ANALYSIS AND COMPARISON OF VARIOUS LOWER-BOUNDS ON SCHEDULE TIMES FOR THE SOLUTION OF FLOW-SHOP PROBLEMS

by 689

MOHAMMED NAWA-ULLAH QURAISHI

B.E. (Mech.), Maulana Azad College of Technology

Bhopal, (M.P.), India, 1966

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1969

Approved by:

[illegible]

Major Professor

LD
2668
R4
1969
Q37

TABLE OF CONTENTS

ACKNOWLEDGEMENT

I wish to express my deep sense of appreciation to my Major Professor, Dr. Said Ashour, for his guidance, constructive criticism, helpful suggestions and the personal interest taken in the preparation of this master's report.

My thanks are also due to Dr. Frank A. Tillman, Head Department of Industrial Engineering; Dr. L. E. Grosh, Department of Industrial Engineering and Professor Alley H. Duncan, Department of Mechanical Engineering for their kind patronage.

I appreciate and thank Miss Kathleen Bergman and S. F. Quadri for their assistance in the proof reading.

I greatly acknowledge and thank Mrs. Marie Jirak for her valuable assistance in typing.

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The importance of the production scheduling problems has long been recognized by various types of industrial organizations. The control of production in industry usually centers around two distinct types of manufacturing processes: batch; and continuous. These two types are usually represented by the classic models of the job-shop and the assembly-line production systems. Consequently, production scheduling problems may be classified as Shop Production Scheduling and Line Production Scheduling. This research is concerned with a special class of the shop production scheduling problem.

The job-shop usually consists of a limited number of multi-purpose machines. A finite number of jobs are to be processed on one or more of these machines. In processing these jobs, certain technological requirements may be specified in advance. These requirements are usually referred to as machine ordering or routing. The most common and frequently referred to scheduling problem consists of finding the job sequence for processing J jobs on M machines such that a certain criterion is optimized. Among the criteria usually considered are: (1) minimization of the total time required to process all jobs on all machines, i.e., minimization of schedule time or make-span; (2) maximization of the profit by meeting the dead lines or due-dates; (3) minimization of the in-process inventory; (4) minimization of the total idle time on all machines; (5) maximization of the facility utilization; and (6) minimization of the total cost. However, the criterion considered in this research is that of minimizing the schedule time. This is because all bounding procedures considered for comparison compute lower-bounds on schedule time.

In shop scheduling problems, there exists a situation in which a certain number of jobs arrive simultaneously in a shop that is idle and immediately available for work. In this case, the number of jobs is considered fixed and known. Such a process refers to the static behavior of job-arrivals. There also exists other situation in which the jobs arrive continuously at random intervals. This process is therefore refers to the dynamic behavior of job-arrivals. The static situation usually leads to various deterministic models; however, the dynamic situation leads to several stochastic models.

In practice, the scheduling problem is usually of dynamic nature and hence stochastic models are of more interest. However, the deterministic models are not without inherent interest of their own. These models can be considered as a prelude to the more realistic stochastic models because of the following. First, it provides an approach to handle more complex dynamic situations as a series of deterministic models. Second, knowledge gained from work with static situations may be directly applicable to the dynamic situations. Thus it would be a wise step to attack more simple static problems because the experience gained from these experiments may direct the researchers to handle more complex and realistic situations.

1.1 Problem Formulation*

In shop scheduling problems, the order in which various machines perform a particular job provides a distinction between the two types of shops,

*Adapted from Ashour, S., Introduction to Scheduling, John Wiley & Sons, Inc., to appear.

usually referred to as flow-shop and job-shop. This research is concerned with the flow-shop scheduling problems in which each job is performed on a certain set of machines in an identical order. Due to this, the jobs are said to flow over machines along the same path. On the contrary, in job-shop scheduling problems, the machine ordering for each job may be different. To account for this, the job-shop problems are more complex than the flow-shop.

To help facilitate the formulation of this scheduling problem, a job is designated by an integer j and a machine by an integer m. Since an operation is defined as the processing of a job j on machine m, it may be symbolized by (jm).

The indexing of jobs and machines is arbitrary and preconceived; it does not necessarily correspond to the sequence in which the jobs are performed on each machine or to the order in which the machines process each job. Since the jobs may be performed in a sequence other than the preconceived one, a sequence of jobs is usually designated as

$$j_1, j_2, \ldots, j_k, \ldots, j_J$$

where J is the total number of jobs. As an example, j_k indicates the job index which is in the kth sequence-position, i.e., the kth position in the job permutation (hereafter referred to as job j_k).

On the other hand, in considering a permutation of the machines with respect to the preconceived order, the machines may be designated as

$$m_1, m_2, \ldots, m_\ell, \ldots, m_M$$

where M is the total number of machines. For example, m_ℓ means the machine index which is in the ℓ^{th} order-position, i.e., the ℓ^{th} position in the machine permutation. In general, the subscripts of j and m are used to denote the position in a job permutation and order in a machine permutation, respectively.

Consequently, since it will be necessary to consider permutations of the job-sequence on a particular machine, permutations of the machine-order for a particular job, and even permutations of both the job-sequence and the machine-order, the following set of operations are defined. First, the operation of a job in the k^{th} sequence-position, on machine m is designated

$$(j_k m), \qquad k = 1, 2, \ldots, J .$$

Second, the operation of job j on a machine in the ℓ^{th} order-position is designated

$$(j m_\ell), \qquad \ell = 1, 2, \ldots, M .$$

Finally, a specific operation involving a particular job j_k and a particular machine m_ℓ is denoted

$$(j_k m_\ell), \qquad k = 1, 2, \ldots, J, \quad \ell = 1, 2, \ldots, M .$$

In scheduling problems it is necessary to consider the order relations for each job performed through the machines and the sequence relations between the jobs processed on each machine. This leads to define a binary relation, usually referred to as precedence relation.

Considering three operations $(j_p m_q)$, $(j_r m_s)$ and $(j_u m_v)$, if the processing of job j_p on machine m_q can not be started after the processing of job j_r on machine m_s, then operation $(j_p m_q)$ is said to precede operation $(j_r m_s)$. This relation is designated

$$(j_p m_q) \prec (j_r m_s) .$$

It can also be said that operation $(j_r m_s)$ follows operation $(j_p m_q)$. The precedence relation has the following properties

1. transitive which means that if

$$(j_p m_q) \prec (j_r m_s) \qquad \text{and} \qquad (j_r m_s) \prec (j_u m_v),$$

then

$$(j_p m_q) \prec (j_u m_v);$$

2. nonreflexive which means that

$$(j_p m_q) \not\prec (j_p m_q) .$$

Or, in words, there is no operation which precedes itself; and

3. antisymmetric which means that if

$$(j_p m_q) \prec (j_r m_s) ,$$

then

$$(j_r m_s) \not\prec (j_p m_q) .$$

Now considering the two operations $(j_p m_q)$ and $(j_r m_s)$ having a job or a machine in common, i.e., $j_p = j_r$ or $m_q = m_s$, the operation $(j_p m_q)$ is said to directly-precede operation $(j_r m_s)$ if there is no intermediary operations. Such a relation is designated

$$(j_p m_q) \prec\prec (j_r m_s) .$$

This implies that operation $(j_r m_s)$ next-follows operation $(j_p m_q)$. The above statement implies that

1. $(j_p m_q) \neq (j_r m_s)$,

2. $(j_p m_q) \prec (j_r m_s)$, and

3. no operation, say $(j_u m_v)$, can exist such that

$$(j_p m_q) \prec (j_u m_v) \prec (j_r m_s) .$$

It should be noted that direct precedence relation is intransitive, non-reflexive and antisymmetric. However, when this relation is extended to have the transitive property, it is called precedence relation.

The direct-precedence relations are usually prescribed in advance because of the technological requirements. For example, a hole drilling operation must precede, or can directly precede a boring operation.

In terms of the above definitions, the prescribed ordering of M machines for a particular job j may be arranged in a single chain of direct-precedences such that

$$(jm_1) \prec\prec (jm_2) \prec\prec \ . \ . \ . \prec\prec (jm_\ell) \prec\prec \ . \ . \ . \prec\prec (jm_M).$$

For convenience, the above machine ordering for a particular job j is designated by a row vector such that

$$M_j = [jm_1 \quad jm_2 \quad . \ . \ . \ jm_\ell \ . \ . \ . \ jm_M], \quad j = 1, 2, . \ . \ . \ , J.$$

These machine ordering vectors, one for each job, may be combined in a (JxM) matrix called the machine ordering matrix denoted by $\mathcal{M}$. For example, consider a problem having two jobs to be processed on three machines. Let the jobs be j = 1, 2, and the machines be m = 1, 2, 3. The machine ordering matrix of this problem is shown below

$$\mathcal{M} = \begin{bmatrix} \mathcal{M}_1 \\ \\ \mathcal{M}_2 \end{bmatrix} = \begin{bmatrix} 1m_1 & 1m_2 & 1m_3 \\ \\ 2m_1 & 2m_2 & 2m_3 \end{bmatrix} = \begin{bmatrix} 11 & 13 & 12 \\ \\ 22 & 21 & 23 \end{bmatrix}$$

This matrix indicates that job 1 must be processed on machine 1 first, machine 3 second, and machine 2 last. However, job 2 must be performed on machine 2 first, machine 1 second, and machine 3 last. It should be noted that the machine m_1 in the element $1m_1$ is not necessarily the same as machine m_1 in the element $2m_1$. In terms of direct-precedence relations, the machine orderings for jobs 1 and 2 are

$$(11) \prec\prec (13) \prec\prec (12),$$

and

$$(22) \prec\prec (21) \prec\prec (23),$$

respectively.

Associated with each operation, (jm_ℓ), there is a processing time, t_{jm_ℓ}; that is, the time required to perform job j on a particular machine m_ℓ. The processing times of each job on the various machines are usually estimated in advance and known exactly. For convenience, the processing times for job j on all machines are designated

$$T_j = [t_{jm_1} \quad t_{jm_2} \quad . \; . \; . \quad t_{jm_\ell} \quad . \; . \; . \quad t_{jm_M}],$$

$$j = 1, 2, . \; . \; . \; . \; , J.$$

The above set of processing time, one for each job, may be combined in a (JxM) matrix referred to as the processing time matrix and denoted by T. The processing time matrix of the above example is shown below

$$T = \begin{bmatrix} T_1 \\ \\ T_2 \end{bmatrix} = \begin{bmatrix} t_{1m_1} & t_{1m_2} & t_{1m_3} \\ \\ t_{2m_1} & t_{2m_2} & t_{2m_3} \end{bmatrix} = \begin{bmatrix} 2 & 4 & 1 \\ \\ 5 & 1 & 3 \end{bmatrix}$$

The above processing time matrix indicates that to perform job 1 on machines m_1, m_2, and m_3, it requires 2, 4, and 1 units of time, respectively. Similarly, job 2 requires 5, 1, and 3 time units to be completed on machines m_1, m_2 , and m_3, respectively. It is obvious that if a job is not to be processed on a particular machine, a zero processing time can be placed in the corresponding element in the processing time matrix.

While the machine ordering specifies the order in which a particular job is processed on various machines, the job sequencing specifies the sequence in which a particular machine performs various jobs. Using the definition of direct-precedence relation, the series of operations on machine m may be designated

$$(j_1m) \prec\prec (j_2m) \prec\prec . \; . \; . \prec\prec (j_km) \prec\prec . \; . \; . \prec\prec (j_Jm)$$

Again, for convenience, the above sets of job sequencings on each machine can be arranged in a row vector

$$S_m = [j_1m \quad j_2m \; . \; . \; . \; j_km \; . \; . \; . \; j_Jm], \quad m = 1, 2, \ldots, M.$$

These job sequencings, one for each machine, may be combined in a (MxJ) matrix called the job sequencing matrix and denoted by S. For example, one of the possible job sequencing matrices of the above problem is shown below

$$S = \begin{bmatrix} S_1 \\ S_2 \\ S_3 \end{bmatrix} = \begin{bmatrix} j_1 1 & j_2 1 \\ j_1 2 & j_2 2 \\ j_1 3 & j_2 3 \end{bmatrix} = \begin{bmatrix} 11 & 21 \\ 22 & 12 \\ 13 & 23 \end{bmatrix}$$

This matrix indicates that machines 1 and 3 process jobs 1 and 2 in the sequence {1 2}; however, machine 2 performs the two jobs in the sequence {2 1}. In other words machines 1 and 3 process job 1 first and then job 2 last; and machine 2 processes job 2 first and job 1 last. It should be noted that job j_1 in the element $j_1 1$ may or may not be the same job as in the elements $j_1 2$ or $j_1 3$. In the above example, the elements $j_1 1$ and $j_1 3$ have the same job which differs from that in $j_1 2$. In terms of precedence relations, for example, $(13) \prec\prec (23)$; or, in words, operation (13) directly precedes operation (23).

The above sequencing matrix may be represented by Gantt charts shown in Figure 1.1. In this figure, each machine is represented by a horizontal bar. The hatched portion in each bar indicates the processing of a job on the corresponding machine. Job indexes are marked on the top of these bars; however, the numbers below show the starting and completion times of the various operations.

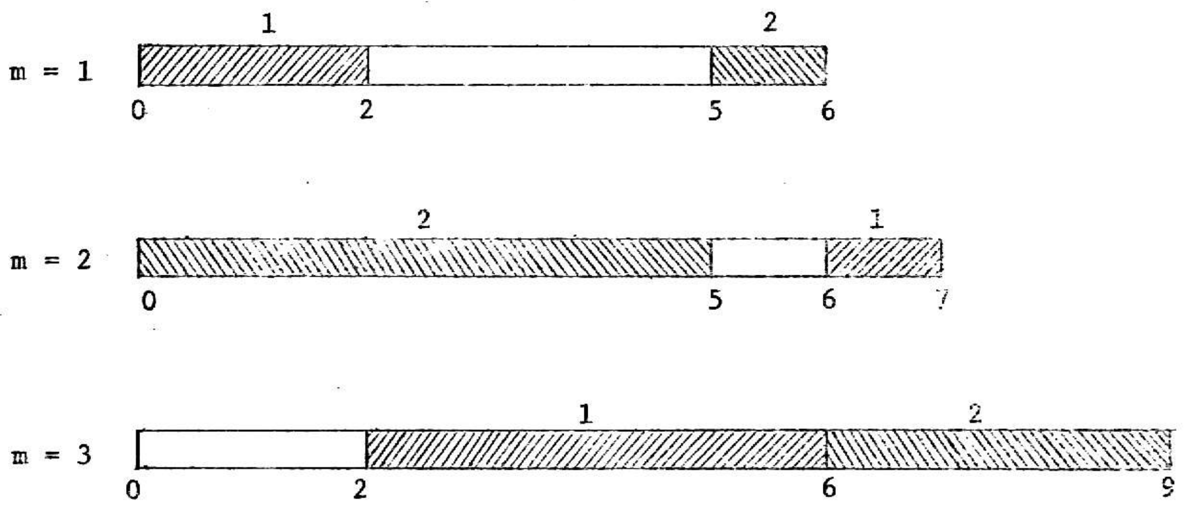

Figure 1.1

Gantt Charts Depicting the Job Sequencing Matrix

It is interesting to illustrate the precedence and direct-precedence relations discussed above, using the Gantt charts shown in Fig 1.1. For example, the precedence relation

$$(j_1 m_1) \prec (j_2 m_3)$$

or

$$(11) \prec (23)$$

holds. In words, the processing of job 1 on machine 1 precedes the processing of job 2 on machine 3.

Similarly, the direct-precedence relation on machine 1

$$(j_1 1) \prec\prec (j_2 1)$$

or

$$(11) \prec\prec (21)$$

holds. This implies that operation (11) directly precedes operation (21) on machine 1.

The direct-precedence relation for job 1

$$(1m_1) \prec\prec (1m_2)$$

or

$$(11) \prec\prec (13)$$

holds. In words, machine 1 performs job 1 immediately before it is performed on machine 3. Note that in the direct-precedence relation, the operations have either the same job or the same machine.

In scheduling problems, the machine ordering matrix is usually specified and it is required to determine the job sequencing matrix, which optimizes some measure of performance. In general, the job sequencing matrix will be referred to as a sequence. A sequence may be defined as a collection or combination of machine orderings and job sequencings. Since the elements appearing in the machine ordering matrix M are the same as those in the job sequencing matrix S , the sequence relations given by S may be inconsistent, i.e., nonfeasible with the order relations specified in M . One of the characteristics of the required sequence is that it must be consistent or compatable with the machine orderings of all jobs.

In terms of the above formulation, the scheduling problem can be stated as: given both the machine ordering and processing time matrices M and T , it is required to find the optimal sequence S with respect to a certain criterion.

The flow shop scheduling problem with which this paper is concerned, is subject to the following assumptions.

1. Assumptions regarding jobs:

   1.1 All jobs are known, ready to be processed as soon as possible, according to specified machine orderings.

   1.2 All jobs are equally important; i.e. no pre-emption, due dates or rush orders.

   1.3 Each job must be processed by a designated machine ordering.

   1.4 A job may not be processed by more than one machine at a time.

   1.5 No job is processed more than once on any machine.

   1.6 Each job, once started for processing in the shop, must be performed to completion; that is, no job cancellation.

   1.7 Each job may have to wait between machines; that is, in-process inventory is allowed.

2. Assumptions regarding machines:

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

   2.2 All machines are ideal; i.e., they operate with constant efficiency, without breakdown, never lack operator, tool, or material, and always produce acceptable products.

   2.3 Each machine can process at most one job at a time.

3. Assumptions regarding operations:

   3.1 Each operation, once started on a machine, must be performed to completion before another operation can begin on that machine; that is, no pre-emptive priorities.

   3.2 Each operation can be performed by only one machine in the shop.

4. Assumptions regarding processing times:

   4.1 The processing times are known, finite and integers.

   4.2 The processing times are independent of the sequence in which the jobs are performed.

   4.3 The processing time also includes the set-up times and the transportation times between machines.

1.2 Proposed Research

The problem of scheduling J jobs on various machines with the same machine orderings, has been studied by several investigators. However, as yet no efficient algorithm has been found for determining an optimal sequence of jobs. In this paper, the branch-and-bound algorithm will be discussed. Various bounding procedures will be analyzed mathematically and evaluated empirically.

The basic concepts of the branch-and-bound was first realized by Land and Doig [9a] which has been named by Little et. al. [11] while solving the travelling salesman problem. This approach which gives an optimal or near optimal solution after the generation of only a small subset of the possible sequences, is considered one of the combinatorial approaches to the production scheduling problem.

Ignall and Schrage [8] have applied the above concept to the two-and three-machine flow shop problem using their sophisticated lower-bound. Their computational experience involves upto nine jobs. Brown and Lomnicki [3] have extended the branch-and-bound algorithm developed by Lomnicki [12] for three machine case to arbitrary number of machines. McMahon and Burton [13] have applied this technique to the three-machine problem, giving a new procedure of obtaining the bound referred by them as composite lower-bound.

Their experience involves upto 10 jobs and 2 machines. They have concluded that the use of composite bound is more efficient. Nabeshima [15] has reported a revised lower-bound and attempted to show the superiority of this lower-bound by three sample problems.

For comparison purposes all bounding procedures are generalized to an arbitrary number of machines by assuming that all jobs are performed on each machine in the same sequence. Thus, the computational algorithm discussed in this paper will develop optimal permutation-sequence and this will be referred to as optimal sequence or optimal solution in the discussion throughout this paper. It should be noted that the permutation sequences will always be optimal for cases involving upto three machines*. However, for more than three machines, the solution is considered optimal within the set of permutation-sequences only.

As mentioned earlier the branch-and-bound technique has been first applied to flow shop scheduling problem by [8, 12]. Since then several researchers have developed various lower-bounds or bounding procedures which are vital in making the branch-and-bound technique efficient. Therefore, the success of the branch-and-bound technique depends on the quality of the lower-bounds. To the knowledge of author, no comparison among these lower-bounds has been yet made. The purpose of this paper is to present a mathematical analysis of the five promising lower-bounds referred to as bounding procedures LB I, LB II, LB III, LB IV, and LB V; to compare the quality of these bounding procedures. This comparison is based on the following: (1) the number of nodes explored; (2) the computational

*See Theorems 5-1 and 5-2, in Conway, et al; Theory of Scheduling, 1967, pp. 81-83.

efficiency. To obtain fair comparison among these procedures and to minimize the variations, considerable experiments were conducted.

In Chapter II, the basic concepts of branch-and-bound technique are discussed. A more general computational algorithm is illustrated by a sample problem. Chapter III is devoted to the mathematical analysis of various bounding procedures under consideration. The various lower-bounds computed by each of the bounding procedures are also illustrated by a sample problem. In Chapter IV, the computational experiments and their results are reported. Finally, summary of results and conclusions are provided in Chapter V.

