing some operations on a particular machine of a facility. For example, the last finishing operation should be done on the machine which can maintain required precision. Also, the technological orders of different jobs may not be independent of other jobs in the sense that a particular operation on a commodity may need the completion of an operation of another commodity. We come across this situation in an assembly operation. The proposed algorithm can be applied in the above cases without any basic modifications.

The algorithm for the multi-machines multiple-facilities system was further modified to be applied to a special type of transportation problem. This modified version could be equivalently applied

in a job shop situation where more than one job can be simultaneously processed on the same machine.

B. Recommendations

The decision criterion used in this dissertation as a measure of performance is minimizing make-span time. In the current literature the emphasis has been on this measure of performance and it is argued that all other measures such as minimizing the in-process inventory, minimizing the costs of the machines or minimizing penalty costs of the jobs (if they are delivered later than the promised due date), are directly or indirectly related to this function. In defense of this measure, Manne (1960) states:

The economist, conditioned as he is to take a dim view of any minimum and other than dollar costs, will find it difficult to be altogether happy with Johnson's criterion, the minimization of t , the make-span. In defending this choice of minimand, however, it should be pointed out that t is likely to be correlated with dollar costs. In minimizing t we may conceivably also obtain the following cost and profit benefits: (a) a lowered amount of inventory ties up in work-in-process, (b) a shorter average customer delay time, and (c) a lower amount of idle time incurred prior to the performance of all currently booked jobs--i.e., a greater capacity to take additional work as new orders materialize. To the extent that all of these factors work in the reasonable proxy variable for economic cost. The job sequence that serves to minimize the make-span might also be one that scores quite well on the criterion of dollar costs.

It may be noted that Manne states that one may conceivably also obtain benefits other than minimizing the maximum flow time "to the extent that other factors were in the same direction." The question "Do the factors work in the same direction or not and even if they work in the same direction, what happens if their rates of change are

different" becomes pertinent. "No mathematical or empirical study has appeared in literature indicating the validity of Manne's statement. Beenhakker's (1963) only attempt in this direction fails to give any conclusion because in proving the equivalence of various criteria, he essentially considers time as a measure of performance instead of costs" (Gupta, 1971). In the absence of any mathematical analysis, Gupta (1971) considered a hypothetical example and showed that Manne's argument doesn't hold in general.

In the future research work, the inclusion of other criteria should be considered. In a job shop, all jobs are not of equal value. Every time a job is processed in a machine its value is increased. The different jobs have different due dates. For each job, we can assign different priority indices, considering the above factors separately or on the basis of a linear combination of those factors. In the process of sampling, jobs should be selected according to higher priority indices. There has been a reasonable amount of discussion in the literature on the selection of a decision criterion for scheduling problems. Quite recently multiple decision criterion (such as goal programming) has been investigated. Therefore, it is suggested that use of multiple decision technique be considered for scheduling problems such as those discussed in this dissertaiton.

Further biasing techniques should be developed in order to provide better schedules. In this regard, multiple left-shifting principle and other techniques mentioned in Chapter 5 may be explored.

As regards to the transportation algorithm, besides the different possible extensions proposed and discussed in Chapter 7, section G, it is expected that the algorithm may be improved with respect to computer time by the following alteration.

Referring to the algorithm developed in Chapter 7, section D, during the solution procedure, many times we know that some origins and destinations are already "blocked" (step 4). Even then, in searching for zeroes and selecting one of them from IC3 (step 2), the algorithm does not exclude those origins and destinations. In the Monte Carlo procedure, we generate many schedules and select the one with the minimum total time. So, especially for a larger problem, a considerable amount of CPU time is expended in executing step 2 for the entire run. The following change in the algorithm will possibly reduce this time to a great extent.

i) For the terminal, make the entry in CLO equal to the number of trucks.

ii) Change steps 2 and 4 of the algorithm in Section D as follows:

In step 2, check CLO whether there is any positive entry. If no positive entry is found, it implies that a feasible route has been obtained for each truck and step 6 follows. Otherwise, count the number of zeroes in IC3 corresponding to the machines having positive entries in CLO and select one of the processes at random. IFIND is the row selected by randomization.

In step 4, multiply the entry only in CLO by -1 corresponding to the machine. Leave the other switches in IC3 as they are.