CHAPTER II

BRANCH-AND-BOUND TECHNIQUE

A survey of the approaches used in the solution of scheduling problems reveals that all the existing approaches can be classified into four basic categories: (1) combinatorial; (2) mathematical programming; (3) queueing; and (4) simulation. This report is concerned with the branch-and-bound technique which is based on combinatorial analysis. It seems reasonable to elaborate the explanation of combinatorial mathematics before discussing the basic concepts of the branch-and-bound technique.

2.1 Combinatorial Mathematics

Combinatorial mathematics, also referred to as combinatorial analysis or combinatorics, is one of the mathematical disciplines. The formal definition of combinatorics becomes difficult due to the diversified applications of combinatorial analysis in various areas of mathematics. Ryser [17] has defined the combinatorial analysis as the mathematics which deals with the study of the arrangement of elements into sets. The elements are usually finite in number, and the arrangement is restricted by certain constraints, if any, imposed by the specific problem under consideration. The problems tackled by combinatorial analysis are classified as: (1) existence problem, and (2) enumeration problem. Existence problems are those in which the existence of solution is doubtful and the study attempts to settle this issue. On the contrary, enumeration problems have the surity of existence of solution and the objective is to find the solutions, the number of such solutions, and their classifications. Thus, each enumeration problem is an extension of existence problem.

According to the definitions of combinatorial problems cited above, it is obvious that the scheduling problem is an enumeration problem. The combinatorial approach for solving the scheduling problem represents quasi-enumeration techniques. By applying many reliable heuristics, this approach has succeeded in arriving at the subset of optimal schedules through a shortest path. This subset is comparatively much smaller than the complete enumeration. The efficiency of the combinatorial techniques depends on how effectively enumeration is curtailed. A number of techniques developed within the concept of combinatorial analysis for solving scheduling problems can be listed as switch-and-check, branch-and-bound and bound-and-resolve technique.

2.2 Basic Concepts of Branch-and-Bound Technique

The branch-and-bound technique is an intelligently designed search to obtain the subset of optimal solutions from a larger set of feasible sequences. It has been pointed out in literature that for any particular scheduling problem, the number of optimal schedules is much smaller than the number of all possible feasible sequences. This fact revealed that even a technique discarding only non-feasible sequences will not be efficient in solving larger size problems. By using certain processes, the search is directed towards an optimal solution in a finite number of steps. The branch-and-bound algorithm described in this chapter assures that it will arrive at an optimal solution and it may be possible to do so in less than complete enumeration.

The basic philosophy behind this technique is the partitioning of the set of feasible sequences into smaller and smaller subsets. A lower-bound is calculated for the partial or complete sequence within each subset.

The objective of this technique is to find the optimal solution with less computational effort involved. The search should be systematically progressed through branching and bounding processes which may be easily discussed by using a scheduling tree.

The scheduling tree consists of nodes, each representing a partial or complete sequence, $\{j_1\ j_2\ .\ .\ .\ j_L\}$, where $L \leq J$. The scheduling tree starts with a node "ALL" which represents J unscheduled jobs at level 0. At level 1, the scheduling tree is initialized by J nodes, each of which consists of a partial sequence having one job, $\{j_1\}$. Each of these J nodes can be branched into J-1 nodes at level 2, each consisting of a partial sequence $\{j_1\ j_2\}$. In general, at any level, L, there are J-L+1 new nodes emanating from each node at the preceding level. Each of these new nodes consists of a partial sequence having L jobs, $\{j_1\ j_2\ .\ .\ .\ j_L\}$. Note that the number of jobs in each node at level L is equal to L. As one moves down the tree, the index of levels increases by one; and the number of nodes branched from a node, decreases by one than that of the preceding level. The tree ends at level J with a number of nodes, each having a complete sequence of J jobs. Any of these complete sequences can be regarded as a solution to the scheduling problem. It should be pointed out that when all nodes at each level are generated, the maximum number of nodes in such a scheduling tree is $J + J(J-1) + J(J-1)(J-2) + . . . + J!$ which increases very rapidly as the number of jobs increases.

The process of generating a new set of nodes at level L, each having a partial sequence $\{j_1\ j_2\ .\ .\ .\ j_L\}$, from a node at the preceding level, L-1, is referred to as Branching process. Further, this node from which the branching takes place is called active node. The nodes are generated by placing one of the unscheduled jobs as a candidate for the next sequence-position in the partial sequence of the active node. The branching characteristic of the branch-and-bound algorithm, used in this report, guarantees that if this algorithm is applied to a flow shop scheduling problem, an optimal solution will eventually be obtained. It merely promises the optimal solution by exploring all nodes of the scheduling tree.

Reduction in the generation of nodes at each level can be effectively achieved by bounding process. In this process a lower-bound for each node is computed and then according to the branching strategy, the active node is picked up for further branching. The various lower-bounds developed by several researchers will be analyzed in detail in Chapter III.

The lower-bound for a node is a lower limit on the schedule time which implies that any solution emanating from this node can not have schedule time less than the value of its lower-bound. In other words, all complete sequences resulted from a particular node under consideration will have schedule times either equal to or greater than the value of the lower-bound for that node. This concept helps select the promising node for branching to be carried down to the next level. As usual, the efficiency of the branch-and-bound algorithm depends on the quality of the bounding process, since this process has a characteristic of recognizing an optimal solution prior to complete enumeration. It is logical to look for a high lower-bound than a low lower-bound for the same node, computed by different lower-bound formulas developed so far. The higher the value of the lower-bound, the more powerful is the bound on the objective function. Five of the more promising possibilities for these bounds are developed in

[3, 8, 12, 13, 15] and discussed in Chapter III.

It should be pointed out that the lower-bounds which consider the idle times created by the scheduled and unscheduled jobs, usually yield higher and more realistic lower-bounds than those which consider the idle times created by the scheduled jobs only. The lower-bounds which do not consider idle times among the unscheduled jobs, assume that there is no overlapping or conflict among these jobs. The bounding process is more strong in recognizing direction of an optimal solution at higher levels down the tree than lower levels up the tree. This is because there are fewer jobs in the nodes at lower levels than those at higher levels. The larger the number of jobs in a partial sequence, the more closer will be the lower-bound to the minimum schedule time.

As one moves down the tree, the lower-bound will never decrease along the same branch of the tree. By moving down a level, an unscheduled job is added to the previous partial sequence and while computing a lower-bound for this newly created node, the idle time created by the last scheduled job is also accounted. Thus, the lower-bound of a node will be either equal to or greater than the lower-bound of the node from which branching took place at the preceding level, depending upon the idle time. However, it will never decrease, since the idle times can never take negative values.

The branching and bounding processes always yield a solution which may or may not be an optimal. However, in order to guarantee optimality a back-tracking process must be imbedded in the branch-and-bound technique. The function of the back-tracking process is to move upwards on the same branch which has led to the previously obtained solution. Then, by using the branching and bounding processes, the unexplored node(s) at the immediately preceding level and having lower-bound less than the value of the previously obtained solution (hereafter referred to as updated solution), is investigated. In moving downwards until the last level, another solution may be obtained which again may or may not be better than the previous one. This updated solution is considered optimal when there is no unexplored node with a lower-bound less than that of the previous solution. The branching, bounding and back-tracking processes which are imbedded in the computational branch-and-bound algorithm stated in Section 2.3, will be illustrated by a sample problem.

2.3 A Computational Algorithm

The branch-and-bound algorithm described in this section is completely general in the sense that it consists of various phases to compute the lower-bounds; and to break the ties between the lower-bounds, if any. This algorithm has been designed to yield an optimal solution which has been guaranteed by back-tracking process.

The computational algorithm may be stated as follows:

Step 1: Set level index $L = 1$, and schedule time $T = +\infty$.

Step 2: Check L:

2.1. If $L < J$, go to step 3.

2.2. If $L = J$, check the minimum lower-bound at level $J - 1$, ${}^{J-1}B$:

2.2.1. If ${}^{J-1}B < T$, set $T = {}^{J-1}B$ and $L = L - 1$. Go to step 7.

2.2.2. If ${}^{J-1}B \geq T$, set $L = L - 1$ and go to step 7.

Step 3: Compute the lower-bounds at level L, for each node n, ${}^{L}B^{n}$, by a particular bounding process.

Step 4: Find the unexplored node(s) which has the minimum lower-bound at level L, ${}^{L}B$, such that

$$ {}^{L}B = \min_{n} [{}^{L}B^{n}] , $$

and compare ${}^{L}B$:

4.1. If ${}^{L}B < T$, go to step 5.

4.2. If ${}^{L}B \geq T$, go to step 7.

Step 5: Branch from an unexplored node with the minimum lower-bound:

5.1. If a tie exists, break it by a particular rule.

5.2. If a tie does not exist, branch from that node.

Step 6: Set L = L + 1, and go to step 2.

Step 7: Back-track along the same branch of the scheduling tree by setting L = L - 1. Compare the lower-bounds for all unexplored nodes at this level:

7.1. If there exist one or more nodes such that ${}^{L}B^{n} < T$, go to step 4.

7.2. If for all unexplored nodes, ${}^{L}B^{n} \geq T$, go to step 8.

Step 8: Check for an optimal solution:

8.1. If L > 1, go to step 7.

8.2. If L = 1, T is an optimal schedule time.

The lower-bound, in step 3 of the algorithm described above, can be computed by one of the five bounding processes summarized in Table 2.1. According to step 5 of the algorithm, a tie, if exists, can be resolved by one of the three rules: the Left Hand Rule (LHR) breaks a tie by selecting the farthest-left node from the tie-set at that level in the scheduling tree. Similarly, the Right Hand Rule (RHR) picks the farthest-right node from the tie-set for branching. If the node is selected by random (RDM) from the tie-set, the tie is said to be resolved by random. It should be pointed out that the (LHR) has been used in this research for resolving the tie.

Table 2-1 Various Bounding Processes

Concept	Bounding Procedures	Discussed in Section	Investigator	Reference
Machine Based	LB I	3.1	Brown & Lomnicki	[3,12]
Machine Based	LB II	3.2	Ignall & Schrage	[8]
Job Based	LB III	3.3	McMahon & Burton	[13]
Composite Based	LB IV	3.4	McMahon & Burton	[13]
Machine Based	LB V	3.5	Nabeshima	[15]

2.4 Sample Problem

The computational algorithm described in the preceding section will be illustrated by a sample problem. The problem consists of six jobs to be processed on each of three machines. The machine ordering and processing time matrices are given below:

$$M = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \\ 51 & 52 & 53 \\ 61 & 62 & 63 \end{bmatrix}, \qquad T = \begin{bmatrix} 6 & 7 & 3 \\ 12 & 2 & 3 \\ 4 & 6 & 8 \\ 3 & 11 & 7 \\ 6 & 8 & 10 \\ 2 & 14 & 12 \end{bmatrix}$$

The purpose of selecting this problem is to make clear to the reader how branching, bounding and back-tracking processes are imbedded in the branch-and-bound technique. The emphasis in this chapter is placed on how the branch-and-bound works, not on how the lower-bound is computed. The computation of the various lower bounds is left to be analyzed in Chapter III. The values of the lower-bounds are computed according to bounding procedure LBI. For convenience, LBI is stated mathematically as follows:

The lower-bound at level L for each node n, ${}^{L}B^{n}$, is given by

$${}^{L}B^{n} = \max_{m} [{}^{L}B^{n}_{m}] ,$$

where ${}^{L}B^{n}_{m}$ is the bound on machine m, at level L, for node n and is computed such that

$${}^{L}B^{n}_{m} = {}^{L}C^{n}_{m} + \sum_{k=L+1}^{J} t_{j_k m} + \min_{j_k \neq n} \left[\sum_{m'=m+1}^{M} t_{j_k m'} \right] ,$$

$$m = 1, 2, \ldots , M-1$$

and

$${}^{L}B^{n}_{M} = {}^{L}C^{n}_{M} + \sum_{k=L+1}^{J} t_{j_k M}$$

where ${}^{L}C^{n}_{m}$ is the completion time of scheduled jobs in node n at level L on machine m. For convenience, the elements from which lower-bounds have been obtained, are displayed in Table 2.2. However, this lower-bound along with others will be analyzed mathematically and illustrated by another sample problem in Chapter III.

In solving this sample problem, a scheduling tree is initialized with a node "ALL" which may be considered as level zero. In order to follow the solution easily, Figure 2.1 should be followed throughout all the steps.

Step 1. Set level L equal to 1 and the initial schedule time T_o equal to ∞. Level 1 is initialized by generating the nodes having partial sequence $\{j_1\}$. Thus, there are J - L + 1 or 6 nodes with partial sequences {1}, {2}, {3}, {4}, {5} and {6}.

Step 2. Check the level L. Since L is less than J, i.e., 1 is less than 6, go to step 3.

Step 3. The lower-bounds are computed for each node at level 1 by bounding procedure LBI such that

Node	(1)	(2)	(3)	(4)	(5)	(6)
Lower-bound	57	63	55	57	57	59

Step 4. Search for the minimum unexplored node(s) at level L. The minimum value of the lower-bound at level 1, ${}^1B$, is equal to 55 for node (3). According to step 4.1, go to step 5, since ${}^1B$ is less than T_o, i.e., 55 is less than ∞.

Step 5. The tie does not exist for the minimum lower-bound. Therefore, the selected node, (3), is branched to generate the nodes (31), (32), (34), (35) and (36) at level 2.

Step 6. Set level L equal to 2 and go to step 2.

Step 2. Check level L. Since level L is less than J, i.e., 2 is less than 6, go to step3.

Step 3. The lower-bounds for each node at level 2 are computed by bounding procedure LBI such that

Node	(31)	(32)	(34)	(35)	(36)
Lower-bound	55	61	56	55	59

Step 4. The minimum lower-bound at level 2, ${}^2B$, is 55 for nodes (31) and (35).

Step 5. The tie is resolved in favor of node (31) according to the LHR in step 5.1. This active node, (31), is then branched to form new nodes (312), (314), (315) and (316) at level 3.

Step 6. Set level L equal to 3 and go to step 2.

Step 2. Check level L. As L or 3 is less than J or 6, go to step 3.

Step 3. The lower-bounds are computed for all nodes at level 3 by bounding procedure LBI such that

Node	(312)	(314)	(315)	(316)
Lower-bound	64	60	57	63

Step 4. The node (315) has the minimum lower-bound at level 3, ${}^3B$, which is equal to 57. According to step 4.1, go to step 5 since ${}^3B$ or 57 is less than T_o or ∞.

Step 5. The active node (315) is branched to create new set of nodes (3152), (3154) and (3156) at level 4.

Step 6. Set level L equal to 4 and go to step 2.

Step 2. Check level L. According to step 2.1 go to step 3 since L or 4 is less than J or 6.

Step 3. At level 4, the lower-bounds are computed for all nodes by bounding procedure LBI such that

Node	(3152)	(3154)	(3156)
Lower-bound	62	58	61

Step 4. At level 4, the node (3154) is picked up for branching since

it has the minimum lower-bound. According to step 4.1, go to step 5, since ${}^4B$ or 58 is less than T_o or ∞.

Step 5. The active node at this level, (3154), is branched to generate new nodes (31542) and (31546) at level 5.

Step 6. Set level L equal to 5 and go to step 2.

Step 2. Check level index L. Since L is less than J or 5 is less than 6, go to step 3.

Step 3. The lower-bounds are computed for all nodes at level 5 by bounding procedure LBI such that

Node	(31542)	(31546)
Lower-bound	64	65

Step 4. The minimum lower-bound at level 5, ${}^5B$, is 64 for node (31542). Following step 4.1, go to step 5 because ${}^5B$ or 64 is less than T_o or ∞.

Step 5. The active node, (31542), at level 5 is branched to generate a node with complete sequence {3 1 5 4 2 6} at level 6.

Step 6. Set level L equal to 6 and go to step 2.

Step 2. Check level L. As level L or 6 is equal to J or 6, according to step 2.2, the minimum lower-bound at level 5 or ${}^5B$ is compared with the initial schedule time, T_o or ∞. Since ${}^5B$ or 64 is less than T_o or ∞, the initial schedule time is updated according to step 2.2.1 by setting $T_1 = {}^5B = 64$. Set L = L - 1 or 5 and go to step 7.

Step 7. Back-track along the same branch of the scheduling tree if there is a possibility for a better solution. Set level index L = L - 1 or 4 and compare the lower-bounds of all unexplored nodes with the updated schedule time, T_1. There are two nodes (3152) and (3156) having lower-bounds

62 and 61, respectively, which are less than T_1 or 64. According to step 7.1, go to step 4.

Step 4. At level 4, the minimum lower-bound, ^4B, is 61 for node (3156). According to step 4.1, go to step 5 since ^4B or 61 is less than T_1 or 64.

Step 5. The active node (3156) at level 4, is branched according to step 5.2 to create level 5 having nodes (31562) and (31564).

Step 6. Set level L equal to 5 and go to step 2.

Step 2. Check level L. Since L or 5 is less than J or 6, go to step 3.

Step 3. By bounding procedure LBI, the lower-bounds are computed for each node at level 5, such that

Node	(31562)	(31564)
Lower-bound	61	61

Step 4. At level 5, the minimum lower-bound, ^5B, is 61 for nodes (31562) and (31564). According to step 4.1, since ^5B or 61 is less than T_1 or 64, go to step 5.

Step 5. According to the (LHR) in step 5.1, the tie at level 5 is resolved in favor of node (31562). This node is then branched according to step 5.2 to generate a node having complete sequence {3 1 5 6 2 4}.

Step 6. Set level, L, equal to 6 and go to step 2.

Step 2. Check level L. The level, L or 6, is equal to J or 6; the minimum lower bound at level 5, ^5B, is compared with the updated schedule time, T_1, according to step 2.2. Since B^5 or 61 is less than T_1 or 64, the updated schedule time is modified by setting $T_2 = B^5 = 61$. Set level L equal to L - 1 or 5 and go to step 7.

Similarily, the back-tracking process is carried along the same branch of the scheduling tree to find the unexplored node(s) which has lower-bound less than the updated schedule time, T_2. At level 3, the active node (314) which has lower-bound equal to 60, is branched to generate nodes (3142), (3145) and (3146) having lower-bounds 62, 61 and 67, respectively. Since any sequence derived from node (3145) can never have schedule time better than 61, the branching is terminated in this direction according to step 4.2. There is no unexplored node at level 3 which has lower-bound less than T_2 or 61. At level 2, the node (35) is picked up for branching which yields a sequence {3 5 1 4 2 6} having schedule time of 6.4. Back-tracking process selects another node (354) at level 3, for further branching. The branching is terminated at level 6 with no improved solution. The node (3542) at level 4, is brauched to yield a sequence {3 5 4 2 6 1} with schedule time of 60 which is less than the updated schedule time, T_2 or 61. Now, the updated solution is modified such that $T_3 = 60$. The back-tracking along the same branch of the scheduling tree finds a node (356) at level 3 which is still unexplored and has lower-bound less than T_3, i.e., ${}^{3}B^{356} < T_3$ or $57 < 60$. The branching from this node leads to a sequence {3 5 6 1 2 4} at level 6, having schedule time of 59. Since 59 is less than T_3 or 60 the updated schedule time is modified such that $T_4 = 59$. Back-tracking picks another node (3562) at level 4 which generates a sequence {3 5 6 2 4 1} with schedule time of 57. Again, the updated schedule time is modified according to step 2.2.1 such that $T_5 = 57$. The back-tracking reveals that there exists only one node (34) at level 2 which has lower-bound less than T_5 or that $56 < 57$. The branching is carried until level 3 where the nodes (3412), (3415) and (3416) having lower-bounds 62, 61 and 67, respectively are generated. The branching in this direction is terminated, since the minimum lower-bound at this level is greater than the updated schedule time, T_5 or 57. The node (342) at level 3 is therefore branched but the branching is terminated at level 4. Now there is no unexplored node at any level which has lower-bound less than T_5 or 57. Hence, the schedule time, T_5 or 57, is the minimum schedule time and the corresponding sequence {3 5 6 2 4 1} is optimal. The problem is solved by exploring 58 nodes. It is interesting to know that 'paper and pencil' solution of this problem took around 5 hours while the author worked at normal pace.

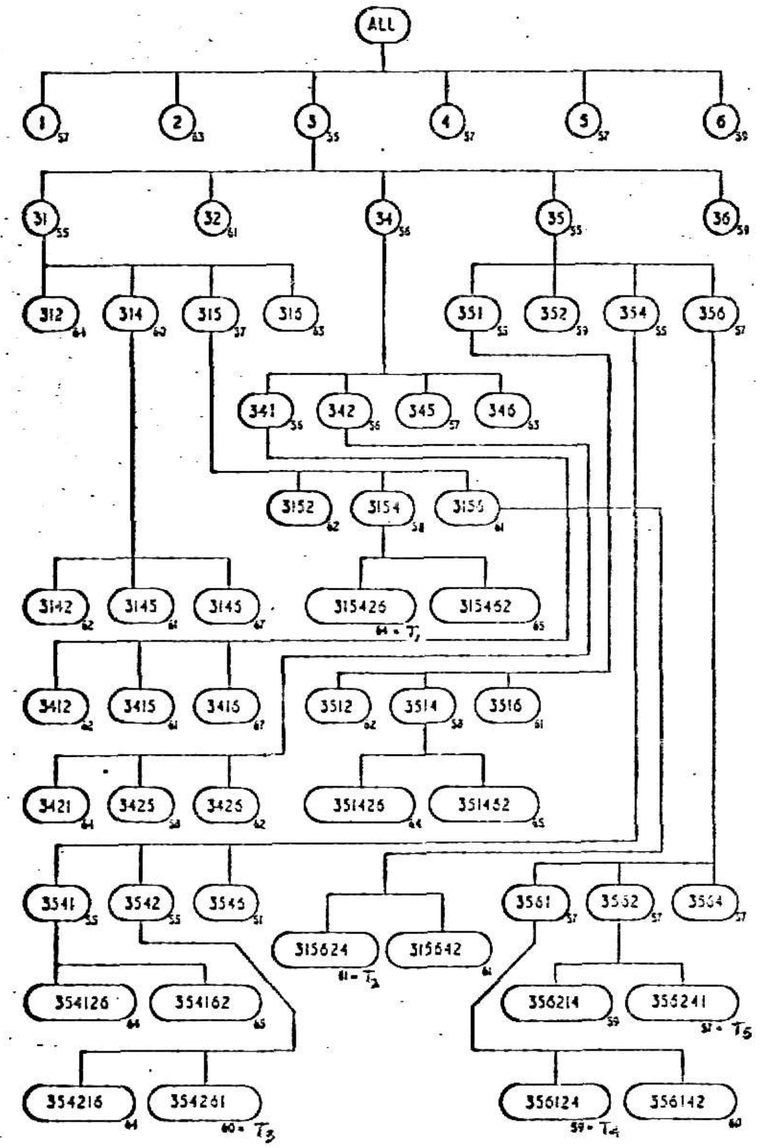

Figure 2.1

The Scheduling Tree

Table 2.2

Computation of Lower-Bounds

No. of Back- -track	Level	Node	$L_{C_m}^n$			$\sum_{k=L+1}^{J} t_{J_k m}$			$\min_{J_k \neq n}\left(\sum_{m'=m+1}^{M} t_{J_k m'}\right)$			$L_{B_m}^n$			L_B^n	T_i
			Machines			Machines			Machines			Machines				
			1	2	3	1	2	3	1	2	3	1	2	3		
	1	(1)	6	13	16	27	41	40	5	3	0	38	57	56	57	
		(2)	12	14	17	21	46	40	10	3	0	43	63	57	63*	
		(3)	4	10	18	29	42	35	5	3	0	38	55	53	55	
		(4)	3	14	21	30	37	36	5	3	0	38	54	57	57	
		(5)	6	14	24	27	40	33	5	3	0	38	57	57	57	
		(6)	2	16	28	31	34	31	5	3	0	38	53	59	59	
	2	(31)	10	17	21	23	35	32	5	3	0	38	55	53	55*	
		(32)	16	18	21	17	40	32	10	3	0	43	61	53	61	
		(34)	7	21	28	26	31	28	5	3	0	38	55	56	56*	
		(35)	10	18	28	23	34	25	5	3	0	38	55	53	55	
		(36)	6	24	36	27	28	23	5	3	0	38	55	59	59	
	3	(312)	22	24	27	11	33	29	18	7	0	51	64	56	64	
		(314)	13	28	35	20	24	25	5	3	0	38	55	60	60*	
		(315)	16	25	35	17	27	22	5	3	0	38	55	57	57	
		(316)	12	31	43	21	21	20	5	3	0	38	55	63	63	
	4	(3152)	28	30	38	5	25	19	18	7	0	51	62	57	62*	
		(3154)	19	36	43	14	16	15	5	3	0	38	55	58	58	
		(3156)	18	39	51	15	13	10	5	3	0	38	55	61	61	
	5	(31542)	31	38	46	2	14	12	26	12	0	59	64	58	64*	
		(31546)	21	50	62	12	2	3	5	3	0	38	55	65	65	
1	6	(315426)	33	52	64	0	0	0	0	0	0	33	52	64	64*	T_1

Table 2.2 (continued)

Computation of Lower-Bounds

No. of Back--track	Level	Node	$L_{C_m}^n$			$\sum_{k=L+1}^{J} t_{j_k m}$			$\min_{j_k \neq n} \left(\sum_{m'=m+1}^{M} t_{j_k m'} \right)$			$L_{B_m}^n$			L_B^n	T_i
			Machines			Machines			Machines			Machines				
			1	2	3	1	2	3	1	2	3	1	2	3		
	5	(31562)	30	41	54	3	11	7	18	7	0	51	59	61	61*	
		(31564)	21	50	58	12	2	3	5	3	0	38	55	61	51*	
2	6	(315624)	33	52	61	0	0	0	0	0	0	33	52	61	61*	T_2
	4	(3142)	25	30	38	8	22	22	18	10	0	51	62	60	62	
		(3145)	19	36	46	14	16	15	5	3	0	38	55	61	61*	
		(3146)	15	42	54	18	10	13	5	3	0	38	55	67	67	
	3	(351)	16	25	31	17	27	22	5	3	0	38	55	53	55*	
		(352)	22	24	31	11	32	22	10	3	0	43	59	53	59	
		(354)	13	29	36	20	23	18	5	3	0	38	55	54	55*	
		(356)	12	32	44	21	20	13	5	3	0	38	55	57	57	
	4	(3512)	28	30	34	5	25	19	18	7	0	51	62	53	62	
		(3514)	19	36	43	14	16	15	5	3	0	38	55	58	58*	
		(3516)	18	39	51	15	13	10	5	3	0	38	55	61	61	
	5	(35142)	31	38	46	2	14	12	26	12	0	59	64	58	64*	
		(35146)	21	50	62	12	2	3	5	3	0	38	55	65	65	
	4	(3541)	19	36	39	14	16	15	5	3	0	38	55	54	55*	
		(3542)	25	31	39	8	21	15	10	3	0	43	55	54	55*	
		(3546)	15	43	55	18	9	6	5	3	0	38	55	61	61	
	5	(35412)	31	38	42	2	14	12	26	12	0	59	64	54	64*	
		(35416)	21	50	62	12	2	3	5	3	0	38	55	65	65	

Table 2.2 (continued)

Computation of Lower-Bounds

No. of Back--track	Level	Node	$L_{C_m}^n$			$\sum_{k=L+1}^{J} t_{j_k m}$			$\min_{j_k \neq n}\left(\sum_{m'=m+1}^{M} t_{j_k m'}\right)$			$L_{B_m}^n$			L_B^n	T_i
			Machines			Machines			Machines			Machines				
			1	2	3	1	2	3	1	2	3	1	2	3		
	5	(35421)	31	38	42	2	14	12	26	12	0	59	64	54	64*	
		(35426)	27	45	57	6	7	3	10	3	0	43	55	60	60	
3	6	(354261)	33	52	60	0	0	0	0	0	0	33	52	60	60*	T_3
	4	(3561)	18	39	47	15	13	10	5	3	0	38	55	57	57*	
		(3562)	24	34	47	9	18	10	10	3	0	43	55	57	57*	
		(3564)	15	43	51	18	9	6	5	3	0	38	55	57	57*	
	5	(35612)	30	41	50	3	11	7	18	7	0	51	59	57	59*	
		(35614)	21	50	57	12	2	3	5	3	0	38	55	60	60	
4	6	(356124)	33	52	59	0	0	0	0	0	0	33	52	59	59*	T_4
	5	(35621)	30	41	50	3	11	7	18	7	0	51	59	57	59*	
		(35624)	27	45	54	6	7	3	10	3	0	43	55	57	57	
5	6	(356241)	33	52	57	0	0	0	0	0	0	33	52	57	57*	T_5
	3	(341)	13	28	31	20	24	25	5	3	0	38	55	56	56*	
		(342)	19	23	31	14	29	25	10	3	0	43	55	56	56*	
		(345)	13	29	39	20	23	18	5	3	0	38	55	57	57	
		(346)	9	35	47	24	17	16	5	3	0	38	55	63	63	
	4	(3412)	25	30	34	8	22	22	18	10	0	51	62	56	62*	
		(3415)	19	36	46	14	16	15	5	3	0	38	55	61	61	
		(3416)	15	42	54	18	10	13	5	3	0	38	55	67	67	

Table 2.2 (continued)

Computation of Lower-Bounds

No. of Back- -track	Level	Node	$L_{C_m}^n$			$\sum_{k=L+1}^{J} t_{j_k m}$			$\min_{j_k \notin n} \left\{ \sum_{m'=m+1}^{M} t_{j_k m'} \right\}$			$L_{B_m}^n$			L_B^n	T_i
			Machines			Machines			Machines			Machines				
			1	2	3	1	2	3	1	2	3	1	2	3		
	4	(3421)	25	32	35	8	22	22	18	10	0	51	64	57	64	
		(3425)	25	33	43	8	21	15	10	3	0	43	57	58	58*	
		(3426)	21	37	49	12	15	13	10	3	0	43	55	62	62	

*indicates minimum lower-bound at that level.

CHAPTER III

LOWER-BOUNDS ON SCHEDULE TIME

As mentioned in the previous chapter, the bounding process reduces the number of explored nodes, and, thus decreases considerable amount of computation time involved. This chapter is devoted to the analytical study of the various lower-bounds which have been developed by several investigators. To illustrate the various bounding procedures, a sample problem will be solved in Section 3.6 employing the branch-and-bound algorithm discussed in Chapter II. The values of the lower-bounds for all nodes, at each level, will be summarized in the solution of this sample problem. However, the lower-bounds for only two nodes at different levels will be computed following the mathematical analysis of each bounding procedure. Since the bound on the schedule time is a function of the processing times of the jobs on the machines, the computation of the lower-bounds by the various bounding procedures requires the data in the processing time matrix. Thus, for convenience, the processing time matrix of the sample problem solved in Section 3.6 is reproduced below:

$$T = \begin{bmatrix} 9 & 13 & 6 \\ 7 & 7 & 20 \\ 6 & 4 & 8 \\ 8 & 3 & 10 \\ 20 & 7 & 2 \\ 10 & 2 & 13 \end{bmatrix}$$

The following notation is considered to discuss those bounding procedures:

L	level index of the scheduling tree, $L = 1, 2, \ldots, J$
n	node at any level, L, consisting of a partial sequence of scheduled jobs, $\{j_1 \, j_2 \ldots j_L\}$
$\bar{n}$	a set of unscheduled jobs, $j_{L+1}, j_{L+2}, \ldots, j_J$
${}^{L}C_m^n$	completion time on machine m, at level L, and for node n. It is also the earliest possible starting time for the first unscheduled job, j_{L+1}, if there is no conflict on preceding machines.
${}^{L}B_m^n$	bound on the schedule time, on machine m, at level L, and for node n.
${}^{L}B^n$	lower-bound on the schedule time, at level L, for node n.
${}^{L}B$	minimum lower-bound on the schedule time at level L.

3.1 Bounding Procedure LB I

This bounding procedure has been developed by Lomnicki [12] for the 3-machine flow shop scheduling problem and then extended to an arbitrary number of machines by Brown and Lomnicki [3]. The extension is based on an assumption that all jobs are processed in the same sequence on each of m machines. This procedure is also referred to as machine-based bound [13], since it estimates a bound by finding a schedule time on each machine.

Mathematically, the bound, ${}^{L}B^{n}_{m}$, for each node n, on machine m, at level L, is given as

$$ {}^{L}B^{n}_{m} = {}^{L}C^{n}_{m} + \sum_{k=L+1}^{J} t_{j_k m} + \min_{\substack{j_k \\ j_k \varepsilon \bar{n}}} \left(\sum_{m'=m+1}^{M} t_{j_k m'} \right) , $$

$$ k = 1, 2, \ldots, J, $$

$$ m = 1, 2, \ldots, M-1, \qquad (1) $$

and

$$ {}^{L}B^{n}_{M} = {}^{L}C^{n}_{M} + \sum_{k=L+1}^{J} t_{j_k M} \qquad (2) $$

The completion time on machine m, at level L and for node n which has the partial sequence $\{j_1\ j_2\ \ldots\ j_L\}$ is such that

$$ {}^{L}C^{n}_{m} = \max \left({}^{L}C^{j_1 j_2 \ldots j_L}_{m-1} , {}^{L-1}C^{j_1 j_2 \ldots j_{L-1}}_{m} \right) + t_{j_L m} , $$

$$ m = 1, 2, \ldots, M, \qquad (3) $$

where

$$ {}^{L}C^{j_1 j_2 \ldots j_L}_{m-1} = 0, \qquad m = 1 $$

and

$$ {}^{L-1}C^{j_1 j_2 \ldots j_{L-1}}_{m} = 0, \qquad L = 1 $$

The lower-bound at level L, for any node n is given by

$$^{L}B^{n} = \max \left\{ {}^{L}B^{n}_{1},\ {}^{L}B^{n}_{2},\ .\ .\ .\ ,\ {}^{L}B^{n}_{M} \right\}$$

$$= \max_{m} \left\{ {}^{L}B^{n}_{m} \right\}$$

In words, Equation (1) consists of the sum of three terms. The first term, ${}^{L}C^{n}_{m}$, is the measure of time elapsed from the start of processing the job in the first sequence-position, j_1, on machine 1, until the completion of the job in the last sequence-position, j_L, of the partial sequence, $\{j_1\ j_2\ .\ .\ .\ j_L\}$, on machine m. It should be noted that this term includes all idle times, if any, on machine m, for all jobs included in the partial sequence of node n.

The term, ${}^{L}C^{n}_{m}$, is calculated by employing Equation (3), the elements of which can be explained as follows:

${}^{L}C^{j_1\ j_2\ .\ .\ .\ j_L}_{m-1}$	the completion time of job j_L in the partial sequence of node n, on machine m-1. In other words, this is the earliest possible time at which job j_L is ready to be processed on machine m.
${}^{L-1}C^{j_1\ j_2\ .\ .\ .\ j_{L-1}}_{m}$	the completion time of job j_{L-1}, in the partial sequence of node n, on machine m. In other words, this gives the earliest possible time at which machine m will be ready to process job j_L.

The maximum of these two periods will give the starting time of job j_L on machine m.

The second term in Equation (1), is the total processing time for the remaining unscheduled jobs, $j_{L+1}, j_{L+2}, \ldots, j_J$; on machine m, regardless of overlapping of these jobs on that machine. This assumes that there is no idle time on machine m for these unscheduled jobs.

The third term in Equation (1), is the sum of processing times of the last job in the complete sequence on the succeeding machines, m+1, m+2, . . ., M. Since the job which will fill up the last sequence-position of the required complete sequence, is not yet determined, a job with the minimum total processing time on all succeeding machines is selected to be in the last sequence-position so as to minimize ${}^{L}B_m^n$. The jobs in the partial sequence have already been scheduled; hence this last job is chosen from unscheduled jobs.

The following computations illustrate the bounding procedure LBI discussed above. For space limitation, the computation of the lower-bounds for only node (3) at level 1 and node (34) at level 2 will be shown below.

At level 1 and for the node n which consists of partial sequence $\{j_1\}$ or {3}, the completion time on each machine is computed such that

$$ {}^{1}C_1^3 = \max \left[{}^{1}C_0^3,\ {}^{0}C_1^0 \right] + t_{31}, $$

$$ {}^{1}C_2^3 = \max \left[{}^{1}C_1^3,\ {}^{0}C_2^0 \right] + t_{32}, $$

and

$$ {}^{1}C_{3}^{3} = \max\left({}^{1}C_{2}^{3},\ {}^{0}C_{3}^{0}\right) + t_{33}, $$

or

$$ {}^{1}C_{1}^{3} = \max\left(0,\quad 0\right) + 6 = 6, $$

$$ {}^{1}C_{2}^{3} = \max\left(6,\quad 0\right) + 4 = 10, $$

and

$$ {}^{1}C_{3}^{3} = \max\left(10,\quad 0\right) + 8 = 18. $$

The bound for this node is computed oneach machine such that

$$ {}^{1}B_{1}^{3} = {}^{1}C_{1}^{3} + \sum_{k=2}^{6} t_{j_k 1} + \min_{\substack{j_k \\ j_k \neq 3}} \left(\sum_{m'=2}^{3} t_{j_k m'}\right), $$

$$ {}^{1}B_{2}^{3} = {}^{1}C_{2}^{3} + \sum_{k=2}^{6} t_{j_k 2} + \min_{\substack{j_k \\ j_k \neq 3}} \left(t_{j_k 3}\right), $$

and

$$ {}^{1}B_{3}^{3} = {}^{1}C_{3}^{3} + \sum_{k=2}^{6} t_{j_k 3}, $$

or

$$ {}^{1}B_{1}^{3} = 6 + (9+7+8+20+10) + \min\,[13+6,\ 7+20,\ 3+10,\ 7+2,\ 2+13] $$

$$ = 6 + 54 + 9 $$

$$ = 69, $$

$$ {}^{1}B_{2}^{3} = 10 + (13+7+3+7+2) + \min\,[6,20,10,2,13] $$

$$ = 10 + 32 + 2 $$

$$ = 44, $$

and

$$ {}^{1}B_{3}^{3} = 18 + (6+20+10+2+13) $$

$$ = 18 + 51 $$

$$ = 69. $$

Thus, the lower-bound for this node is

$$ {}^{1}B^{3} = \max \left({}^{1}B_{1}^{3},\ {}^{1}B_{2}^{3},\ {}^{1}B_{3}^{3} \right) $$

$$ = \max \left(69,\ 44,\ 69 \right) $$

$$ = 69. $$

Similarly, the lower-bounds for nodes (1), (2), (4), (5) and (6) is found to be 81, 73, 70, 86 and 71, respectively.

At level 2, the completion times for node having partial sequence $\{j_1\ j_2\}$ or {3 4} are computed such that

$$ {}^{2}C_{1}^{34} = \max \left({}^{2}C_{0}^{34},\ {}^{1}C_{1}^{3} \right) + t_{41}, $$

$$ {}^{2}C_{2}^{34} = \max \left({}^{2}C_{1}^{34},\ {}^{1}C_{2}^{3} \right) + t_{42}, $$

and

$$ {}^{2}C_3^{34} = \max\left\{ {}^{2}C_2^{34},\ {}^{1}C_3^{3} \right\} + t_{43}, $$

or

$$ {}^{2}C_1^{34} = \max\left\{ 0,\quad 6 \right\} + 8 = 14, $$

$$ {}^{2}C_2^{34} = \max\left\{ 14,\quad 10 \right\} + 3 = 17, $$

and

$$ {}^{2}C_3^{34} = \max\left\{ 17,\quad 18 \right\} + 10 = 28. $$

At this level, the bounds for this node on each machine are computed such that

$$ {}^{2}B_1^{34} = {}^{2}C_1^{34} + \sum_{k=3}^{6} t_{j_k 1} + \min_{\substack{j_k \\ j_k \neq 3,4}} \left\{ \sum_{m'=2}^{3} t_{j_k m'} \right\}, $$

$$ {}^{2}B_2^{34} = {}^{2}C_2^{34} + \sum_{k=3}^{6} t_{j_k 2} + \min_{\substack{j_k \\ j_k \neq 3,4}} \left(t_{j_k 3} \right), $$

and

$$ {}^{2}B_3^{34} = {}^{2}C_3^{34} + \sum_{k=3}^{6} t_{j_k 3}, $$

or

$$^{2}B_{1}^{34} = 14 + (9+7+20+10) + \min\,[13+6,\ 7+20,\ 7+2,\ 2+13]$$

$$= 14 + 46 + 9 = 69,$$

$$^{2}B_{2}^{34} = 17 + (13+7+7+2) + \min\,[6,20,2,13]$$

$$= 17 + 29 + 2 = 48,$$

and

$$^{2}B_{3}^{34} = 28 + (6+20+2+13)$$

$$= 69.$$

Thus, the lower bound for this node is

$$^{2}B^{34} = \max\,[69,48,69]$$

$$= 69.$$

Similarly, the lower-bounds for each node at all levels are computed.

3.2 Bounding Procedure LB II

Ignall and Schrage [8] have developed the sophisticated bounding procedure LB II for 3-machine flow shop scheduling problem. For comparison purposes, this procedure is extended for an arbitrary number of machines, M. It differs with LB I only in the computation of the completion time. This procedure considers the bound on each machine independently, such as in LB I. It forms a bound from the total processing time remaining on that machine, together with the minimum run-out time for that machine.