IX. BIBLIOGRAPHY

- Akers, S. B. and Friedman, J. (1955), "A Non-Numerical Approach to Production Scheduling Problems", Operations Research, 3, 657.
- Ashour, S. (1967), "A Decomposition Approach for the Machine Scheduling Problem", International Journal of Production Research, 6, 109.
- Ashour, S. (1970), "A Branch-and-Bound Algorithm for Flow Shop Scheduling Problems", AIIE Transactions, 2, 172.
- Ashour, S. and Parker, R. G. (1971), "A Precedence Graph Algorithm for the Shop Scheduling Problem", Operational Research Quarterly, 22, 165.
- Ashour, S. and Quraishi, N. (1969), "Investigation of Various Bounding Procedures for Production Scheduling Problems", International Journal of Production Research, 7, 249.
- Bae, H. M. (1972), "A Heuristic Solution to the General Job Shop Scheduling Problem", M. S. Thesis, Iowa State University of Science and Technology, Ames, Iowa.
- Balas, E. (1967), "Discrete Programming by Filler Method", Operations Research, 15, 915.
- Balas, E. (1969), "The Machine Sequencing via Disjunctive Graphs: An Implicit Enumeration Algorithm", Operations Research, 17, 941.
- Beenhakker, H. L. (1963), "The Development of Alternative Criteria for Optimality in the Machine Sequencing Problems", Unpublished Ph.D. Thesis, Purdue University, West Lafayette, Indiana.
- Bellman, R. (1956), "Mathematical Programming of Scheduling Theory", SIAM Journal on Applied Mathematics, 4, 168.
- Bellman, R. (1962), "Dynamic Programming Treatment of the Travelling Sales Problem", Journal of Association for Computing Machinery, 9, 61.
- Bellman, R. and Malone, J. C. (1971), "Pathology of Travelling Salesman Subtour-Elimination Algorithms", Operations Research, 19, 778.
- Bowman, E. H. (1959), "Schedule Sequencing Problems", Operations Research, 7, 621.
- Brooks, G. H. and White, C. R. (1965), "An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem", Journal of Industrial Engineering, 16, 34.
- Brooks, Samuel H. (1953), "A Discussion of Random Methods for Seeking Maxima", Operations Research, 6, 244.

- Brown, A. P. G. and Lomnicki, Z. A. (1966), "Some Applications of the Branch-and-Bound Algorithm to Machine Scheduling Problems", Operational Research Quarterly, 17, 173.
- Charlton, J. M. and Death, C. C. (1970a), "A Generalized Machine-Scheduling Algorithm", Operational Research Quarterly, 21, 127.
- Charlton, J. M. and Death, C. C. (1970b), "A Method of Solution for General Machine-Scheduling Problems", Operations Research, 18, 689.
- Conway, R. W., Maxwell, W. L. and Miller, L. W. (1967), Theory of Scheduling, Addison-Wesley, Reading, Mass.
- Elmaghraby, S. E. (1968a), "The Machine Sequencing Problem--Review and Extension", Naval Research Logistic Quarterly, 15, 205.
- Elmaghraby, S. E. (1968b), "The One Machine Sequencing Jobs with Delay Costs", Journal of Industrial Engineering, 2, 105.
- Elmaghraby, S. E. and Cole, R. T. (1963), "On the Control of Production in Small Job Shops", Journal of Industrial Engineering, 14,
- Epstein, B. (1960), "Elements of the Theory of Extreme Values", Technometrics, 2, 27.
- Fisher, H. and Thompson, G. L. (1963), in Industrial Scheduling, Muth, J. F. and Thompson, G. L., Eds., Prentice-Hall, Inc., Englewood Cliffs, New Jersey, Chapter 15.
- Gere, W. S. (1966), "Heuristics in Job Shop Scheduling", Management Science, 13, 167.
- Giffler, B. and Thompson, G. L. (1960), "Algorithms for Solving Production Scheduling Problems", Operations Research, 8, 487.
- Giffler, B., Thompson, G. L. and Van Ness, V. (1963), in Industrial Scheduling, Muth, J. F. and Thompson, G. L., Eds., Prentice-Hall, Inc., Englewood Cliffs, New Jersey, Chapter 3.
- Gillett, B. E. (1976), Introduction to Operations Research--A Computer-Oriented Algorithmic Approach, McGraw-Hill Book Company, New York.
- Greenberg, H. H. (1968), "A Branch-and-Bound Solution to the General Scheduling Problems", Operations Research, 16, 353.
- Gumbel, E. J. (1954), Statistical Theory of Extreme Values and Some Practical Applications, National Bureau of Standards Applied Mathematics Series 33.
- Gumbel, E. J. (1953), Statistics of Extremes, Columbia University, New York, N. Y.

Gupta, J. N. D. (1970), "M-Stage Flow Shop Scheduling by Branch and Bound", Operations Research (India) 7, 37.

Gupta, J. N. D. (1971), "M-Stage Scheduling Problem--A Critical Approach", International Journal of Production Research, 7, 249.

Hardgrave, W. W. and Nemhauser, G. L. (1963), "A Geometric Model and a Graphical Algorithm for a Sequencing Problem", Operations Research, 11, 889.

Heller, J. (1960), "Some Numerical Experiments for an MXJ Flow Shop and Its Decision Theoretic Aspects", Operations Research, 8, 178.

Heller, J. and Logemann, G. (1962), "An Algorithm for the Construction and Evaluation of Feasible Schedules", Management Science, 8, 168.

Ignal, E. and Schrage, L. (1965), "Application of the Branch-and-Bound Technique to Some Flow Shop Scheduling Problems", Operations Research, 13, 400.