In mathematical terms, the bound, $^{L}B_{m}^{n}$, on machine m, at level L, for node n, can be stated as follows:

$$^{L}B^{n}_{m} = {}^{L}D^{n}_{m} + \sum_{k=L+1}^{J} t_{j_k m} + \min_{\substack{j_k \\ j_k \varepsilon \bar{n}}} \left(\sum_{m'=m+1}^{M} t_{j_k m'} \right),$$

$$k = 1, 2, \ldots, J,$$

$$m = 1, 2, \ldots, M, \qquad (4)$$

where

$$\min_{\substack{j_k \\ j_k \varepsilon \bar{n}}} \left(\sum_{m'=m+1}^{M} t_{j_k m'} \right) = 0, \qquad m = M,$$

At level L, the earliest starting time of the first unscheduled job, j_{L+1}, on machine m, for each node n, ${}^{L}D^{n}_{m}$, is given by

$${}^{L}D^{n}_{m} = \max_{i} \left({}^{L}C^{n}_{m},\ {}^{L}C^{n}_{m-i} + \min_{j_k \varepsilon \bar{n}} \left(\sum_{m'=m-i}^{m-1} t_{j_k m'} \right) \right),$$

$$k = 1, 2, \ldots, J,$$

$$m = 1, 2, \ldots, M, \qquad (5)$$

$$i = 1, 2, \ldots, m-1$$

where

$${}^{L}C^{n}_{m-i} + \min_{j_k \varepsilon \bar{n}} \left(\sum_{m'=m-i}^{m-1} t_{j_k m'} \right) = 0, \qquad m = 1$$

In words, Equation (4) states that at level L, for node n, the schedule time on machine m is bounded from below by the sum of three terms: the earliest starting time of the first unscheduled job, j_{L+1}, on machine m, plus the total processing time of all unscheduled jobs on the same machine, plus the minimum of the total processing times required to perform the unscheduled jobs on succeeding machines, m+1, m+2, . . . , M, i.e., the minimum run-out time on machine m. As in Equation (1), the last term becomes zero when the bound is computed on machine M. Thus, the lower-bound at level L and for each node n is such that

$$^{L}B^{n} = \max_{m} \left[{}^{L}B^{n}_{m} \right]$$

The first term, ${}^{L}D^{n}_{m}$, is obtained by the recursive Equation (5). This term computes the earliest possible starting time for the first unscheduled job on machine m. This is done by estimating the minimum idle time between the completion of last job, j_L, in the partial sequence of node n, and the start of the first unscheduled job, j_{L+1}.

As in LB I, it is sufficient to show the computation of the lower-bounds for node (3) at level 1 and node (34) at level 2.

At level 1 and for node (3), the completion time for scheduled job on each machine is computed according to Equation (3) such that

$$^{1}C^{3}_{1} = \max \left[{}^{1}C^{3}_{0}, \; {}^{0}C^{0}_{1} \right] + t_{31},$$

$$^{1}C^{3}_{2} = \max \left[{}^{1}C^{3}_{1}, \; {}^{0}C^{0}_{2} \right] + t_{32},$$

and

$$^{1}C^{3}_{3} = \max \left[{}^{1}C^{3}_{2}, \; {}^{0}C^{0}_{3} \right] + t_{33},$$

or

$$ {}^{1}C_{1}^{3} = \max \left\{0,\ \ 0\right\} + 6 = 6, $$

$$ {}^{1}C_{2}^{3} = \max \left\{6,\ \ 0\right\} + 4 = 10, $$

and

$$ {}^{1}C_{3}^{3} = \max \left\{10,\ 0\right\} + 8 = 18\ . $$

The earliest starting time of the first unscheduled job on each machine is obtained by using Equation (5), such that

$$ {}^{1}D_{1}^{3} = \max \left\{{}^{1}C_{1}^{3},\ 0\right\}\ , $$

$$ {}^{1}D_{2}^{3} = \max \left\{{}^{1}C_{2}^{3},\ {}^{1}C_{1}^{3} + \min_{\substack{j_k \\ j_k \neq 3}} \left(t_{j_k 1}\right)\right\}, $$

and

$$ {}^{1}D_{3}^{3} = \max \left\{{}^{1}C_{3}^{3},\ {}^{1}C_{2}^{3} + \min_{\substack{j_k \\ j_k \neq 3}} \left(t_{j_k 2}\right)\ ,\ {}^{1}C_{1}^{3} + \min_{\substack{j_k \\ j_k \neq 3}} \left(\sum_{m'=1}^{2} t_{j_k m'}\right)\right\} $$

or

$$ {}^{1}D_{1}^{3} = \max \left\{\ 6,\ \ 0\right\} = 6, $$

$$ \begin{aligned} {}^{1}D_{2}^{3} &= \max \left\{10,\ \ 6 + \min\,[9,\ 7,\ 8,\ 20,\ 10]\right\} \\ &= \max \left\{10,\ \ 6+7\right\} = 13, \end{aligned} $$

and

$$^1D_3^3 = \max \Big(18,\ 10 + \min\,[13,7,3,7,2]\ ,\ 6 + \min\,[9+13,7+7,8+3,20+7,10+2]\Big)$$

$$= \max\,[18,10+2,6+11] = 18.$$

The bounds for node (3) at level 1 on each machine is given as follows:

$$^1B_1^3 = {}^1D_1^3 + \sum_{k=2}^{6} t_{j_k1} + \min_{\substack{j_k \\ j_k \neq 3}} \left(\sum_{m'=2}^{3} t_{j_km'}\right) ,$$

$$^1B_2^3 = {}^1D_2^3 + \sum_{k=2}^{6} t_{j_k2} + \min_{\substack{j_k \\ j_k \neq 3}} \left(t_{j_k3}\right) ,$$

and

$$^1B_3^3 = {}^1D_3^3 + \sum_{k=2}^{6} t_{j_k3} ,$$

or

$$^1B_1^3 = 6 + (9+7+8+20+10) + \min\,[13+6,7+20,3+10,7+2,2+13]$$

$$= 6 + 54 + 9 \qquad\qquad = 69,$$

$$^1B_2^3 = 13 + (13+7+3+7+2) + \min\,[6,20,10,2,13]$$

$$= 13 + 32 + 2 \qquad\qquad = 47,$$

and

$$^{1}B_3^3 = 18 + (6+20+10+2+13)$$

$$= 18 + 51 \qquad = 69.$$

Thus, the lower-bound at level 1 for node (3) is

$$^{1}B^3 = \max\left[{}^{1}B_1^3,\ {}^{1}B_2^3,\ {}^{1}B_3^3\right]$$

$$= \max\left[69,\ 47,\ 69\right]$$

$$= 69.$$

Following the same steps, the lower-bounds for nodes (1), (2), (4), (5) and (6) is found to be 81, 73, 70, 87 and 71, respectively.

At level 2 and for node (34), the completion time on each machines are computed such that

$$^{2}C_1^{34} = \max\left[{}^{2}C_0^{34},\ {}^{1}C_1^3\right] + t_{41},$$

$$^{2}C_2^{34} = \max\left[{}^{2}C_1^{34},\ {}^{1}C_2^3\right] + t_{42},$$

and

$$^{2}C_3^{34} = \max\left[{}^{2}C_2^{34},\ {}^{1}C_3^3\right] + t_{43},$$

or

$$^{2}C_1^{34} = \max\left[0,\ 6\right] + 8 = 14,$$

$$^{2}C_2^{34} = \max\left[14,\ 10\right] + 3 = 17,$$

and

$${}^{2}C_{3}^{34} = \max \left(17, \quad 18 \right) + 10 = 28.$$

The earliest starting time of the first unscheduled job on each machine is given such that

$${}^{2}D_{1}^{34} = \max \left({}^{2}C_{1}^{34}, \quad 0 \right) ,$$

$${}^{2}D_{2}^{34} = \max \left({}^{2}C_{2}^{34}, \ {}^{2}C_{1}^{34} + \min_{\substack{j_k \\ j_k \neq 3,4}} \left(t_{j_k 1} \right) \right) ,$$

and

$${}^{2}D_{3}^{34} = \max \left({}^{2}C_{3}^{34}, \ {}^{2}C_{2}^{34} + \min_{\substack{j_k \\ j_k \neq 3,4}} \left(t_{j_k 2} \right) , \ {}^{2}C_{1}^{34} + \min_{\substack{j_k \\ j_k \neq 3,4}} \left(\sum_{m'=1}^{2} t_{j_k m'} \right) \right) ,$$

or

$${}^{2}D_{1}^{34} = \max \left(14, \quad 0 \right)$$

$$= 14,$$

$${}^{2}D_{2}^{34} = \max \left(17, \ 14 + \min \ [9,7,20,10] \right)$$

$$= \max \left(17, \ 21 \right)$$

$$= 21,$$

and

$${}^{2}D_{3}^{34} = \max\left(28,\ 17 + \min\,[13,7,7,2],\ 14 + \min\,[9+13,7+7,20+7,10+2]\right)$$

$$= \max\,[28,19,26]$$

$$= 28.$$

At level 2 and for node (34) the bounds on all machines are given such that

$${}^{2}B_{1}^{34} = {}^{2}D_{1}^{34} + \sum_{k=3}^{6} t_{j_k 1} + \min_{\substack{j_k \\ j_k \neq 3,4}} \left(\sum_{m'=2}^{3} t_{j_k m'}\right),$$

$${}^{2}B_{2}^{34} = {}^{2}D_{2}^{34} + \sum_{k=3}^{6} t_{j_k 2} + \min_{\substack{j_k \\ j_k \neq 3,4}} \left(t_{j_k 3}\right),$$

and

$${}^{2}B_{3}^{34} = {}^{2}D_{3}^{34} + \sum_{k=3}^{6} t_{j_k 3},$$

or

$${}^{2}B_{1}^{34} = 14 + (9+7+20+10) + \min\,[13+6,7+20,7+2,2+13]$$

$$= 14 + 46 + 9$$

$$= 69,$$

$${}^{2}B_{2}^{34} = 21 + (13+7+7+2) + \min\,[6,20,2,13]$$

$$= 21 + 29 + 2$$

$$= 52,$$

and

$$^{2}B_{3}^{34} = 28 + (6+20+2+13)$$

$$= 69.$$

Therefore, the lower-bound at level 2 and for the node (34) is

$$^{2}B^{34} = \max\ [69,52,69]$$

$$= 69\ .$$

Similarly, the remaining lower-bounds are computed for all nodes at each level.

3.3 Bounding Procedure LB III

This bounding procedure has been introduced by McMahon and Burton [13], which in some circumstances gives a larger value than that obtainable by other lower-bounds. The bounding procedure, LB III, has been referred to as job-based bound, since it expresses the fact that the schedule time may be determined by the total processing time for a job, rather than by the total processing time on one machine.

The bound on the schedule time, $^{L}B_{m}^{n}$, is expressed such that

$$^{L}B_{m}^{n} = {}^{L}C_{m}^{n} + \max_{j_k \varepsilon \bar{n}} \left(\sum_{m'=m}^{M} t_{j_k m'} + \sum_{\substack{x=L+1 \\ j_x \neq j_k}}^{J} \min \left(t_{j_x m},\ t_{j_x M} \right) \right),$$

$$k = L+1,\ L+2,\ .\ .\ .\ ,\ J$$

$$m = 1,\ 2,\ .\ .\ .\ .\ ,\ M \qquad (6)$$

$$L = 1,\ 2,\ .\ .\ .\ ,\ J$$

where

$$\max_{j_k \varepsilon \bar{n}} \left(\sum_{m'=m}^{M} t_{j_k m'} + \sum_{\substack{x=L+1 \\ j_x \neq j_k}}^{J} \min \left(t_{j_x m}, t_{j_x M} \right) \right) = \sum_{k=L+1}^{J} t_{j_k M}, \quad m = M.$$

In words, Equation (6) states that at level L, the schedule time of each node n, on machine m, is bounded from below by the sum of the following two terms. The first term represents the completion time of all jobs included in the partial sequence of node n, at level L, on machine m. This is also an estimate of the earliest possible starting time for any unscheduled job on machine m, assuming no conflict for this job. The second term consists of two parts: (1) the total processing time on one of the unscheduled jobs, $j_k \varepsilon \bar{n}$, on machines m, m+1, . . . , M, assuming no conflict or overlapping for this job on any of these machines; (2) the sum of minimum processing time either on machine m or on machine M, for all remaining unscheduled jobs because each of these jobs must either precede job j_k on machine m or follow it on machine M.

Similar to the previous bounding procedures, the computation of LB III is shown for only node (3) at level 1 and node (34) at level 2.

At level 1 and for node (3), the completion time for scheduled jobs on each machine is computed such that

$${}^{1}C_{1}^{3} = \max \left({}^{1}C_{0}^{3}, {}^{0}C_{1}^{0} \right) + t_{31},$$

$${}^{1}C_{2}^{3} = \max \left({}^{1}C_{1}^{3}, {}^{0}C_{2}^{0} \right) + t_{32},$$

and

$$^{1}C_3^3 = \max\left[{}^{1}C_2^3,\ {}^{0}C_3^0\right] + t_{33},$$

or

$$^{1}C_1^3 = \max\left[\ 0,\ \ 0\ \right] + 6 = 6,$$

$$^{1}C_2^3 = \max\left[\ 6,\ \ 0\ \right] + 4 = 10,$$

and

$$^{1}C_3^3 = \max\left[\ 10,\ \ 0\ \right] + 8 = 18.$$

Bounds on schedule time, at level 1 and for node (3) on each machine is given by

$$^{1}B_1^3 = {}^{1}C_1^3 + \max_{\substack{j_k \\ j_k \neq 3}} \left[\sum_{m'=1}^{3} t_{j_k m'} + \sum_{\substack{x=2 \\ j_x \neq j_k}}^{6} \min\left(t_{j_x 1},\ t_{j_x 3}\right)\right],$$

$$^{1}B_2^3 = {}^{1}C_2^3 + \max_{\substack{j_k \\ j_k \neq 3}} \left[\sum_{m'=2}^{3} t_{j_k m'} + \sum_{\substack{x=2 \\ j_x \neq j_k}}^{6} \min\left(t_{j_x 2},\ t_{j_x 3}\right)\right],$$

and

$$^{1}B_3^3 = {}^{1}C_3^3 + \sum_{k=2}^{6} t_{j_k 3},$$

or

$$^{1}B_1^3 = 6 + \max\ [(9+13+6) + (7+8+2+10),\ (7+7+20)$$

$$+ (6+8+2+10),\ (8+3+10) + (6+7+2+10),\ (20+7+2)$$

$$+ (6+7+8+10),\ (10+2+13) + (6+7+8+2)]$$

$$= 6 + \max\,[55, 60, 46, 60, 48]$$

$$= 66,$$

$${}^{1}B_{2}^{3} = 10 + \max\,[19+14, 27+13, 13+17, 9+18, 15+18]$$

$$= 10 + \max\,[33, 40, 30, 27, 33]$$

$$= 50,$$

and

$${}^{1}B_{3}^{3} = 18 + (6 + 20 + 10 + 2 + 13)$$

$$= 18 + 51$$

$$= 69\ .$$

The lower-bound for this node is given by

$${}^{1}B^{3} = \max\,[66, 50, 69]$$

$$= 69.$$

At level 2 and for node (34), the completion times for scheduled jobs on each machine is computed such that

$${}^{2}C_{1}^{34} = \max\left({}^{2}C_{0}^{34},\ {}^{1}C_{1}^{3}\right) + t_{41},$$

$${}^{2}C_{2}^{34} = \max\left({}^{2}C_{1}^{34},\ {}^{1}C_{2}^{3}\right) + t_{42},$$

and

$$^{2}C_{3}^{34} = \max\left[{}^{2}C_{2}^{34},\ {}^{1}C_{3}^{3}\right] + t_{43},$$

or

$$^{2}C_{1}^{34} = \max\left[\ 0,\quad 6\right] + 8$$

$$= 14,$$

$$^{2}C_{2}^{34} = \max\left[\ 14,\quad 10\right] + 3$$

$$= 17,$$

and

$$^{2}C_{3}^{34} = \max\left[\ 17,\quad 18\right] + 10$$

$$= 28\ .$$

The bounds for this node on each machine is given by

$$^{2}B_{1}^{34} = {}^{2}C_{1}^{34} + \max_{\substack{j_k \\ j_k \neq 3,4}} \left[\sum_{m'=1}^{3} t_{j_k m'} + \sum_{\substack{x=3 \\ j_x \neq j_k}}^{6} \min\left(t_{j_x 1},\ t_{j_x 3}\right)\right],$$

$$^{2}B_{2}^{34} = {}^{2}C_{2}^{34} + \max_{\substack{j_k \\ j_k \neq 3,4}} \left[\sum_{m'=2}^{3} t_{j_k m'} + \sum_{\substack{x=3 \\ j_x \neq j_k}}^{6} \min\left(t_{j_x 2},\ t_{j_x 3}\right)\right],$$

and

$$^{2}B_{3}^{34} = {}^{2}C_{3}^{34} + \sum_{k=3}^{6} t_{j_k 3}\ ,$$

or

$$
\begin{aligned}
{}^{2}B_1^{34} &= 14 + \max\,[(9+13+6) + (7+2+10),\ (7+7+20) + (6+2+10),\ (20+7+2) \\
&\qquad + (6+7+10),\ (10+2+13) + (6+7+2)] \\
&= 14 + \max\,[47,\ 52,\ 52,\ 40] \\
&= 14 + 52 \\
&= 66,
\end{aligned}
$$

$$
\begin{aligned}
{}^{2}B_2^{34} &= 17 + \max\,[19+11,\ 27+10,\ 9+15,\ 15+15] \\
&= 17 + \max\,[30,\ 37,\ 24,\ 30] \\
&= 54,
\end{aligned}
$$

and

$$
\begin{aligned}
{}^{2}B_3^{34} &= 28 + (6 + 20 + 2 + 13) \\
&= 28 + 41 \\
&= 69\ .
\end{aligned}
$$

Thus the lower-bound for node (34) at level 2 is

$$
\begin{aligned}
{}^{2}B^{34} &= \max\,[66,\ 54,\ 69] \\
&= 69\ .
\end{aligned}
$$

Similarly, the other lower-bounds for all nodes at each level are computed.

3.4 Bounding Procedure LB IV

This procedure has been developed by McMahon and Burton [13] with the idea of selecting the more powerful bound from the machine-based bound and the job-based bound. The bounding procedure, LB IV, has been referred to as a composite lower-bound, which is computed such that

$$\text{LB IV} = \max\,[\text{LB I}, \text{LB III}] \;.$$

To obtain composite lower-bound for any node n, the lower-bounds by bounding procedures LB I and LB III are first computed for that node. The maximum of these two lower-bounds is the composite lower-bound for that particular node.

3.5 Bounding Procedure LB V

This procedure has been developed by Nabeshima [15], referred to as revised lower-bound. This may be considered as a machine-based bound. Nabeshima has claimed that the bounding procedure, LB V, is superior over the other bounding procedures because it reduces the number of explored nodes in his sample problem.

This bounding procedure utilizes Johnson's criterion [9] of finding the optimal sequences on 2 machines, for the purpose of estimating powerful lower-bound. It has already been pointed out that LB I does not account for any idle time for the unscheduled jobs while computing total processing time for these jobs on machine m. However, LB V takes into consideration the estimation of this idle time by applying Johnson's criterion for the two-machine case as explained below.

The estimation of the idle time for the unscheduled jobs requires that they should have a specific sequence determined by any reasonable rule. LB V considers the sequence of the unscheduled jobs on consecutive two machines m and m+1, where $m = 1, 2, \ldots, M-1$; by using criterion given by Johnson for independent two machines m, m+1. For any two jobs

j_r, j_s in $\bar{n}$, if

$$\min \left\{t_{j_r m},\ t_{j_s m+1}\right\} \leq \min \left\{t_{j_r m+1},\ t_{j_s m}\right\} \tag{7}$$

holds, then job j_r must precede job j_s in order to minimize the schedule time of the sequence of J-L jobs in $\bar{n}$ on machines m and m+1. The sequence constructed according to Johnson's criterion is referred to as preliminary sequence. Thus for M-machine problem, there will be M-1 preliminary partial sequences, $\bar{n}_{m,m+1}$ or $\{(j_{L+1}\ j_{L+2} \ldots j_J)_{m,m+1}\}$, m = 1, 2, . . . , M-1, of unscheduled jobs in $\bar{n}$. Hence, the sequence $\{j_1\ j_2 \ldots j_L\ (j_{L+1}\ j_{L+2} \ldots j_J)_{m,m+1}\}$, must be processed on machines m, m+1 in this order. Due to the nature of this sequence, it is referred to as dynamic sequence. The first L jobs in this dynamic sequence do not interchange their sequence-positions on any of the M machines, but the remaining J-L jobs may interchange their sequence-positions on some or all of the M machines. The bounding procedure, LB V is stated below.

At level L, for node n, on machine m, the bound ${}^{L}B^{n}_{m}$ is given by

$${}^{L}B^{n}_{m} = {}^{L}C^{n}_{m} + {}^{L}F^{\bar{n}_{m-1,m}}_{m} + \min_{j_k \varepsilon \bar{n}} \left[\sum_{m'=m+1}^{M} t_{j_k m'}\right],$$

$$m = 2, 3, \ldots, M-1 \tag{8}$$

and

$${}^{L}B^{n}_{M} = {}^{L}C^{n}_{M} + {}^{L}F^{\bar{n}_{M-1,M}}_{M}$$

where ${}^{L}F^{\bar{n}_{m-1,m}}_{m}$ is the total processing time of jobs in the preliminary partial sequence, $\{(j_{L+1}\ j_{L+2} \ldots j_J)_{m-1,m}\}$, on machine m after the completion time, ${}^{L}C^{n}_{m}$.

Equation (8) may further be simplified by combining the first two terms in the right hand side such that

$$ {}^{L}B^{n}_{m} = {}^{L}C^{n_{m-1,m}}_{m} + \min_{j_k \varepsilon \bar{n}} \left(\sum_{m'=m+1}^{M} t_{j_k m'} \right) $$

$$ m = 2, 3, \ldots, M-1, \tag{9} $$

and

$$ {}^{L}B^{n}_{M} = {}^{L}C^{n_{M-1,M}}_{M} $$

where $n_{m-1,m}$ is the dynamic sequence of J jobs on machines m-1 and m, $\{j_1\ j_2 \ldots j_L\ (j_{L+1}\ j_{L+2} \ldots j_J)_{m-1,m}\}$.

The completion time for the scheduled jobs, ${}^{L}C^{n}_{m}$, is computed such that

$$ {}^{L}C^{n}_{m} = \max \left({}^{L}C^{j_1 j_2 \ldots j_L}_{m-1} , {}^{L-1}C^{j_1 j_2 \ldots j_{L-1}}_{m} \right) + t_{j_L m} , $$

$$ m = 1, 2, \ldots, M, \tag{10} $$

where

$$ {}^{L}C^{j_1 j_2 \ldots j_L}_{m-1} = 0, \qquad m = 1 $$

and

$$ {}^{L-1}C^{j_1 j_2 \ldots j_{L-1}}_{m}, \qquad L = 1 $$

The completion times for the unscheduled jobs which are arranged according to preliminary sequence, are computed by the following recurrent relation:

$$ {}^{L}C_{m}^{n}(j_{L+1}, j_{L+2}, \ldots, j_{L+h}) = \max\left[{}^{L}C_{m}^{n}(j_{L+1} \ldots j_{L+h-1}),\ {}^{L}C_{m-1}^{n} + \sum_{\ell=1}^{h} t_{j_{L+\ell}m-1}\right] + t_{j_{L+h}m}, $$

$$ h = 1, 2, \ldots, J-L, $$

$$ m = 2, 3, \ldots, M, \qquad (11) $$

where

$$ {}^{L}C_{m}^{n}(j_{L+1} \ldots j_{L+h-1}) = {}^{L}C_{m}^{n}, \qquad h = 1 $$

The lower-bound, ${}^{L}B^{n}$, at level L, for node n is given by

$$ {}^{L}B^{n} = \max_{m}\left[{}^{L}B_{m}^{n}\right] . $$

In words, Equation (9) is composed of two terms: The first term is the measure of the completion time of job j_J in the last sequence-position of the dynamic sequence, $\{j_1\ j_2 \ldots j_L\ (j_{L+1}\ j_{L+2} \ldots j_J)_{m-1,m}\}$, on machine m. This is the time elapsed between the start of job j_1 on machine 1 and completion of job j_J on machine m, where j_1 and j_J belong to the same dynamic sequence. In addition to the sum of processing times of all jobs on machine m, this term also includes the exact idle time for jobs in the partial sequence of node n, and an estimate of idle time for unscheduled jobs, $\bar{n}$. This term is computed by recursive Equations (10) and (11) which are similar to the Equation (3) of LB I.

The second term in Equation (9) computes the minimum run-out time for machine m. Run-out time is the duration of time between the completion of last job, j_J, on machine m and the completion of the same job on the last machine M. This term does not include idle time on any of the succeeding machines, m+1, m+2, . . . , M. It is obvious from definition that the run-out time on machine M is zero.

Theoretically, it may be stated that this lower-bound is more powerful in reducing the total number of nodes explored. This will always estimate bound which is equal to or greater than that computed by LB I, since

$$^{L\,\bar{n}}F_m^{m-1,m} \geq \sum_{k=L+1}^{J} t_{j_k m}$$

Again, as in the previous bounding procedures, the computation of the lower-bounds for node (3) at level 1 and node (34) at level 2 will be shown below.

As required in the bounding procedure LB V, the preliminary sequence on machines 1 and 2 is {2 1 5 3 4 6} , and on machines 2 and 3 is {6 4 3 2 1 5} . For more detail refer to Appendix A in which Johnson's algorithm is applied.

At level 1 and for node (3), the completion time for scheduled job is computed according to Equation (10) such that

$$^{1}C_1^3 = \max\left({}^{1}C_0^3,\ {}^{0}C_1^0\right) + t_{31},$$

and

$${}^{1}C_{2}^{3} = \max\left({}^{1}C_{1}^{3},\ {}^{0}C_{2}^{0}\right) + t_{32},$$

or

$${}^{1}C_{1}^{3} = \max\left(0,\quad 0\right) + 6$$

$$= 6,$$

and

$${}^{1}C_{2}^{3} = \max\left(6,\quad 0\right) + 4$$

$$= 10.$$

The completion time for the last job in the dynamic sequence, $\{3(2\ 1\ 5\ 4\ 6)_{1,2}\}$, is obtained such that

$${}^{1}C_{2}^{3(2)} = \max\left({}^{1}C_{2}^{3},\ {}^{1}C_{1}^{3} + t_{j_2 1}\right) + t_{j_2 2}\ ,$$

$${}^{1}C_{2}^{3(21)} = \max\left({}^{1}C_{2}^{3(2)},\ {}^{1}C_{1}^{3} + \sum_{\ell=1}^{2} t_{j_{1+\ell}1}\right) + t_{j_3 2}\ ,$$

$${}^{1}C_{2}^{3(215)} = \max\left({}^{1}C_{2}^{3(21)},\ {}^{1}C_{1}^{3} + \sum_{\ell=1}^{3} t_{j_{1+\ell}1}\right) + t_{j_4 2}\ ,$$

$${}^{1}C_{2}^{3(2154)} = \max\left({}^{1}C_{2}^{3(215)},\ {}^{1}C_{1}^{3} + \sum_{\ell=1}^{4} t_{j_{1+\ell}1}\right) + t_{j_5 2}\ ,$$

and

$${}^{1}C_{2}^{3(21546)} = \max\left\{{}^{1}C_{2}^{3(2154)},\ {}^{1}C_{1}^{3} + \sum_{\ell=1}^{5} t_{j_{1+\ell}1}\right\} + t_{j_6 2}\ ,$$

or

$${}^{1}C_{2}^{3(2)} = \max\left\{10,\ 6+7\right\} + 7 = 20\ ,$$

$${}^{1}C_{2}^{3(21)} = \max\left\{20,\ 6+7+9\right\} + 13 = 35\ ,$$

$${}^{1}C_{2}^{3(215)} = \max\left\{35,\ 6+7+9+20\right\} + 7 = 49,$$

$${}^{1}C_{2}^{3(2154)} = \max\left\{49,\ 6+7+9+20+8\right\} + 3 = 53\ ,$$

and

$${}^{1}C_{2}^{3(21546)} = \max\left\{53,\ 6+7+9+20+8+10\right\} + 2 = 62\ .$$

Similarly, the completion time for the job in the last sequence-position of the dynamic sequence, $\{3(6\ 4\ 2\ 1\ 5)_{2,3}\}$, on machine 3 is computed.

Firstly, the time within which all jobs included in the partial sequence of node n are completed on machine 3, is computed by using Equation (10) such that

$${}^{1}C_{3}^{3} = \max\left\{{}^{1}C_{2}^{3},\ {}^{0}C_{3}^{0}\right\} + t_{33}\ ,$$

$$= \max\left\{\ 10,\ \ 0\ \right\} + 8 = 18\ .$$

Using Equation (11),

$$ {}^1C_3^{3(6)} = \max\left\{{}^1C_3^3,\ {}^1C_2^3 + t_{j_2 2}\right\} + t_{j_2 3} , $$

$$ {}^1C_3^{3(64)} = \max\left\{{}^1C_3^{3(6)},\ {}^1C_2^3 + \sum_{\ell=1}^{2} t_{j_{1+\ell} 2}\right\} + t_{j_3 3} , $$

$$ {}^1C_3^{3(642)} = \max\left\{{}^1C_3^{3(64)},\ {}^1C_2^3 + \sum_{\ell=1}^{3} t_{j_{1+\ell} 2}\right\} + t_{j_4 3} , $$

$$ {}^1C_3^{3(6421)} = \max\left\{{}^1C_3^{3(642)},\ {}^1C_2^3 + \sum_{\ell=1}^{4} t_{j_{1+\ell} 2}\right\} + t_{j_5 3} , $$

and

$$ {}^1C_3^{3(64215)} = \max\left\{{}^1C_3^{3(6421)},\ {}^1C_2^3 + \sum_{\ell=1}^{5} t_{j_{1+\ell} 2}\right\} + t_{j_6 3} , $$

or

$$ {}^1C_3^{3(6)} = \max\left\{18,\ 10+2\right\} + 13 = 31 , $$

$$ {}^1C_3^{3(64)} = \max\left\{31,\ 10+2+3\right\} + 10 = 41 , $$

$$ {}^1C_3^{3(642)} = \max\left\{41,\ 10+2+3+7\right\} + 20 = 61 , $$

$$ {}^1C_3^{3(6421)} = \max\left\{61,\ 10+2+3+7+13\right\} + 6 = 67 , $$

and

$$ {}^{1}C_3^{3(64215)} = \max\left(67,\ 10+2+3+7+13+7\right) + 2 = 69 \ . $$

The bounds for node (3) at level 1 are computed on machines 2 and 3 such that

$$ {}^{1}B_2^{3} = {}^{1}C_2^{3(21546)} + \min_{\substack{j_k \\ j_k \neq 3}} \left(t_{j_k 3}\right) , $$

and

$$ {}^{1}B_3^{3} = {}^{1}C_3^{3(64215)} , $$

or

$$ {}^{1}B_2^{3} = 62 + \min[6,\ 20,\ 10,\ 2,\ 13] $$

$$ = 64 \ , $$

and

$$ {}^{1}B_3^{3} = 69 \ . $$

The lower-bound for this node at level 1 is then given by

$$ {}^{1}B^{3} = \max\ [64,\ 69] $$

$$ = 69 \ . $$

Similarly, the lower-bounds for nodes (1), (2), (4), (5) and (6) at level 1 are found to be 81, 73, 70, 86, and 71, respectively.

At level 2, for node (34) the completion time for all scheduled jobs is computed on machines 1, 2 and 3 by Equation (10) such that

$$ {}^{2}C_1^{34} = \max\left({}^{2}C_0^{34},\ {}^{1}C_1^{3}\right) + t_{41} \ , $$

$$ {}^{2}C_{2}^{34} = \max\left[{}^{2}C_{1}^{34},\ {}^{1}C_{2}^{3}\right] + t_{42}\ , $$

and

$$ {}^{2}C_{3}^{34} = \max\left[{}^{2}C_{2}^{34},\ {}^{1}C_{3}^{3}\right] + t_{43}\ , $$

or

$$ {}^{2}C_{1}^{34} = \max\left[\ 0,\ \ 6\ \right] + 8 = 14, $$

$$ {}^{2}C_{2}^{34} = \max\left[14,\ \ 10\right] + 3 = 17, $$

and

$$ {}^{2}C_{3}^{34} = \max\left[17,\ \ 18\right] + 10 = 28. $$

The completion time for the job in the last sequence-position of the dynamic sequence, $\{3\ 4\ (2\ 1\ 5\ 6)_{1,2}\}$, on machine 2 is computed, such that

$$ {}^{2}C_{2}^{34(2)} = \max\left[{}^{2}C_{2}^{34},\ {}^{2}C_{1}^{34} + t_{j_3 1}\right] + t_{j_3 2}\ , $$

$$ {}^{2}C_{2}^{34(21)} = \max\left[{}^{2}C_{2}^{34(2)},\ {}^{2}C_{1}^{34} + \sum_{\ell=1}^{2} t_{j_{2+\ell} 1}\right] + t_{j_4 2}\ , $$

$$ {}^{2}C_{2}^{34(215)} = \max\left[{}^{2}C_{2}^{34(21)},\ {}^{2}C_{1}^{34} + \sum_{\ell=1}^{3} t_{j_{2+\ell} 1}\right] + t_{j_5 2}\ , $$

and

$$ {}^{2}C_{2}^{34(2156)} = \max\left[{}^{2}C_{2}^{34(215)},\ {}^{2}C_{1}^{34} + \sum_{\ell=1}^{4} t_{j_{2+\ell} 1}\right] + t_{j_6 2}\ , $$

or

$$^{2}C_{2}^{34(2)} = \max\left[17,\ 14+7\right] + 7 = 28\ ,$$

$$^{2}C_{2}^{34(21)} = \max\left[28,\ 14+7+9\right] + 13 = 43\ ,$$

$$^{2}C_{2}^{34(215)} = \max\left[43,\ 14+7+9+20\right] + 7 = 57,$$

and

$$^{2}C_{2}^{34(2156)} = \max\left[57,\ 14+7+9+20+10\right] + 2 = 62\ .$$

Similarly, the completion time for the job in the last sequence-position of the dynamic sequence, $\{3\ 4\ (6\ 2\ 1\ 5)_{2,3}\}$, on machine 3 is computed such that

$$^{2}C_{3}^{34(6)} = \max\left(^{2}C_{3}^{34},\ ^{2}C_{2}^{34} + t_{j_3 2}'\right) + t_{j_3 3}\ ,$$

$$^{2}C_{3}^{34(62)} = \max\left(^{2}C_{3}^{34(6)},\ ^{2}C_{2}^{34} + \sum_{\ell=1}^{2} t_{j_{2+\ell}2}\right) + t_{j_4 3}\ ,$$

$$^{2}C_{3}^{34(621)} = \max\left(^{2}C_{3}^{34(62)},\ ^{2}C_{2}^{34} + \sum_{\ell=1}^{3} t_{j_{2+\ell}2}\right) + t_{j_5 3}\ ,$$

and

$$^{2}C_{3}^{34(6215)} = \max\left(^{2}C_{3}^{34(621)},\ ^{2}C_{2}^{34} + \sum_{\ell=1}^{4} t_{j_{2+\ell}2}\right) + t_{j_6 3}\ ,$$

or

$${}^{2}C_{3}^{34(6)} = \max\left[28, \ 17+2\right] + 13 = 41,$$

$${}^{2}C_{3}^{34(62)} = \max\left[41, \ 17+2+7\right] + 20 = 61,$$

$${}^{2}C_{3}^{34(621)} = \max\left[61, \ 17+2+7+13\right] + 6 = 67,$$

and

$${}^{2}C_{3}^{34(6215)} = \max\left[67, \ 17+2+7+13+7\right] + 2 = 69.$$

The bounds for node (34) at level 2 are computed on machines 2 and 3 such that

$${}^{2}B_{2}^{34} = {}^{2}C_{2}^{34(2156)} + \min_{\substack{j_k \\ j_k \neq 3,4}} \left(t_{j_k 3}\right) ,$$

and

$${}^{2}B_{3}^{34} = {}^{2}C_{3}^{34(6215)},$$

or

$${}^{2}B_{2}^{34} = 62 + \min\left[6, \ 20, \ 2, \ 13\right]$$

$$= 64,$$

and

$${}^{2}B_{3}^{34} = 69$$

The lower-bound for this node is given by

$${}^{2}B^{34} = \max\left[64, \ 69\right]$$

$$= 69.$$

Similarly, the lower-bounds for each node in the scheduling tree are computed.

3.6 Sample Problem

As mentioned earlier, a sample problem is solved to illustrate the various bounding procedures. In order to be fair in the illustration of the bounding procedures, this problem is selected in which the solution is obtained by exploring the minimum number of nodes, using the various bounding procedures. However, the quality of the lower-bounds will be investigated by several computational experiments reported in Chapter IV. The sample problem is solved by the branch-and-bound algorithm stated in Section 2.3.

The sample problem consists of sequencing six jobs on three machines such that the schedule time is minimized. The machine ordering and processing time matrices are given below:

$$M = \begin{bmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \\ 41 & 42 & 43 \\ 51 & 52 & 53 \\ 61 & 62 & 63 \end{bmatrix} ; \qquad T = \begin{bmatrix} 9 & 13 & 6 \\ 7 & 7 & 20 \\ 6 & 4 & 8 \\ 8 & 3 & 10 \\ 20 & 7 & 2 \\ 10 & 2 & 13 \end{bmatrix}$$

Following the branch-and-bound algorithm, level 1 is initialized by generating the nodes (1), (2), (3), (4), (5), and (6). The lower-bounds obtained by the various bounding procedures are tabulated for all nodes at level 1 as follows:

Level	Node	LB I	LB II	LB III	LB IV	LB V
1	(1)	81	81	81	81	81
	(2)	73	73	73	73	73
	(3)	69*	69*	69*	69*	69*
	(4)	70	70	70	70	70
	(5)	86	87	86	86	86
	(6)	71	71	71	71	71

Note that * indicates the minimum lower-bound in each bounding procedures. Node (3) has the least lower-bound. Therefore, this node is branched to form the nodes (31), (32), (34), (35), and (36) at level 2.

At level 2, the computed lower-bounds for each of the above nodes are summarized as follows:

Level	Node	LB I	LB II	LB III	LB IV	LB V
2	(31)	79	79	79	79	79
	(32)	71	71	71	71	71
	(34)	69*	69*	69*	69*	69*
	(35)	84	86	84	84	84
	(36)	69*	69*	69*	69*	69*

Upon examining the above lower-bounds, the tie exists between the nodes (34) and (36) which have the least lower-bounds, 69. The tie is resolved in favor of node (34).

At level 3, node (34) is branched to create the nodes (341), (342), (345), and (346). The lower-bounds are computed and tabulated below.

Level	Node	LB I	LB II	LB III	LB IV	LB V
3	(341)	77	77	77	77	77
	(342)	69*	69*	69*	69*	69*
	(345)	82	85	84	84	82
	(346)	69*	69*	69*	69*	69*

Again, the minimum lower-bound is 69 and a tie exists between those nodes (342) and (346). The tie is resolved in favor of node (342).

At level 4, node (342) is branched to generate the nodes (3421), (3425), and (3426). The lower-bound for each of these newly created nodes is computed and summarized below.

Level	Node	LB I	LB II	LB III	LB IV	LB V
4	(3421)	69*	69*	69*	69*	69*
	(3425)	75	75	79	79	71
	(3426)	69*	69*	69*	69*	69*

The nodes (3421) and (3426) have the least lower-bound. The tie is broken and the node (3421) is selected for branching.

At level 5, the newly created nodes are (34215), and (34216); and the lower-bounds are tabulated below.

Level	Node	LB I	LB II	LB III	LB IV	LB V
5	(34215)	75	75	75	75	75
	(34216)	69*	69*	69*	69*	69*

At this level, the node having the least lower-bound is selected. By adding the only remaining job to the partial sequence of that node, a complete sequence is obtained. The minimum lower-bound at level 5 is 69 for node (34216), and the unscheduled job for this node is job 5. Thus the feasible sequence {3 4 2 1 6 5} is a solution with schedule time T or 69.

Now, the back-tracking process is carried over all preceding levels and since there is no node having lower-bound less than 69, the sequence {3 4 2 1 6 5} is an optimal solution with minimum schedule time of 69. It should be noted that the solution is obtained after exploring 20 nodes.

CHAPTER IV

COMPUTATIONAL EXPERIMENTS

To establish a fair comparison among the various bounding procedures discussed in Chapter III, a series of computational experiments were conducted on IBM 360/50 computer. The computational algorithm presented in Section 2.3 was programmed in FORTRAN IV language. The flow chart and the computer program will appear in Appendix B. This chapter is devoted to the computational experiments and their results.

4.1 Experiments I - X

The experiments performed in this research consist of 395 problems which were selected with six to twelve jobs and three to five machines. The entries of the processing time matrices were generated at random from a uniform distribution between one and 30, inclusive. The number of problems in these experiments varied from 10 to 50. These computational experiments were designed in order to investigate the effects of changes in both the number of jobs and the number of machines. A summary of these experiments is tabulated below.