Jackson, J. R. (1956), "An Extension of Johnson's Results on Job Lot Scheduling", Naval Research Logistics Quarterly, 3, 201.

Jackson, J. R. (1957), "Simulation Research on Job Shop Production", Naval Research Logistic Quarterly, 4, 287.

Johnson, S. M. (1954), "Optimal Two-and-Three Stage Production Schedules with Setup Time Included", Naval Research Logistic Quarterly, 1, 61.

Kammin, Tamra J. (1976), "Optimizing Truck Routes", Unpublished M. S. Research Paper, Department of Mathematics, Iowa State University, Ames, Iowa.

Little, J., Murry, K., Sweeney, D., and Karmel, C. (1963), "An Algorithm for the Travelling Salesman Problem", Operations Research, 11, 972.

Lomnicki, Z. A. (1965), "A Branch-and-Bound Algorithm for the Exact Solution of the Three-Machine Scheduling Problem", Operational Research Quarterly, 16, 89.

Manne, A. S. (1960), "On the Job-Shop Scheduling Problem", Operations Research, 8, 219.

McRoberts, K. L. (1966), "Optimization of Facility Layout", Ph. D. Thesis, Iowa State University of Science and Technology, Ames, Iowa.

McRoberts, K. L. (1971), "A Search Model for Evaluating Combinatorially Explosive Problems", Operations Research, 19, 1331.

- Nabeshima, I. (1967a), "Some Extensions of the M-machine Scheduling Problem", Journal of Operations Research Society, Japan, 10, 1.
- Nabeshima, I. (1967b), "On the Bounds of Make-spans and Its Application in M-machine Scheduling Problems", Journal of Operations Research Society, Japan, 9, 98.
- Randolph, P. H. (1968), "Optimal Stopping Rules for Multinomial Observations", International Journal for Theoretical and Applied Statistics, 14, 48.
- Randolph, P. H., Swinson, G. and Ellingsen, C. (1973), "Stopping Rules for Sequencing Problems", Operations Research, 21, 1309.
- Reiter, S. and Sherman, G. (1965), "Discrete Optimization", Journal of the Society for Industrial and Applied Mathematics, 13, 864.
- Rowe, A. J. (1960), "Toward a Theory of Scheduling", Journal of Industrial Engineering, 11, 125.
- Schrage, L. (1970), "Solving Resource Constrained Network Problems by Implicit Enumeration--Non Preemptive Case", Operations Research, 18, 263.
- Schwartz, E. S. (1964), "A Heuristic Procedure for Parallel Sequencing with Choice of Machines", Management Science, 10, 767.
- Smith, Wayne E. (1956), "Various Optimizers for Single-Stage Production", Naval Research Logistics Quarterly, 3, 59.
- Story, A. E. and Wagner, H. M. (1963), in "Industrial Scheduling", Muth, J. E. and Thompson, G. L. Eds., Prentice-Hall, Inc., Englewood Cliffs, N.J. Chapter 14.
- Thompson, G. L. and Karg, R. L. (1964), "A Heuristic Approach to Solving Travelling-Salesman Problems", Management Science, 10, 225.
- Wagner, H. M. (1959), "An Integer Linear Programming Model for Machine Scheduling", Naval Research Logistic Quarterly, 6, 131.

X. ACKNOWLEDGEMENTS

A number of individuals deserve my appreciation in this research effort.

I wish to thank, without reservation, my major professors, Dr. Keith L. McRoberts and Dr. Howard Meeks, for their assistance, patience, willingness, and valuable guidance and support in the preparation of this dissertation.

Special thanks are also due to Dr. V. A. Sposito for many valuable discussions, research ideas and help in the development of all the computer programming for this research study, which greatly facilitated early completion of my work.

I would also like to thank Dr. John C. Even and Dr. B. V. Sukhatme for serving on my committee.

I would like to thank Bangladesh University of Engineering and Technology for providing me the scholarship and deputation during my graduate study at Iowa State University.

I should sincerely mention the help of my dear friend, Abdurrazzagh Ennagiari, in making different figures and sketches for my dissertation. Three individuals who deserve special thanks for their inspiration and encouragement in different phases of my graduate study are Dr. John Trzeciak, Mufazzal Chowdhury and Yeakup Busmaci.

Last but not least, I want to express my deepest appreciation to all my friends and relatives for their good wishes and especially to my wife Monowara for her love and understanding and my children, Zakia, Mahboob and Syeda for their patience during my three years' absence from them.

XI. APPENDIX I. LEFT-SHIFTING ALGORITHM

```

COMMON IC1(500,4),IC2(500,4,15),IC3(500),IC4(500),IC5(500)
COMMON IC6(500),IC7(500),IND(500),ISAVE(500,3),ICS3(500)
DIMENSION IDT(1500)
DO 1 I=1,500
  IC3(I)=0
  IC4(I)=0
  IC5(I)=0
  IC6(I)=0
  IC7(I)=0
  ICS3(I)=0
1 IND(I)=0
DO 3 I=1,500
DO 3 J=1,4
3 IC1(I,J)=0
DO 4 I=1,500
DO 4 J=1,4
DO 4 K=1,15
4 IC2(I,J,K)=0
DO 88 I=1,1500
88 IDT(I)=0
C
C READ
C IN
C SCH. INFORMATION
C
  READ(5,5) NN,NN2,IX,IG
5 FORMAT(20I4)
DO 80 I=1,NN
80 READ(5,11) IND(I)
11 FORMAT(I2)
  JL=0
DO 6 I=1,NN
  READ(5,8)(ICS3(J),J=1,52)
DO 90 J=1,4
90 IC1(I,J)=ICS3(J)
  J=1
  JJ=4
DO 91 K=1,15
  J=J+4
  JJ=JJ+4
  JL=0
DO 91 JK=J,JJ
  JL=JL+1
91 IC2(I,JL,K)=ICS3(JK)
6 CONTINUE

```



```

9  FORMAT(' ',20(I1))
9  FORMAT(13(I2,I1,I2,I1))
  WRITE(6,12)
12  FORMAT('1')
    DO 7 I=1,NN
  7  READ(5,687) IC3(I),IC4(I)
687  FORMAT(I1,I2)
    DO 24 I=1,NN
24   ICS3(I)=IC3(I)
C
C  NOW TO
C  FIND AN
C  OPTIMAL
C  SCHED.
C
  NPP=0
  MXX=100000
  IFF=0
  MYX=0
  NMN=-1
  NPROB=0
  LX=1
  LLX=NN
505  CONTINUE
  NPROB=NPROB+1
  NMN=NMN+1
500  CONTINUE
  IF(NMN.EQ.NN2) GO TO 1000
1010  CONTINUE
  IFIND=0
  CALL RANDU(IX,IY,XX)
  IX=IY
  NX=XX*NN+.99999
  IF(NX.GE.NN) NX=NN
  IF(NX.LE.0)  NX=1
  DO 15 I=1,NN
  IF(IC3(I).EQ.0) GO TO 16
15  CONTINUE
C
C  BRANCH OUT IF
C  C3 HAS
C  NO ZEROS
C
  GO TO 100
16  DO 17 I=NX,NN
  IF(IC3(I).EQ.0) IFIND=I
  IF(IC3(I).EQ.0) GO TO 19
17  CONTINUE
  DO 19 I=1,NX

```

```

      IF(IC3(I).EQ.0) IFIND=I
      IF(IC3(I).EQ.0) GO TO 18
12 CONTINUE
18 CONTINUE
C A PROCESS
C HAS BEEN
C SELECTED
      IC3(IFIND)=-1
      MAX=0
      IU=IC1(IFIND,4)
      IMNU=IC1(IFIND,1)
      IJCB=IC1(IFIND,3)
      DO 20 I=1,NN
      IF((IC1(I,1).EQ.IMNU.AND.IC1(I,3).EQ.IJCB.AND.IU.EQ.IC1
1(I,4))
      XIC3(I)=-1
      IF(IC1(I,3).NE.IJCB) GO TO 20
      IF(IC7(I).GE.MAX) MAX=IC7(I)
20 CONTINUE
C
C TO SELECT
C AT RANDOM THE
C NEXT PROCESS
C
      CALL TIMELG(IFIND,NN,MAX)
      IF(IC2(IFIND,3,1).EQ.0) GO TO 30
      NC=JND(IFIND)
      CALL RANDU(IX,IY,XX)
      IX=IY
      XXX=XX*NC+.99999
      ISFL=XXX
      IF(ISFL.LT.1) ISFL=1
      IF(ISEL.GT.NC) ISEL=NC
      DO 21 L=ISEL,ISFL
      I=IFIND
      N1=IC2(I,1,L)
      N2=IC2(I,2,L)
      N3=IC2(I,3,L)
      N4=IC2(I,4,L)
      DO 22 J=1,NN
      IF(I.EQ.J) GO TO 22
      IF(N1.NE.IC1(J,1)) GO TO 22
      IF(N1.EQ.IC1(J,1).AND.N2.EQ.IC1(J,2).AND.N3.EQ.IC1
1(J,3).AND.N4.
      XEQ.IC1(J,4)) IC3(J)=0
22 CONTINUE
21 CONTINUE
30 CONTINUE
501 GO TO 500