Experiment No.	Problem Size (JxM)	Number of Problems	Experiment No.	Problem Size (JxM)	Number of Problems
I	6x3	50	VII	8x3	35
II	6x4	50	VIII	8x4	25
III	6x5	50	IX	8x5	25
IV	7x3	50	X	12x3	10
V	7x4	50			
VI	7x5	50			

4.2 Experimental Results

This section is devoted to the results of the various computational experiments. These results will help explore empirically the performance of the various bounding procedures. The number of nodes and computation time of all experiments are summarized in Tables 4.1 - 4.5 and discussed below. Various statistics such as the minimum, maximum, mean and standard deviation are reported for both the number of nodes explored and the computation time spent to obtain the solution.

The results of Experiments X which consists of 10 problems solved by Ashour [2] are summarized in Table 4.6. The purpose of this experiment was to note the feasibility and efficiency of various bounding procedures for larger problems. The solutions obtained by LB I, LB II, LB III, LB IV and LB V were for 9, 9, 6, 10 and 5 problems, respectively. On comparing this result from that given in [2], it is obvious that branch-and-bound technique worked more efficiently than the various other techniques discussed therein.

As mentioned earlier, the branch-and-bound algorithm requires the exploration of at least $J + (J-1) + . . . + 2$ nodes; however in many cases more nodes may be explored; their number can be regarded as a measure of computational effort required in arriving at the solution for a specific bounding procedure. It is of interest to point out that the size of the scheduling tree, i.e., the number of all the possible nodes for exploration is $J + J(J-1) + . . . + J!$ which increases with the increase of J. For problems of various sizes, the minimum and maximum number of nodes which might be explored are shown below.

	Number of Jobs			
	6	7	8	12
Minimum number of explored nodes	20	27	35	77
Maximum number of explored nodes*	1,236	8,659	69,280	3,297,901,344

*That is the size of the scheduling tree.

In analyzing the results obtained by various bounding procedures, the effects of the changes in the number of jobs and machines are as follows:

The number of nodes explored to reach an optimal solution increases rapidly as the number of jobs increases. It was also noticed that the rate of change of increase in the number of nodes, increases rapidly with the increase in the number of jobs. For example in Table 4.1, Experiments I, IV and VII having problems of sizes (6x3), (7x3), and (8x3), the number of nodes explored increases by 127.28 and 235.45 nodes as the number of jobs increases from 6 to 7 and 7 to 8, respectively. It should be noted that the increase of similar nature is also reported for other bounding procedures in Tables 4.2 - 4.5. As an example, for LB III the number of explored nodes increases by factors of about 4.43 and 5.48 when the problem sizes change from (6x3) to (7x3) and from (7x3) to (8x3), respectively. However, these factors vary greatly with both the bounding procedure and the problem size. The steep increase in the rate of change of the explored nodes with increase in the number of jobs, is observed for each bounding procedure in all the experiments. Table 4.7 shows the various factors α for all bounding procedures obtained from different problem sizes. The reason for the increase in the number of nodes explored with the increase of the number of jobs is that the addition of one job to the problem increases the level by one in the scheduling tree. Since nodes are formed by various permutations of these jobs, the number of nodes formed should increase rapidly with the increase in the number of jobs. Thus by intuition, an increase must be expected.

The computation time spent to find an optimal solution also increases rapidly with increase of the number of jobs and for fixed number of machines. As in Table 4.1, the computation times increase from 0.68 to 2.00 seconds and from 2.00 to 4.99 seconds when the problem's size changes from (6x3) to (7x3) and from (7x3) to (8x3), respectively. This is because the computation time depends on the number of nodes explored. The number of these nodes vary greatly from one problem to another of the same size. Tables 4.1 - 4.5 show how the number of nodes explored varies within each experiment. For example, the number of nodes explored in Experiment VII, shown in Table 4.3, ranges between 35 and 26,984. The variation in the computation time is also due to the variation in the elements of the processing time matrices of problems of the same size, since the processing times affect the amount of computational time involved. The reason is that the criterion used in this research is the schedule time which is a function of the processing times. The rate of change of increase in the computation time seems to increase rapidly as the number of jobs increases. The effects of variation in the number of jobs on both the number of nodes explored and the computation time are shown in Figures 4.1 - 4.3. The factors (β) by which the computation time changes are shown in Tables 4.7 and 4.8 for various bounding procedures.

The number of nodes explored to obtain an optimal solution increases as the number of machines increases with a fixed number of jobs. However, when problems were solved by LB III in all experiments but V and VI, the number of nodes explored decreases as the number of machines increases. LB V also show a similar reduction in the number of explored nodes except for Experiments II, III, V and VI. The increase in the number of nodes depends upon the quality of the various bounding procedures used to compute the lower-bounds. If the bounding procedure works efficiently, it will recognize the optimal solution by exploring small number of nodes. Otherwise, the optimal solution is reached after a number of back-tracking which in turn increases the number of nodes explored. By similar reasoning a decrease in the number of nodes may be expected. For example, as shown in Table 4.8, the number of nodes explored changes by factors about 2.34 and 0.83 when the problem size changes from (8x3) to (8x4) and from (8x4) to (8x5), respectively.

As the number of machines increases the computation times increases in most of the cases; however, in some experiments it appears to take negative changes. Comparing the results of Experiments VIII and IX in which each problem has 8 jobs, a reduction in the number of nodes was realized by all bounding procedures. For the same experiments, the computation times also seem to decrease for all bounding procedures except LB II, which shows an increase of 3.05 seconds. Similarly, in Experiments IV and V having problems of size (7x3) and (7x4), only LB III shows a decrease in computation time from 20.30 to 15.92 seconds when the number of machines changes from 3 to 4, respectively. It should be noted that for same experiments, the number of nodes decreases from 622.72 to 357.92. The increase in the number of machines causes an increase in the number of bounds computed since the lower-bound for each node is selected as the maximum value of the bounds for all machines. Thus, the computation time depends upon the number of machines.

In all bounding procedures, the computation time spent per node increases as the number of machines increases from M to M+1. For example in Experiments I and II, an increase of 0.0019 second is observed when number of machines changes from 3 to 4. This is due to the reason mentioned above.

In all bounding procedures, the computation time spent per node increases as the number of jobs increases from J to J+1. In Experiments I and IV having problems of sizes (6x3) and (7x3) respectively, the computation time per node increases from 0.0087 to 0.0097 second as the number of jobs changes from 6 to 7.

It is observed that the change in the computation time per node is more due to one unit change in the number of machines than one unit change in the number of jobs.

It should be pointed out that for some problems LB V produced better solutions. The LB V considers the permutations of unscheduled jobs on each pair of machines. These permutations change from one machine to another and help estimate a powerful lower-bound. Other bounding procedures can not reach that solution since they do not permute among the unscheduled jobs. For example, a flow shop problem of size (7x4) having the following processing time matrix:

$$T = \begin{bmatrix} 11 & 11 & 5 & 5 \\ 19 & 23 & 26 & 13 \\ 30 & 19 & 4 & 18 \\ 25 & 25 & 10 & 3 \\ 28 & 7 & 23 & 25 \\ 5 & 2 & 4 & 22 \\ 16 & 21 & 17 & 20 \end{bmatrix}$$

The optimal permutation-sequences and their schedule times of this problem, when solved using bounding procedures LB I - V, are as follows:

Bounding Procedure	Sequence	Schedule Time
LB I	{6 3 7 5 2 4 1}	169
LB II	{6 3 7 5 2 4 1}	169
LB III	{6 7 2 3 5 4 1}	169
LB IV	{6 7 2 3 5 4 1}	169
LB V	{6 7 2 3 5 1 4}	161

The number of solutions obtained by LB V found different from those obtained by the other bounding procedures are

Experiment	Problem Size	Number of Problems	Number of Different Solutions by LB V
II	(6x4)	50	16
III	(6x5)	50	20
V	(7x4)	50	13
VI	(7x5)	50	21
VIII	(8x4)	25	5
IX	(8x5)	25	11

ILLEGIBLE

THE FOLLOWING DOCUMENT (S) IS ILLEGIBLE DUE TO THE PRINTING ON THE ORIGINAL BEING CUT OFF

ILLEGIBLE

Table 4.1

Results for Bounding Procedure LB I

Experiment Number	Problem Size (JxM)	Number of Problems	Number of Nodes Explored: Minimum	Frequency of the Minimum	Maximum	Mean	Standard Deviation	Computation Time Spent*: Minimum	Maximum	Mean	Standard Deviation	Per Node
I	(6x3)	50	20	13	477	77.98	86.89	0.18	3.78	0.68	0.70	0.0087
II	(6x4)	50	20	7	440	117.00	112.08	0.23	4.89	1.24	1.14	0.0106
III	(6x5)	50	20	2	425	129.40	103.00	0.28	5.24	1.64	1.27	0.0127
IV	(7x3)	50	27	12	1465	205.26	302.91	0.29	14.08	2.00	2.85	0.0097
V	(7x4)	50	27	4	1322	243.44	297.67	0.34	15.69	2.89	3.43	0.0119
VI	(7x5)	50	27	3	2643	332.76	429.89	0.43	35.58	4.70	5.83	0.0141
VII	(8x3)	35	35	12	6148	440.71	1172.93	0.43	67.84	4.99	12.93	0.0113
VIII	(8x4)	25	35	3	4884	1031.56	1495.85	0.55	68.55	14.24	20.52	0.0138
IX	(8x5)	25	35	1	4793	862.04	999.23	0.64	75.68	13.83	15.80	0.0160

*Computation time in seconds (IBM 360/50 computer)

Table 4.2

Results for Bounding Procedure LB II

Experiment Number	Problem Size (JxM)	Number of Problems	Number of Nodes Explored					Computation Time Spent*				
			Minimum	Frequency of the Minimum	Maximum	Mean	Standard Deviation	Minimum	Maximum	Mean	Standard Deviation	Per Node
I	(6x3)	50	20	17	468	74.18	84.62	0.73	14.27	2.45	2.65	0.033
II	(6x4)	50	20	9	436	104.20	103.85	1.06	19.92	4.93	4.66	0.047
III	(6x5)	50	20	3	389	118.62	96.64	1.50	24.15	7.72	5.96	0.0651
IV	(7x3)	50	27	12	1350	193.12	291.00	1.16	48.51	7.34	10.56	0.0380
V	(7x4)	50	27	6	1210	227.16	286.45	1.67	63.79	12.09	14.67	0.0532
VI	(7x5)	50	27	3	2443	292.90	399.88	2.31	162.39	20.78	26.91	0.0709
VII	(8x3)	35	35	15	5504	410.31	1085.79	1.80	231.83	17.77	45.26	0.0433
VIII	(8x4)	25	35	3	4719	986.68	1447.81	2.55	274.07	58.64	84.03	0.0591
IX	(8x5)	25	35	1	4113	776.64	877.06	3.50	319.89	61.69	68.23	0.0791

*Computation time in seconds (IBM 360/50 computer)

Table 4.3

Results for Bounding Procedure LB III

Experiment Number	Problem Size (JxM)	Number of Problems	Number of Nodes Explored					Computation Time Spent*				
			Minimum	Frequency of the Minimum	Maximum	Mean	Standard Deviation	Minimum	Maximum	Mean	Standard Deviation	Per Node
I	(6x3)	50	20	17	791	140.44	187.57	0.73	19.40	3.88	4.67	0.027
II	(6x4)	50	20	9	418	112.04	116.22	1.00	4.14	4.39	15.53	0.039
III	(6x5)	50	20	9	527	100.38	107.91	1.26	22.63	4.92	4.63	0.049
IV	(7x3)	50	27	5	3210	622.72	800.53	1.30	100.69	20.30	24.53	0.032
V	(7x4)	50	27	6	2140	357.92	468.30	1.73	80.92	15.92	18.61	0.044
VI	(7x5)	50	27	1	2633	396.98	525.16	0.54	128.71	21.97	26.45	0.055
VII	(8x3)	35	35	10	26984	3417.71	6657.66	2.13	978.18	122.39	234.20	0.035
VIII	(8x4)	25	35	2	6875	1347.20	1975.57	2.85	346.96	68.99	94.98	0.051
IX	(8x5)	25	35	2	2490	638.80	626.48	3.58	169.06	43.20	41.12	0.067

*Computation time in seconds (IBM 360/50 computer)

Table 4.4

Results for Bounding Procedure LB IV

Experiment Number	Problem Size (JxM)	Number of Problems	Number of Nodes Explored					Computation Time Spent*				
			Minimum	Frequency of the Minimum	Maximum	Mean	Standard Deviation	Minimum	Maximum	Mean	Standard Deviation	Per Node
I	(6x3)	50	20	25	196	40.54	37.49	0.88	6.73	1.64	1.34	0.040
II	(6x4)	50	20	16	218	48.42	39.82	1.18	10.52	2.59	1.90	0.053
III	(6x5)	50	20	11	371	58.34	70.49	1.50	20.38	3.78	3.90	0.064
IV	(7x3)	50	27	17	980	109.22	177.92	1.50	38.69	5.10	7.19	0.046
V	(7x4)	50	27	10	664	104.44	120.14	2.00	36.02	6.48	6.58	0.062
VI	(7x5)	50	27	4	938	149.02	167.19	2.16	62.93	11.03	11.27	0.074
VII	(8x3)	35	35	17	3889	291.77	749.29	2.43	182.51	14.85	34.99	0.050
VIII	(8x4)	25	35	6	2992	424.84	748.87	3.23	194.75	29.27	48.58	0.068
IX	(8x5)	25	35	4	1254	308.76	346.24	4.07	97.40	26.18	26.71	0.084

*Computation time in seconds (IBM 360/50 computer)

Table 4.5

Results for Bounding Procedure LB V

Experiment Number	Problem Size (JxM)	Number of Problems	Number of Nodes Explored					Computation Time Spent*				
			Minimum	Frequency of the Minimum	Maximum	Mean	Standard Deviation	Minimum	Maximum	Mean	Standard Deviation	Per Node
I	(6x3)	50	20	14	566	106.30	133.43	0.40	9.90	1.86	2.23	0.017
II	(6x4)	50	20	9	323	95.32	84.82	0.59	8.38	2.46	2.09	0.025
III	(6x5)	50	24	1	432	123.26	85.32	0.92	14.01	4.17	2.73	0.033
IV	(7x3)	50	27	14	2706	314.92	534.39	0.61	50.90	6.22	10.18	0.019
V	(7x4)	50	27	3	1087	230.96	241.86	0.90	30.95	6.65	6.76	0.028
VI	(7x5)	50	27	3	2936	370.34	469.64	1.21	102.92	13.64	16.58	0.036
VII	(8x3)	35	35	14	12741	1240.37	3278.86	0.90	295.46	27.80	72.82	0.022
VIII	(8x4)	25	35	4	8610	1237.32	2082.05	1.31	264.59	38.91	64.24	0.031
IX	(8x5)	25	63	1	2731	570.52	577.00	3.00	113.64	23.98	23.92	0.042

*Computation time in seconds (IBM 360/50 computer)

Table 4.6

Results for Experiment X

Problem No.	Optimal Schedule Time	Number of Nodes Explored					Computation Time Spent[†]				
		LB I	LB II	LB III	LB IV	LB V	LB I	LB II	LB III	LB IV	LB V
1	230	141	141	*	77	77	3.17	12.68	#	12.21	2.96
2	197	77	77	*	77	*	2.06	8.23	#	12.26	#
3	257	206	77	192	192	206	4.84	7.92	20.84	24.20	7.92
4	201	77	77	77	77	224	2.02	8.14	10.74	12.25	8.39
5	229	77	77	77	77	77	1.89	7.91	10.47	11.91	3.13
6	227	*	*	98	86	*	#	#	11.89	12.49	#
7	222	93	93	*	93	*	2.15	8.85	#	13.02	#
8	262	77	77	77	77	77	1.92	7.88	10.61	11.81	3.43
9	186	522	458	*	237	*	10.31	34.53	#	25.60	#
10	248	86	77	408	77	*	2.03	7.65	28.78	11.74	#

[†] Computation time on IBM 360/50 computer in seconds.

[*] Solutions were not obtained because of computer time limitation.

[#] Computer was stopped after 1200 seconds without obtaining the solutions to problems labeled with *.