```

```

100   CONTINUE
      MAX=0
      DO 60 I=1,NN
      IF(IC7(I).GE.MAX) MAX=IC7(I)
60    CONTINUE
      IDT(NPROB)=MAX
      J=0
      IF(MAX.GT.MYX) MYX=MAX
      IF(MAX.GT.MXX) GO TO 600
      DO 35 I=1,NN
35    WRITE(6,40) IC3(I),IC5(I),IC6(I),IC7(I)
40    FORMAT(' ',4(15,2X))
      WRITE(6,12)
      NPP=NPROB
      MXX=MAX
      DO 61 I=LX,LLX
      J=J+1
      ISAVE(I,1)=IC5(J)
      ISAVE(I,2)=IC6(J)
61    ISAVE(I,3)=IC7(J)
600   CONTINUE
      DO 62 I=1,NN
      IC5(I)=0
      IC6(I)=0
      IC7(I)=0
      62 IC3(I)=ICS3(I)
      GO TO 505
1000  CONTINUE
      CALL STOPJ(MXX,MYX,NPROB,IDT,ISX,IFF)
      IF(ISX.EQ.1) GO TO 1005
      NMN=0
      GO TO 1010
1005  CONTINUE
      WRITE(6,1002) MXX
1002  FORMAT('I',I4)
602   FORMAT(3I5)
      DO 6012 I=1,NPROB
      WRITE(6,6013) IDT(I)
6012  CONTINUE
6013  FORMAT(' ',15)
      STOP
      END

```

```

SUBROUTINE STOPJ(MXX,MYX,NPROB,IPT,ISX,IFF)
DIMENSION IPT(1500)
ISX=0
IFF=IFF+1
IF(IFF.NE.1) GO TO 10
LX1=MYX-MXX
RETURN
10 LL2=MYX-MXX
WRITE(6,6) NPROB,MXX,MYX,LX1,LL2,LINV
6  FORMAT(' ',6(2X,I5))
LINV=LL2-LX1
IF(LINV.LT.1) GO TO 20
LX1=LL2
RETURN
20 CONTINUE
ISX=1
L=1
JJ=NPROB+1
DO 30 I=L,NPROB
IJ=I+1
DO 40 J=IJ,JJ
IF(IPT(I).GE.IPT(J)) GO TO 40
IS=IPT(J)
IPT(J)=IPT(I)
IPT(I)=IS
40 CONTINUE
L=L+1
30 CONTINUE
RETURN
END

```

```

SUBROUTINE TIMELG(IFIND,NN,MAX)
COMMON IC1(500,4),IC2(500,4,15),IC3(500),IC4(500),IC5(500)
COMMON IC6(500),IC7(500),IND(500),ISAVE(500,3),ICS3(500)
IS=IC1(IFIND,1)
ISS=IC1(IFIND,2)
DO 1 I=1,NN
L=I
IF(IS.EQ.IC1(I,1).AND.ISS.EQ.IC1(I,2)) GO TO 2
1 CONTINUE
2 DO 3 I=L,NN
LL=I
IF(IS.EQ.IC1(I,1).AND.ISS.EQ.IC1(I,2)) GO TO 3
LL=LL-1
GO TO 4
3 CONTINUE
IF(IS.EQ.IC1(NN,1).AND.ISS.EQ.IC1(NN,2)) LL=NN
4 L5=0
DO 5 I=L,LL
IF(IC5(I).LE.IMAX) GO TO 5
L5=I
IMAX=IC5(I)
5 CONTINUE
IF(L5.EQ.0) MAX5=0
IOUT=0
CALL RECHK(IOUT,L,LL,IFIND,IMAX,NN,MJM)
MAX=MJM
IF(IOUT.EQ.1) RETURN
IF(L5.EQ.0) GO TO 6
MAX5=IC7(L5)
IF(IMAX.NE.0) GO TO 6
IC5(IFIND)=1
IC7(IFIND)=IC4(IFIND)
GO TO 10
6 IC5(IFIND)=IMAX+1
IF(MAX.GE.MAX5) GO TO 9
IC7(IFIND)=MAX5 +IC4(IFIND)
IC6(IFIND)=MAX5
RETURN
9 IC6(IFIND)=MAX
IC7(IFIND)=MAX+IC4(IFIND)
10 CONTINUE
MAX=0
DO 11 I=1,NN
11 IF(IC7(I).GE.MAX) MAX=IC7(I)
RETURN
END

```

```

SUBROUTINE RECHK(IOUT,L,LL,IFIND,IMAX,NN,MJM)
COMMON IC1(500,4),IC2(500,4,15),IC3(500),IC4(500),IC5(500)
COMMON IC6(500),IC7(500),IND(500),ISAVE(500,3),ICS3(500)
IOUT=0
ISS=0
INTV=IC4(IFIND)
MJM=0
DO 41 K=1,NN
IF(IC1(K,3).NE.IC1(IFIND,3)) GO TO 41
IF(IC7(K).GE.MJM) MJM=IC7(K)
41 CONTINUE
MAXM=IMAX
IF(MJM.EQ.0.AND.MAXM.EQ.0) GO TO 2000
IF(MAXM.EQ.0) GO TO 1001
DO 50 K=1,MAXM
DO 50 I=L,LL
IF(IC5(I).NE.K) GO TO 50
ITIME=IC7(I)
IZ=I
L2=L
IF(MJM.GT.ITIME) GO TO 50
II=I
IAVA=IC6(I)
IF(IAVA.LE.MJM) MJM=ITIME
IF(MJM.GE.IAVA) GO TO 50
IAAA=IAVA-MJM
IF(INTV.GT.IAAA.AND.K.EQ.1) MJM=ITIME
IF(INTV.GT.IAAA.AND.K.EQ.1) GO TO 50
IF(IAVA.GT.0.AND.IC5(I).EQ.1.AND.INTV.LE.IAAA) GO TO 500
IF(ISS.EQ.1) GO TO 6000
ISS=1
IF(INTV.LE.IAAA) GO TO 500
6000 CONTINUE
KK=K+1
IF(KK.GT.MAXM) GO TO 50
DO 52 IL=L,LL
IF(IC5(IL).NE.KK) GO TO 52
IF(INTV.LE.IAVA) MJM=ITIME
IAVA=IC6(IL)-ITIME
IZ=IL
IF(INTV.LE.IAVA) GO TO 500
52 CONTINUE
GO TO 50
500 IAM=I7
IXX=IC5(IZ)
DO 60 KK=L,LL
IF(IC5(KK).LT.IXX) GO TO 60
IC5(KK)=IC5(KK)+1

```

```
60 CONTINUE
   IAM=IFIND
   ICS(IAM)=IXX
   IF(IXX.NE.1) GO TO 70
   IC7(IAM)=INTV+MJM
   IC6(IAM)=0+MJM
   IOUT=1
   RETURN
70  CONTINUE
   IC6(IAM)=MJM
   IC7(IAM)=MJM      +IC4(IFIND)
   IOUT=1
   GO TO 1001
80  CONTINUE
50  CONTINUE
1001 RETURN
2000 ICS(IFIND)=1
     IC6(IFIND)=0
     IC7(IFIND)=IC4(IFIND)
     IOUT=1
     RETURN
     END
```

XII. APPENDIX II. TRUCK ROUTING ALGORITHM

```

COMMON IC1(500,4),IC2(500,4,15),IC3(500),IC4(500),IC5
1(500)
COMMON IC6(500),IC7(500),IND(500),ISAVE(500,3),ICS3(500)
COMMON IDX(25),IMX(25)
DIMENSION IDT(1500)
DO 1 I=1,500
  IC3(I)=0
  IC4(I)=0
  IC5(I)=0
  IC6(I)=0
  IC7(I)=0
  ICS3(I)=0
1 IND(I)=0
  DO 3 I=1,500
    DO 3 J=1,4
      3 IC1(I,J)=0
        DO 4 I=1,500
          DO 4 J=1,4
            DO 4 K=1,15
              4 IC2(I,J,K)=0
                DO 88 I=1,1500
118  IDT(I)=0
C
C READ
C IN
C SCH. INFORMATION
C
  READ(5,97) IDX
  READ(5,97) IMX
97  FORMAT(25I3)
  READ(5,5) NN,NN2,IX,IG
  5 FORMAT(20I4)
  DO 80 I=1,NN
80  READ(5,11) IND(I)
11  FORMAT(I2)
  JL=0
  DO 6 I=1,NN
  READ(5,8)(ICS3(J),J=1,52)
8  FORMAT(13(2I1,I2,I2))
  DO 90 J=1,4
90  IC1(I,J)=ICS3(J)
  J=1
  JJ=4
  DO 91 K=1,15
  J=J+4
  JJ=JJ+4
  JL=0
  DO 91 JK=J,JJ

```



```

      JL=JL+1
91  IC2(I,JL,K)=ICS3(JK)
      6  CONTINUE
      7  FORMAT(' ',20I1)
      WRITE(6,12)
      12  FORMAT('1')
      DO 7 I=1,NN
      7  READ(5,687) IC3(I),IC4(I)
687  FORMAT(I1,I2)
      DO 24 I=1,NN
      24  ICS3(I)=IC3(I)
C
C  NOW TO
C  FIND AN
C  OPTIMAL
C  SCHED.
C
      NPP=0
      MXX=100000
      IFF=0
      MYX=0
      NMN=-1
      NPROB=0
      LX=1
      LLX=NN
505  CONTINUE
      NPROB=NPROB+1
      NMN=NMN+1
      500  CONTINUE
      IF(NMN.EQ.NN2) GO TO 1000
      1010 CONTINUE
      IFIND=0
      CALL RANDU(IX,IY,XX)
      IX=IY
      NX=XX*NN+.99999
      IF(NX.GE.NN) NX=NN
      IF(NX.LE.0) NX=1
      DO 15 I=1,NN
      IF(IC3(I).EQ.0) GO TO 16
      15  CONTINUE
C
C  BRANCH OUT IF
C  C3 HAS
C  NO ZEROS
C
      GO TO 100