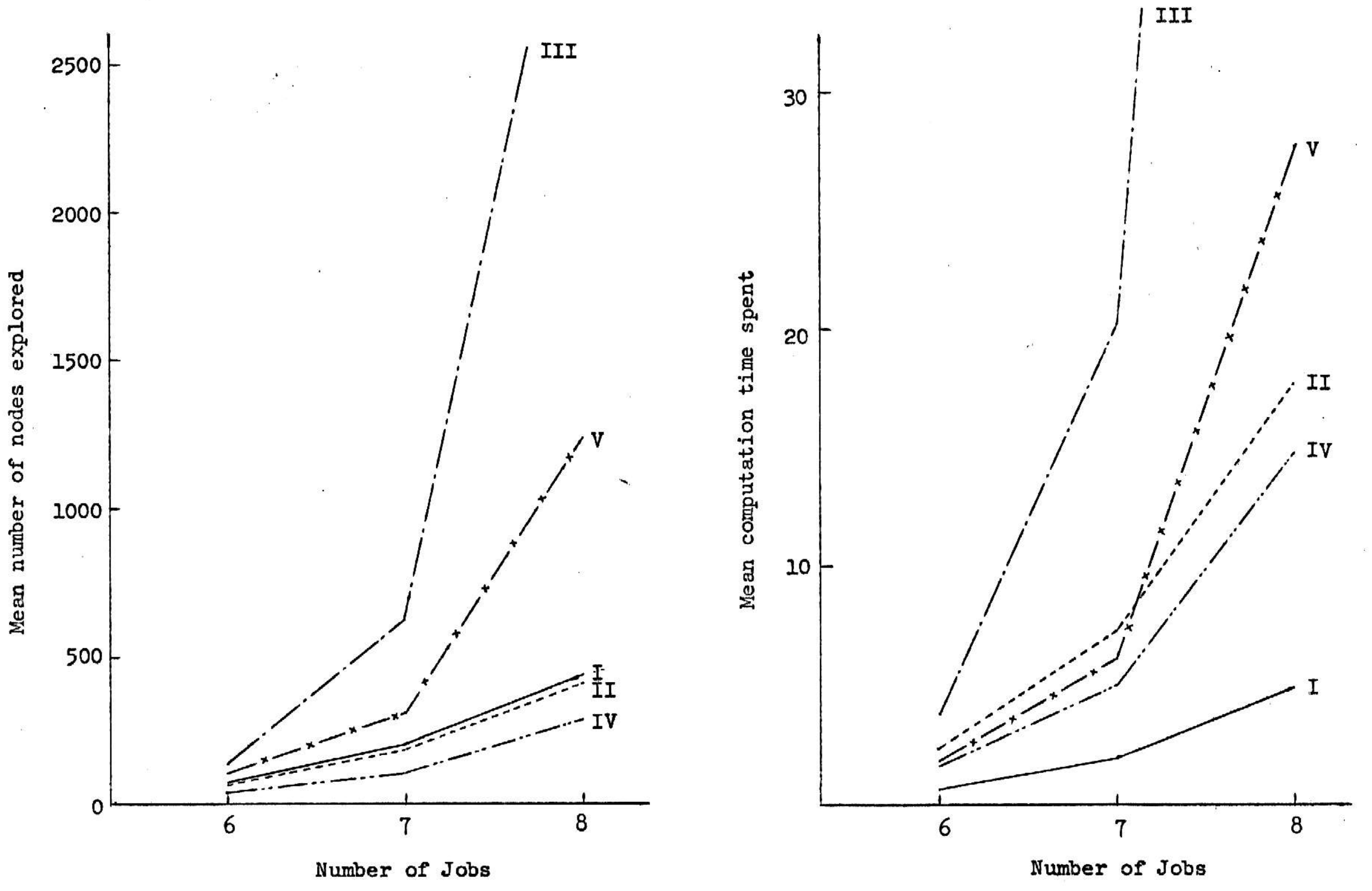

Fig. 4.1 The Effect of Problem Size Having Three Machines

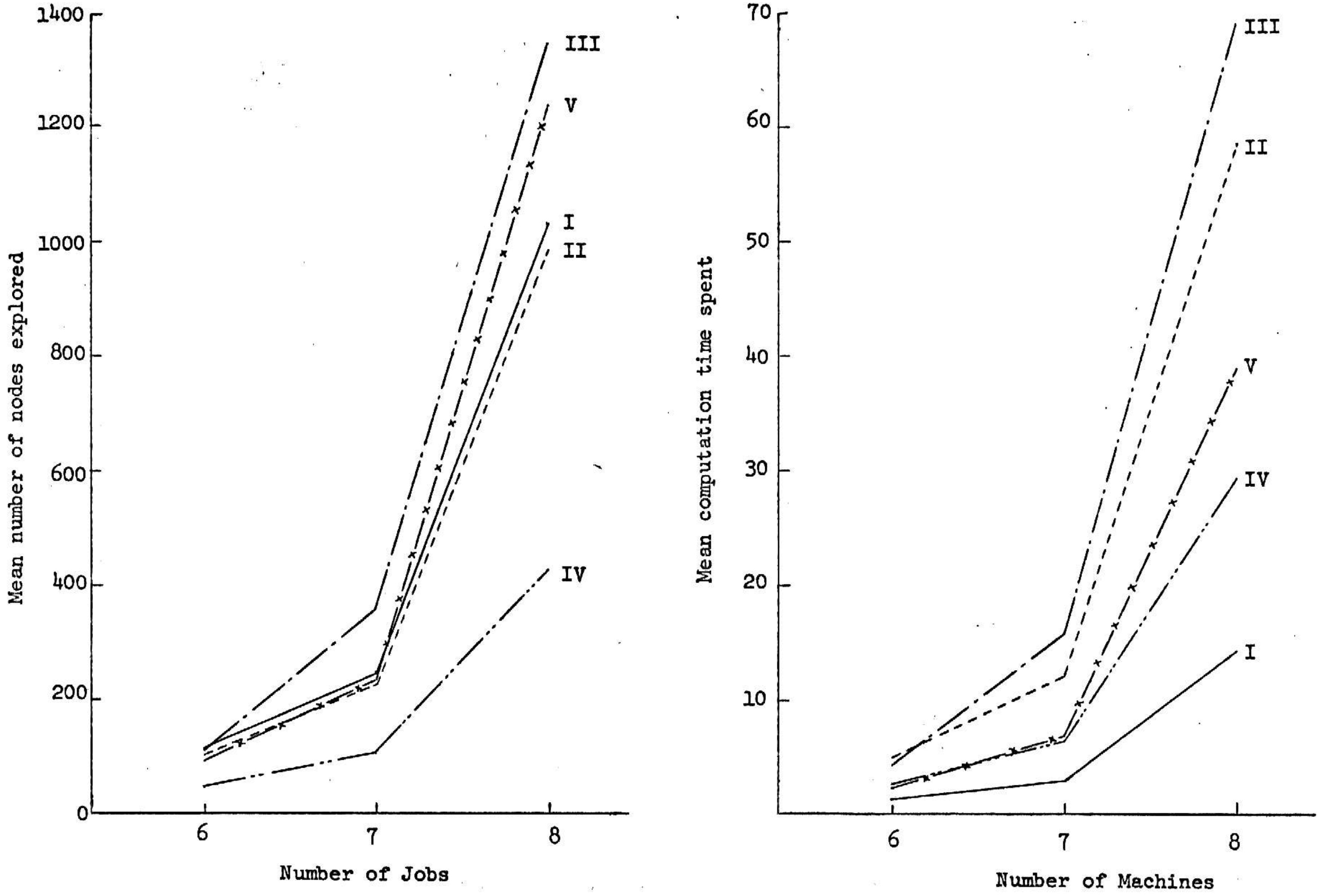

Fig. 4.2 The Effect of Problem Size Having Four Machines

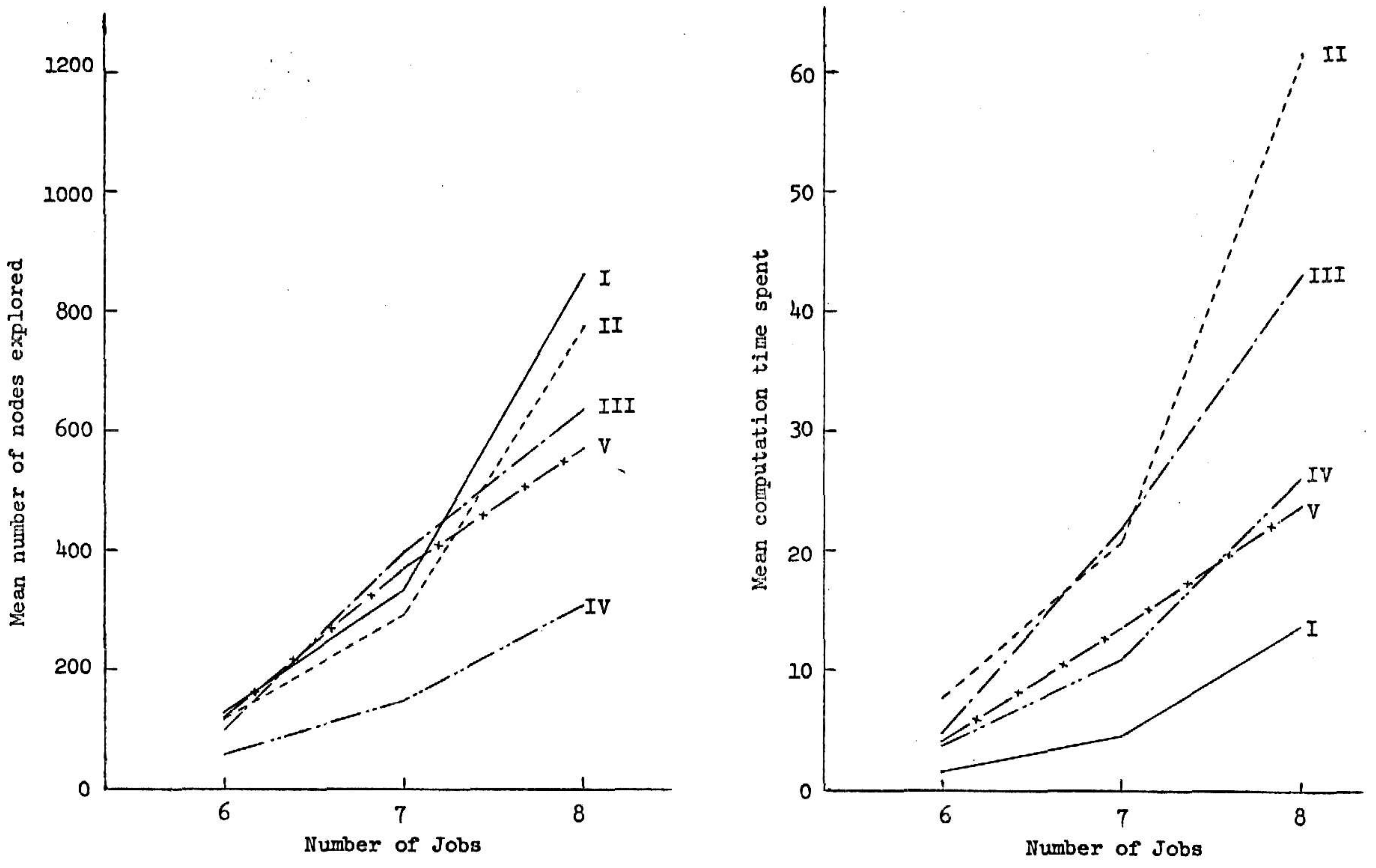

Fig. 4.3 The Effect of Problem Size Having Five Machines

Table 4.7

The Effect of Change in the Number of Jobs

Number of jobs changes from $J \rightarrow J+1$	Number of machines	LB I		LB II		LB III		LB IV		LB V	
		Factor* α	Factor* β	Factor α	Factor β	Factor α	Factor β	Factor α	Factor β	Factor α	Factor β
$6 \rightarrow 7$	3	2.63	2.94	2.60	2.99	4.43	5.23	2.69	3.10	2.96	3.34
	4	2.08	2.33	2.18	2.45	3.19	3.62	2.15	2.50	2.42	2.70
	5	2.57	2.86	2.46	2.69	3.95	4.46	2.55	2.91	3.00	3.27
$7 \rightarrow 8$	3	2.14	2.49	2.12	2.42	5.48	6.02	2.67	2.91	3.93	4.46
	4	4.23	4.92	4.34	4.85	3.76	4.33	4.06	4.51	5.35	5.85
	5	2.59	2.94	2.65	2.96	1.60	1.96	2.07	2.37	1.54	1.75

* α factor by which the mean number of explored nodes changes.

β factor by which the mean computation time changes.

Table 4.8

The Effects of Change in the Number of Machines

Number of machines changes from M → M+1	Number of jobs	LB I		LB II		LB III		LB IV		LB V	
		Factor* α	Factor* β	Factor α	Factor β	Factor α	Factor β	Factor α	Factor β	Factor α	Factor β
3 → 4	6	1.50	1.82	1.40	2.01	0.79	1.13	1.19	1.57	0.89	1.32
	7	1.18	1.44	1.17	1.64	0.57	0.78	0.95	1.27	0.73	1.06
	8	2.34	2.85	2.40	3.29	0.39	0.56	1.45	1.97	0.99	1.39
4 → 5	6	1.10	1.32	1.13	1.56	0.89	1.12	1.20	1.45	1.29	1.69
	7	1.36	1.62	1.28	1.71	1.10	1.38	1.42	1.70	1.60	2.05
	8	0.83	0.97	0.78	1.05	0.47	0.62	0.72	0.89	0.46	0.61

*
α factor by which the mean number of explored nodes changes.

β factor by which the mean computation time changes.

CHAPTER V

SUMMARY AND CONCLUSION

The basic objective of this report is to analyze and compare various bounding procedures used in branch-and-bound technique to obtain the lower-bounds for the solution of flow shop scheduling problem. In flow shop certain jobs are performed on various machines in an identical routing. This problem is first defined and formulated mathematically. The branch-and-bound concept, and branching, bounding and back-tracking processes are also discussed in detail. The bounding procedures subject to comparison in this report are analyzed. Various computational experiments were performed to carry out the investigation. The comparison was based on both the number of nodes explored and the computation time spent in obtaining an optimal solution.

For convenience, the summarized results of Experiments I - IX are given in Table 5.1. This will help analyze the results vertically and horizontally. The vertical analysis means the study of the effects in the changes of the number of jobs or machines in each bounding procedure. However, by horizontal analysis, is meant the study of the performance (number of nodes explored and computation time) of the various bounding procedures in each experiment.

In analyzing the results of all bounding procedures, most significant observations are obtained -

1. The number of nodes explored and the computation time increases with the increase of the number of jobs.
2. The computation time is proportional to the number of nodes explored.

3. In general, the number of explored nodes, and, thus the computation time increase as the number of machines increases with a fixed number of jobs; however, this was not the case in few experiments.
4. Computation time required to explore a node depends upon the number of machines.
5. Computation time is a better criterion for comparing various bounding procedures, since the computation time required to explore a node varies from one bounding procedure to another. For example, in Table 5.1, LB I and LB IV explored on the average 77.98 and 40.54 nodes, respectively, to solve a problem having 6 jobs and 3 machines. At first glance, it appears that on the average LB IV is more efficient than LB I, since it reached the optimal after exploring comparatively less number of nodes. On the other hand, LB I and LB IV require on the average 0.68 and 1.64 seconds to solve a (6x3) problem. Thus, the LB I is more efficient than LB IV, though it requires to explore more nodes. The reason is that LB I spends less computation time in calculating a lower-bound for each node than that by LB IV.
6. The ranking according to the number of explored nodes and the computational time are shown in Table 5.2 and 5.3. Bounding procedure LB IV is ranked first according to the number of explored nodes in all experiments; however, it is ranked second in Table 5.3. This is because the computation of lower-bound for a node by LB IV requires the lower-bounds computed by LB I and LB III, and, hence, more computation effort is required. LB III is generally ranked last according to both the number of explored nodes and the computation time. As

Table 5.1

Summary of Results for All Bounding Procedures

Experiment Number	Problem Size	Mean Number of Nodes Explored					Mean Computation Time Spent*				
		LB I	LB II	LB III	LB IV	LB V	LB I	LB II	LB III	LB IV	LB V
I	(6x3)	77.98	74.18	140.44	40.54	106.30	0.68	2.45	3.88	1.64	1.86
IV	(7x3)	205.26	193.12	622.72	109.22	314.92	2.00	7.34	20.30	5.10	6.22
VII	(8x3)	440.71	410.31	3417.71	291.77	1240.37	4.99	17.77	122.39	14.85	27.80
II	(6x4)	117.00	104.20	112.04	48.42	95.32	1.24	4.93	4.39	2.59	2.46
V	(7x4)	243.44	227.16	357.92	104.44	230.96	2.89	12.09	15.92	6.48	6.65
VIII	(8x4)	1031.56	986.68	1347.20	424.84	1237.32	14.24	58.64	68.99	29.27	38.91
III	(6x5)	129.40	118.62	100.38	58.34	123.26	1.64	7.72	4.92	3.78	4.17
VI	(7x5)	332.76	292.90	396.98	149.02	370.34	4.70	20.78	21.97	11.03	13.64
IX	(8x5)	862.04	776.64	638.80	308.76	570.52	13.83	61.69	43.20	26.18	23.98
X	(12x3)	150.66	128.22	154.83	107.00	132.20	3.37	11.53	15.55	14.74	5.16

*Computation time on IBM 360/50 computer, in seconds.

Table 5.2

Rank of Bounding Procedures Based on Number of Explored Nodes

Rank	Experiment								
	(6x3)	(6x4)	(6x5)	(7x3)	(7x4)	(7x5)	(8x3)	(8x4)	(8x5)
1	LB IV	LB IV	LB IV	LB IV	LB IV	LB IV	LB IV	LB IV	LB IV
2	LB II	LB V	LB III	LB II	LB II	LB II	LB II	LB II	LB V
3	LB I	LB II	LB II	LB I	LB V	LB I	LB I	LB I	LB III
4	LB V	LB III	LB V	LB V	LB I	LB V	LB V	LB V	LB II
5	LB III	LB I	LB I	LB III	LB III	LB III	LB III	LB III	LB I

Table 5.3

Rank of Bounding Procedures Based on Computation Time

Rank	Experiment								
	(6x3)	(6x4)	(6x5)	(7x3)	(7x4)	(7x5)	(8x3)	(8x4)	(8x5)
1	LB I	LB I	LB I	LB I	LB I	LB I	LB I	LB I	LB I
2	LB IV	LB V	LB IV	LB IV	LB IV	LB IV	LB IV	LB IV	LB V
3	LB V	LB IV	LB V	LB V	LB V	LB V	LB II	LB V	LB IV
4	LB II	LB III	LB III	LB II	LB II	LB II	LB V	LB II	LB III
5	LB III	LB II	LB II	LB III	LB III	LB III	LB III	LB III	LB II

clear from Table 5.3, LB V and LB II are ranked third and fourth, respectively in most of the experiments. However, these rankings vary from one experiment to another as shown in tables.

APPENDIX A

In this appendix, an algorithm for finding an optimal sequence for two-machine flow shop problem is stated. This algorithm is based on Johnson's criterion [9] for two machines which can be restated as: job j_r must precedes job j_s on machines m and m+1 in order to minimize the schedule time, if

$$\min \left(t_{j_r m}, \; t_{j_s m+1} \right) \leq \min \left(t_{j_r m+1}, \; t_{j_s m} \right)$$

holds.

The algorithm may be summarized in the following steps.

Step 1: Arrange the processing times of the jobs on machines m and m+1, as follows:

Job Index	Machine m	Machine m+1
1	t_{1m}	t_{1m+1}
2	t_{2m}	t_{2m+1}
.	.	.
.	.	.
.	.	.
J	t_{Jm}	t_{Jm+1}

Step 2: Examine all processing times for the minimum value.

2.1 If the minimum processing time is t_{jm}, schedule the corresponding job first on machine m.

2.2 If the minimum processing time is t_{jm+1}, schedule the corresponding job last on machine m.

2.3 If a tie exists among the processing times on the same machine, schedule the job with the smallest designation first.

2.4 If a tie exists for the same job on both machines, consider it as in step 2.1.

Step 3: Cross off the job just assigned and repeat step 2 on the reduced set of processing times.

To illustrate the algorithm, the processing time matrix of the sample problem solved in Section 3.6 is reproduced below

$$T = \begin{bmatrix} 9 & 13 & 6 \\ 7 & 7 & 20 \\ 6 & 4 & 8 \\ 8 & 3 & 10 \\ 20 & 7 & 2 \\ 10 & 2 & 13 \end{bmatrix}$$

To find the preliminary sequence on machines 1 and 2, the first two columns of the processing time matrix are considered. Thus, according to step 1 the processing times are arranged as follows:

j	t_{j1}	t_{j2}
1	9	13
2	7	7
3	6	4
4	8	3
5	20	7
6	10	2

According to step 2, the minimum processing time is 2 units for job 6 on machine 2. Therefore, job 6 is scheduled in the last sequence-position.

According to step 3, job 6 is crossed off, since it has been already scheduled.

Repeating steps 2-3 on the reduced set of processing times, the preliminary sequence {2 1 5 3 4 6} is obtained. Similarly, the preliminary sequence on machines 2 and 3 is found to be {6 4 3 2 1 5}.