```

```

16 DO 17 I=NX,NN
    IF(IC3(I).EQ.0) IFIND=I
    IF(IC3(I).EQ.0) GO TO 18
17 CONTINUE
    DO 19 I=1,NX
    IF(IC3(I).EQ.0) IFIND=I
    IF(IC3(I).EQ.0) GO TO 18
19 CONTINUE
18 CONTINUE
C A PROCESS
C HAS BEEN
C SELECTED
  IC3(IFIND)=-1
  MAX=0
  IU=IC1(IFIND,4)
  IMNU=IC1(IFIND,1)
  IJOB=IC1(IFIND,3)
  DO 20 I=1,NN
  IF(IC1(I,1).EQ.IMNU.AND.IC1(I,3).EQ.IJOB.AND.IU.EQ.IC1
XIC3(I)=-1
  IF(IC1(I,3).NE.IJOB) GO TO 20
  I(I,4))
  IF(IC7(I).GE.MAX) MAX=IC7(I)
20 CONTINUE
C
C TO SELECT
C AT RANDOM THE
C NEXT PROCESS
C
  CALL TIMELG(IFIND,NN,MAX)
  IF(IC2(IFIND,3,1).EQ.0) GO TO 30
  NC=IND(IFIND)
  LS=1
  IVV=1
  DO 21 LB=1,IG
21 IF(IDX(LB).LT.IFIND) IVV=IVV+1
  ICH=IMX(IVV)
  IF(IVV.EQ.1) GO TO 497
  LS=IDX(IVV-1)+1
497 LSS=IDX(IVV)
  DO 494 LB=LS,LSS
  IF(IC5(LB).EQ.ICH) GO TO 395
494 CONTINUE
  GO TO 300
395 CONTINUE
  DO 396 LB=LS,LSS
396 IC3(LB)=-1
  IMX(IVV)=-1*ICH
300 CONTINUE

```

```

DO 22 IKK=1,NC
I=IFIND
IVV=1
N1=IC2(I,1,IKK)
N2=IC2(I,2,IKK)
N3=IC2(I,3,IKK)
N4=IC2(I,4,IKK)
DO 22 J=1,NN
IF(I.EQ.J) GO TO 22
IF(N1.NE.IC1(J,1)) GO TO 22
IF(N4.NE.IC1(J,4)) GO TO 22
IF(N2.NE.IC1(J,2).OR.N3.NE.IC1(J,3)) GO TO 22
DO 23 INK=1,IG
23 IF(IDX(INK).LT.J) IVV=IVV+1
IF(IMX(IVV).LF.0) GO TO 22
IC3(J)=0
22 CONTINUE
30 CONTINUE
501 GO TO 500
100 CONTINUE
MAX=0
DO 60 I=1,NN
IF(IC7(I).GE.MAX) MAX=IC7(I)
60 CONTINUE
MAX=0
IG1=IG-1
IJ4=IDX(IG1)
DO 327 I=IJ4,NN
327 MAX=MAX+IC7(I)
IDT(NPROB)=MAX
J=0
IF(MAX.GT.MYX) MYX=MAX
IF(MAX.GT.MXX) GO TO 600
DO 35 I=1,NN
35 WRITE(6,40) IC3(I),IC5(I),IC6(I),IC7(I)
40 FORMAT(' ',5(15,2X))
WRITE(6,12)
NPP=NPROB
MXX=MAX
DO 61 I=LX,LLX
J=J+1
ISAVE(I,1)=IC5(J)
ISAVE(I,2)=IC6(J)
61 ISAVE(I,3)=IC7(J)
600 CONTINUE
DO 62 I=1,NN
IC5(I)=0
IC6(I)=0

```

```
      IC7(I)=0
62   IC3(I)=ICS3(I)
      I=IG-1
      DO 875 JW=1,I
875  IMX(JW)=-1*IMX(JW)
      GO TO 505
1000 CONTINUE
      CALL STOPJ(MXX,MYX,NPROB,IDT,ISX,IFF)
      IF(ISX.EQ.1) GO TO 1005
      NMN=0
      GO TO 1010
1005 CONTINUE
      WRITE(6,1002) MXX
1002  FORMAT(*1*,I4)
602  FORMAT(3I5)
      DO 6012 I=1,NPROB
      WRITE(6,6013) IDT(I)
6012 CONTINUE
6013  FORMAT(* ',I5)
      STOP
      END
```

```
SUBROUTINE STOPJ(MXX,MYX,NPROB,IPT,ISX,IFF)
DIMENSION IPT(1500)
ISX=0
IFF=IFF+1
IF(IFF.NE.1) GO TO 10
LX1=MYX-MXX
RETURN
10 LL2=MYX-MXX
WRITE(6,6) NPROB,MXX,MYX,LX1,LL2,LINV
6  FORMAT(' ',6(2X,I5))
LINV=LL2-LX1
IF(LINV.LT.1) GO TO 20
LX1=LL2
RETURN
20 CONTINUE
ISX=1
L=1
JJ=NPROB+1
DO 30 I=L,NPROB
IJ=I+1
DO 40 J=IJ,JJ
IF(IPT(I).GE.IPT(J)) GO TO 40
IS=IPT(J)
IPT(J)=IPT(I)
IPT(I)=IS
40 CONTINUE
L=L+1
30 CONTINUE
RETURN
END
```

```

SUBROUTINE TIMELG(IFIND,NN,MAX)
COMMON IC1(500,4),IC2(500,4,15),IC3(500),IC4(500),IC5(500)
COMMON IC6(500),IC7(500),IND(500),ISAVE(500,3),ICS3(500)
IS=IC1(IFIND,1)
ISS=IC1(IFIND,2)
DO 1 I=1,NN
L=I
IF(IS.EQ.IC1(I,1).AND.ISS.EQ.IC1(I,2)) GO TO 2
1 CONTINUE
2 DO 3 I=L,NN
LL=I
IF(IS.EQ.IC1(I,1).AND.ISS.EQ.IC1(I,2)) GO TO 3
LL=LL-1
GO TO 4
3 CONTINUE
IF(IS.EQ.IC1(NN,1).AND.ISS.EQ.IC1(NN,2)) LL=NN
4 IMAX=0
L5=0
DO 5 I=L,LL
IF(IC5(I).LE.IMAX) GO TO 5
L5=I
IMAX=IC5(I)
5 CONTINUE
IF(L5.EQ.0) MAX5=0
IQOUT=0
CALL RECHK(IQOUT,L,LL,IFIND,IMAX,NN,MJM)
MAX=MJM
IF(IQOUT.EQ.1) RETURN
IF(L5.EQ.0) GO TO 6
MAX5=IC7(L5)
MAX5=0
IF(IMAX.NE.0) GO TO 6
IC5(IFIND)=1
IC7(IFIND)=IC4(IFIND)
GO TO 10
6 IC5(IFIND)=IMAX+1
IF(MAX.GE.MAX5) GO TO 9
IC7(IFIND)=MAX5 +IC4(IFIND)
IC6(IFIND)=MAX5
RETURN
9 IC6(IFIND)=MAX
IC7(IFIND)=MAX+IC4(IFIND)
10 CONTINUE
MAX=0
DO 11 I=1,NN
11 IF(IC7(I).GE.MAX) MAX=IC7(I)
RETURN
END

```

```

SUBROUTINE RECHK(IOUT,L,LL,IFIND,IMAX,NN,MJM)
COMMON IC1(500,4),IC2(500,4,15),IC3(500),IC4(500),IC5
1(500)
COMMON IC6(500),IC7(500),IND(500),ISAVE(500,3),ICS3(500)
IOUT=0
ISS=0
INTV=IC4(IFIND)
MJM=0
DO 41 K=1,NN
IF(IC1(K,3).NE.IC1(IFIND,3)) GO TO 41
IF(IC7(K).GE.MJM) MJM=IC7(K)
41 CONTINUE
IF(MJM.GE.0) RETURN
MAXM=IMAX
IF(MJM.EQ.0.AND.MAXM.EQ.0) GO TO 2000
IF(MAXM.EQ.0) GO TO 1001
DO 50 K=1,MAXM
DO 50 I=L,LL
IF(IC5(I).NE.K) GO TO 50
ITIME=IC7(I)
IZ=I
L2=L
IF(MJM.GT.ITIME) GO TO 50
II=I
IAVA=IC6(I)
IF(IAVA.LE.MJM) MJM=ITIME
IF(MJM.GE.IAVA) GO TO 50
IAAA=IAVA-MJM
IF(INTV.GT.IAAA.AND.K.EQ.1) MJM=ITIME
IF(INTV.GT.IAAA.AND.K.EQ.1) GO TO 50
IF(IAVA.GT.0.AND.IC5(I).EQ.1.AND.INTV.LE.IAAA) GO TO 500
IF(ISS.EQ.1) GO TO 6000
ISS=1
IF(INTV.LE.IAAA) GO TO 500
6000 CONTINUE
KK=K+1
IF(KK.GT.MAXM) GO TO 50
DO 52 IL=L,LL
IF(IC5(IL).NE.KK) GO TO 52
IF(INTV.LE.IAVA) MJM=ITIME
IAVA=IC6(IL)-ITIME
IZ=IL
IF(INTV.LE.IAVA) GO TO 500
52 CONTINUE
GO TO 50
500 IAM=IZ
IXX=IC5(IZ)
DO 60 KK=L,LL
IF(IC5(KK).LT.IXX) GO TO 60
IC5(KK)=IC5(KK)+1

```

```
60 CONTINUE
   IAM=IFIND
   IC5(IAM)=IXX
   IF(IXX.NE.1) GO TO 70
   IC7(IAM)=INTV+MJM
   IC6(IAM)=0+MJM
   IDUT=1
   RETURN
70  CONTINUE
   IC6(IAM)=MJM
   IC7(IAM)=MJM   +IC4(IFIND)
   IDUT=1
   GO TO 1001
90  CONTINUE
50  CONTINUE
1001 RETURN
2000 IC5(IFIND)=1
      IC6(IFIND)=0
      IC7(IFIND)=IC4(IFIND)
      IDUT=1
      RETURN
      END
```