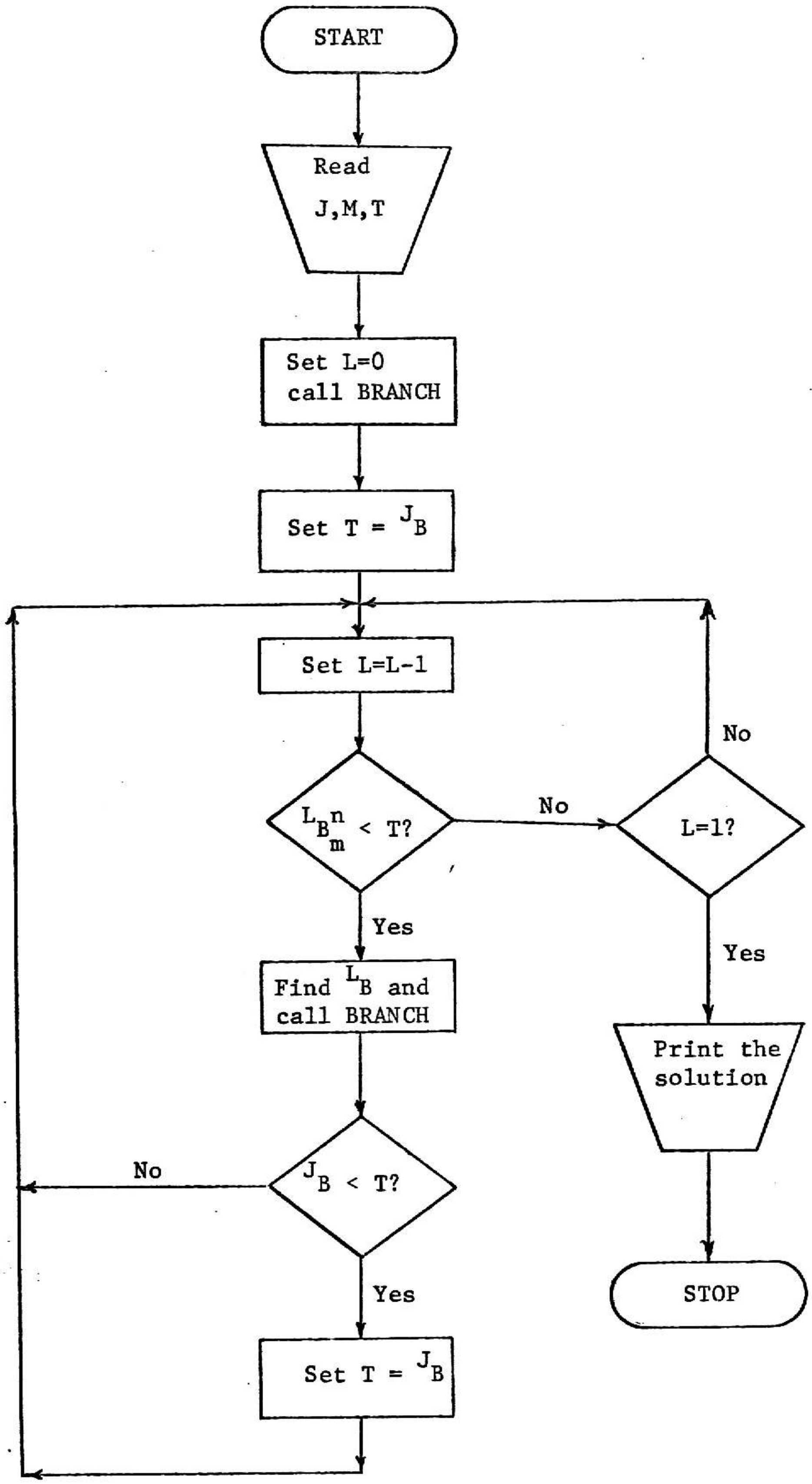

Figure B.1

Flow Chart for Branch-and-Bound Algorithm

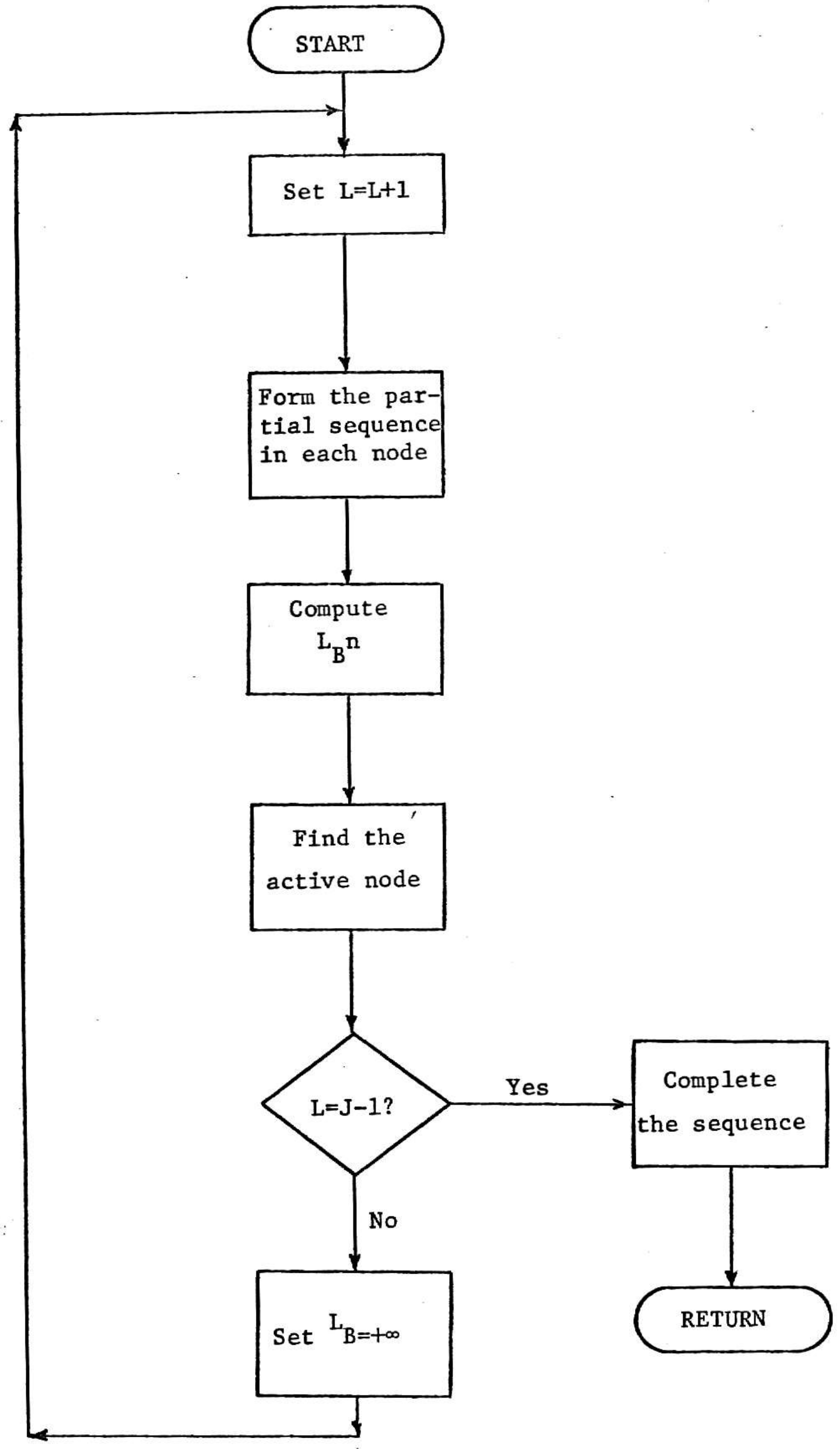

Figure B.2

Flow Chart for Branch Subroutine

```
$JOB            MNQ,RUN=CHECK
C
C      PROGRAMMED BY  ..............  M. NAWA QURAISHI
C
C      THIS PROGRAM CONSISTS OF MAIN AND SEVEN SUBROUTINES.
C      THE BRANCH-AND-BOUND ALGORITHM OF SECTION 2.3 IS
C      PROGRAMMED IN FORTRAN IV . THE BACK-TRACKING PROCESS
C      GUARANTEES THE OPTIMALITY. THIS PROGRAM IS COMPLETELY
C      GENERAL IN THE SENSE IT CAN HANDLE ANY NUMBER OF JOBS
C      AND ANY NUMBER OF MACHINES. IT CAN READ AS WELL GENERATE
C      THE PROCESSING TIMES.
C
C      CONTROL CARD FOR THIS PROGRAM CONTAINS FOLLOWING
C      VARIABLES-
C
C      JOBS          NUMBER OF JOBS
C
C      MACH          NUMBER OF MACHINES
C
C      IREAD         DATA ORIGINATION
C                    1 = READ PROCESSING TIMES FROM CARDS
C                    0 = GENERATE PROCESSING TIMES
C
C      LIMIT1        USED ONLT WHEN IREAD = 0,
C                    SMALLEST VALUE IN THE INTERVAL ( A,B )
C
C      LIMIT2        USED ONLY WHEN IREAD = 0,
C                    LARGEST VALUE IN THE INTERVAL ( A,B )
C
C      ISKIP         CONDITIONAL PRINT OUT
C                    1 = PRINT THE MINIMUM LOWER-BOUND FOR
C                    EACH NODE AT EACH LEVEL
C                    0 = DO NOT PRINT THE MINIMUM LOWER-BOUNDS
C
C      IPRINT        CONDITIONAL PRINT OUT
C                    1 = PRINT THE ELEMENTS FROM WHICH LOWER-
C                    BOUND IS COMPUTED, AND THE LOWER-BOUND
C                    FOR EACH MACHINE
C                    0 = DO NOT PRINT THE ABOVE
C
C      ICARD         CONDITIONAL PUNCHED OUTPUT
C                    1 = PUNCHED OUTPUT NEEDED
C                    0 = NO PUNCHED OUTPUT NEEDED
C
C      NPROB         NUMBER OF PROBLEMS
C
C      IY            STARTING POINT FOR RANDOM NUMBER
C                    GENERATOR
C
C      NBF           FIRST BOUND
C
C      NBL           LAST BOUND
C
       COMMON IT(20,10),MACH,ISTMIN,KOUNT,IPRINT,IB
       COMMON J(20),ILB(20,20),KONTMB,KONTJB,KONTMJ
       COMMON NREPET,JSQ(20,20,20)
  1500 FORMAT(9I5,I10,2I5)
  1501 FORMAT(10I5)
  1502 FORMAT(1H1,10X,'THIS IS THE PROCESSING TIME MATRIX',
      160X,'PROBLEM NUMBER',I3)
```

```
 1503 FORMAT(1H ,10I5)
 1511 FORMAT(1H ,10X,'COMPUTATION TIME =',F10.2,'  SECONDS')
 1512 FORMAT(1H ,'THIS SOLUTION IS OBTAINED BY BOUND',I1)
 1513 FORMAT(1H ,10X,'TOTAL NUMBER OF NODES FOR WHICH ',
     1'COMPOSITE BOUND IS PICKED UP FROM MACHINE BASED BOUND',
     1' ONLY = ',I9)
 1514 FORMAT(1H ,10X,'TOTAL NUMBER OF NODES FOR WHICH ',
     1'COMPOSITE BOUND IS PICKED UP FROM JOB BASED BOUND ',
     1'ONLY = ',I9)
 1515 FORMAT(1H ,10X,'TOTAL NUMBER OF NODES FOR WHICH COMP',
     1'OSITE BOUND IS PICKED UP FROM BOTH BOUNDS = ',I9)
 1516 FORMAT(1H0,130(1H*))
 1517 FORMAT(1H ,10X,'NUMBER OF TIMES THE SCHEDULE TIME IS',
     1' IMPROVED BY BACK-TRACKING = ',I9)
 1518 FORMAT(1H ,10X,'TOTAL NUMBER OF NODES EXPLORED =',I9)
 1519 FORMAT(1H0,130(1H*))
 1520 FORMAT(I10,F10.2,I10)
 1521 FORMAT(I10,F10.2,4I10)
 1600 FORMAT(1H ,10X,'SCHEDULE TIME = ',I5,10X,'OPTIMAL ',
     1'SEQUENCE = ',20I4)
      READ(1,1500) JOBS,MACH,IREAD,LIMIT1,LIMIT2,ISKIP,
     1IPRINT,ICARD,NPROB,IY,NBF,NBL
      DO 14 NP = 1,NPROB
      IF(IREAD.EQ.0)                    GO TO 990
C
C     READ PROCESSING TIME MATRIX FROM CARDS IF IREAD = 1
C
      DO 1 JC = 1,JOBS
    1 READ(1,1501) (IT(JC,M),M = 1,MACH)
      GO TO 992
C
C     GENERATE PROCESSING TIME MATRIX IT(J,M)
C
  990 DO 991 K = 1,MACH
      DO 991 JA = 1,JOBS
      IT(JA,K) = BANDNO(IY)*LIMIT2 + LIMIT1
  991 CONTINUE
  992 CONTINUE
C
C     PRINT THE PROCESSING TIME MATRIX
C
      WRITE(3,1502) NP
      DO 993 JC = 1,JOBS
  993 WRITE(3,1503)(IT(JC,M), M = 1,MACH)
      JOB = JOBS - 1
      DO 15 IB = NBF,NBL
      ITERM = 0
      LV = 0
      KOUNT = 0
      KONT = 0
      KONTMJ = 0
      KONTMB = 0
      KONTJB = 0
      NREPET = 0
      ISTMIN = 99999
      CALL INTIME(ITIME1)
      CALL BRANCH (LV,JOBS,KONT,NACTIV,ITERM)
      ISTMIN = ILB(JOBS,1)
      DO 5 K = 1,JOB
      JSQ(JOBS,1,K) = JSQ(JOB,NACTIV,K)
    5 CONTINUE
C
C     BACK TRACKING
C
 1200 LV = LV-1
      ITERM = 0
      ND = JOBS + 1 - LV
      DO 700  N = 1,ND
      IF(ISTMIN-ILB(LV,N))700,700,710
  700 CONTINUE
      IF(LV.GT.1)                            GO TO 1200
      IF(LV.EQ.1)                            GO  TO  1300
  710 LBX = ILB(LV,1)
      NACTIV = 1
      DO 730  NOD = 2,ND
      IF(LBX-ILB(LV,NOD))730,730,750
  750 LBX = ILB(LV,NOD)
      NACTIV = NOD
  730 CONTINUE
      ILB(LV,NACTIV) = 99999
      CALL BRANCH (LV,JOBS,KONT,NACTIV,ITERM)
      IF(ITERM.EQ.1)                         GO TO 1200
      IF(ISTMIN-ILB(JOBS,1))760,760,770
  770 ISTMIN = ILB(JOBS,1)
      DO 771 K=1,JOB
      JSQ(JOBS,1,K)=JSQ(JOB,NACTIV,K)
  771 CONTINUE
  760 GO TO 1200
 1300 WRITE(3,1516)
      WRITE(3,1512) IB
      DO 984 K=1,JOB
      J(K)=JSQ(JOB,NACTIV,K)
  984 CONTINUE
      DO 985 K=1,JOBS
      DO 986 KS = 1,JOB
      IF(K.EQ.J(KS))                         GO TO 985
  986 CONTINUE
      JSQ(JOBS,1,JOBS)=K
      GO TO 987
  985 CONTINUE
  987 CONTINUE
      CALL  INTIME(ITIME2)
      COTIME = ( ITIME2-ITIME1 )/100.
      WRITE(3,1600) ISTMIN,(JSQ(JOBS,1,K),K=1,JOBS)
      WRITE(3,1518) KOUNT
      WRITE(3,1511) COTIME
      WRITE(3,1517) KONT
      IF(IB.EQ.5)                            GO TO 982
      IF(ICARD.EQ.0)                         GO TO 994
      WRITE(2,1520) KOUNT,COTIME,KONT
  994 CONTINUE
      GO TO 16
  982 WRITE(3,1513) KONTMB
      WRITE(3,1514) KONTJB
      WRITE(3,1515) KONTMJ
      IF(ICARD.EQ.0)                         GO TO 995
      WRITE(2,1521) KOUNT,COTIME,KONT,KONTMB,KONTJB,KONTMJ
  995 CONTINUE
   16 CONTINUE
   15 CONTINUE
   14 CONTINUE
      STOP
      END
      SUBROUTINE BRANCH (LV,JOBS,KONT,NACTIV,ITERM)
      COMMON IT(20,10),MACH,ISTMIN,KOUNT,IPRINT,IB
      COMMON J(20),ILB(20,20),KONTMB,KONTJB,KONTMJ
      COMMON NREPET,JSQ(20,20,20)
 5000 FORMAT(1H-,5X,5HLEVEL,15X,4HNODE,10X,16HPARTIAL SEQUENCE)
 5100 FORMAT(1H0,7X,I2,18X,I2,16X,20I2)
      L=LV+1
      JOB =JOBS-1
      DO 10  LL = L,JOB
      LLP=LL-1
      NN=JOBS+1-LL
      KOUNT = KOUNT + NN
      IF(LL.GT.1)                        GO TO 1000
C
C     FORMING THE PARTIAL SEQUENCES AT LEVEL 1 AND COMPUTING
C     THE LOWER BOUND FOR EACH NODE.
C
      DO 25  N = 1,NN
      JSQ(LL,N,1) = N
      IF(IPRINT) 26,26,27
   27 WRITE(3,5000)
      WRITE(3,5100) LL,N,JSQ(LL,N,1)
   26 IF(IB.EQ.1)                        GO TO 701
      IF(IB.EQ.2)                        GO TO 702
      IF(IB.EQ.3)                        GO TO 703
      IF(IB.EQ.4)                        GO TO 704
      IF(IB.EQ.5)                        GO TO 705
  701 CALL BOUND1 (NN,LL,JOBS,N)
      GO TO 25
  702 CALL BOUND2 (NN,LL,JOBS,N)
      GO TO 25
  703 CALL BOUND3 (NN,LL,JOBS,N)
      GO TO 25
  704 CALL BOUND4 ( NN,LL,JOBS,N)
      GO TO 25
  705 CALL BOUND5 ( NN,LL,JOBS,N)
      GO TO 25
   25 CONTINUE
      GO TO 100
C
C     FORMING THE PARTIAL SEQUENCES AT LEVEL LL ( LLIS GREATER
C     THAN 1 ) AND COMPUTING THE LOWER BOUND FOR EACH NODE
C
 1000 DO 50  K = 1,LLP
      J(K)= JSQ(LLP,NACTIV,K)
   50 CONTINUE
      DO 60 K = LL,JOBS
      KK = K-1
      DO 70 JUNK = 1,JOBS
      DO 80 I = 1,KK
      IF(JUNK.EQ.J(I))                   GO TO 70
   80 CONTINUE
      J(K) = JUNK
      GO TO 60
   70 CONTINUE
   60 CONTINUE
      KA = LL-1
      DO 51 N = 1,NN
      KA = KA + 1
      JSQ(LL,N,LL) = J(KA)
   51 CONTINUE
      DO 52 N=1,NN
      DO 53 K = 1,LLP
   53 JSQ(LL,N,K) = J(K)
   52 CONTINUE
      DO 90 N = 1,NN
      IF(IPRINT) 91,91,92
   92 WRITE(3,5000)
      WRITE(3,5100) LL,N,(JSQ(LL,N,K),K=1,LL)
   91 IF(IB.EQ.1)                          GO TO 61
      IF(IB.EQ.2)                          GO TO 62
      IF(IB.EQ.3)                          GO TO 63
      IF(IB.EQ.4)                          GO TO 64
      IF(IB.EQ.5)                          GO TO 65
   61 CALL BOUND1 (NN,LL,JOBS,N)
      GO TO 90
   62 CALL BOUND2 (NN,LL,JOBS,N)
      GO TO 90
   63 CALL BOUND3 (NN,LL,JOBS,N)
      GO TO 90
   64 CALL BOUND4(NN,LL,JOBS,N)
      GO TO 90
   65 CALL BOUND5(NN,LL,JOBS,N)
      GO TO 90
   90 CONTINUE
C
C     SEARCH FOR THE ACTIVE NODE AT LEVEL LL
C
  100 LLB = ILB(LL,1)
      NACTIV = 1
      DO 30  N = 2,NN
      IF(LLB-ILB(LL,N))30,30,40
   40 LLB = ILB(LL,N)
      NACTIV = N
   30 CONTINUE
      IF(ILB(LL,NACTIV)-ISTMIN)1100,1400,1400
 1400 LV = LL
      LTERM = 1
      GO TO 15
 1100 IF(LL.EQ.JOB)                        GO TO 35
      ILB(LL,NACTIV) = 99999
      GO TO 10
   35 ILB(LL+1,1) = ILB(LL,NACTIV)
   10 CONTINUE
      LV = LL
      KONT = KONT + 1
   15 RETURN
      END
      SUBROUTINE BOUND1 (NN,LL,JOBS,N)
      DIMENSION ITRR(10),ITRMM(10),ICT(20,20)
      COMMON IT(20,10),MACH,ISTMIN,KOUNT,IPRINT,IB
      COMMON J(20),ILB(20,20),KONTMB,KONTJB,KONTMJ
      COMMON NREPET,JSQ(20,20,20)
 1700 FORMAT(1H ,10X,'JOB',10X,'M1      M2      M3      M4      M5',
     1'      M6      M7      M8      M9      M10 ')
 1710 FORMAT(1H ,10X,I2,6X,10I7)
 1720 FORMAT(1H ,'       *****  COMPLETION  TIME  MATRIX  *****')
 1740 FORMAT(1H ,10X,'SECOND TERM',10X,'MACHINE')
 1750 FORMAT(1H ,13X,I4,14X,I4)
 1760 FORMAT(1H ,11X,'THIRD TERM',10X,'MACHINE')
 1770 FORMAT(1H ,10X,'ON MACHINE ',I2,'  LOWER BOUND IS  ',I4)
 1780 FORMAT(1H ,11HLOWER BOUND,15X,5HLEVEL,15X,4HNODE)
 1790 FORMAT(1H ,4X,I5,19X,I2,18X,I2)
C
C
      IF(IB.EQ.5)                       GO TO 155
      DO 120 K = 1,LL
      J(K) = JSQ(LL,N,K)
  120 CONTINUE
C
C     SPLITTING THE JOBS WHICH ARE NOT INCLUDED IN THE
C     PARTIAL SEQUENCE
C
      JU = LL+1
      DO 160 K = JU,JOBS
      KK = K-1
      DO 150 JUSK = 1,JOBS
      DO 130 I = 1,KK
      IF(JUSK.EQ.J(I))                  GO TO 150
  130 CONTINUE
      J(K) = JUSK
      GO TO 160
  150 CONTINUE
  160 CONTINUE
  155 CONTINUE
C
C     FORMING THE COMPLETION TIME MATRIX FOR JOBS INCLUDED IN
C     THE PARTIAL SEQUENCE
C
      DO 400 I=1,LL
      JJ = J(I)
      IF(I.GT.1)                        GO TO 430
      DO 410  M = 1,MACH
      IF(M.GT.1)                        GO TO 420
      ICT(JJ,M) = IT(JJ,M)
      ICTF = ICT(JJ,M)
      GO TO 410
  420 ICT(JJ,M) = ICTF + IT(JJ,M)
      ICTF = ICT(JJ,M)
  410 CONTINUE
      GO TO 400
  430 II = I-1
      JP = J(II)
      DO 500 M = 1,MACH
      IF(M.GT.1)                        GO TO 510
      ICT(JJ,M) = ICT(JP,M)+IT(JJ,M)
      GO TO 500
  510 MM = M-1
```

```
      IF(ICT(JJ,MM).GE.ICT(JP,M))         GO TO 520
      ICT(JJ,M) = ICT(JP,M)+IT(JJ,M)
      GO TO 500
  520 ICT(JJ,M) = ICT(JJ,MM)+IT(JJ,M)
  500 CONTINUE
  400 CONTINUE
C
C     PRINT THE COMPLETION TIME MATRIX IF IPRINT = 1
C
      IF(IPRINT.EQ.0)                     GO TO 105
      WRITE(3,1720)
      WRITE(3,1700)
      DO 106 K = 1,LL
      JK = J(K)
  106 WRITE(3,1710) JK,(ICT(JK,M),M = 1,MACH)
  105 CONTINUE
C
C     SECOND TERM IN THE BROWN AND LOMNICKI FORMULA
C
      KU = LL + 1
      DO 185  M = 1,MACH
      ITR = 0
      DO 180  I = KU,JOBS
      K = J(I)
  180 ITR = ITR + IT(K,M)
      ITRR(M) = ITR
  185 CONTINUE
      IF(IPRINT.EQ.0)                     GO TO 108
      WRITE(3,1740)
      DO 109 M = 1,MACH
  109 WRITE(3,1750) ITRR(M),M
  108 CONTINUE
C
C     THIRD TERM IN THE BROWN AND LOMNICKI FORMULA
C
      MMM = MACH - 1
      DO 250  M = 1,MMM
      MA = M+1
      DO 200  I = KU,JOBS
      ITRM = 0
      DO 190  MR = MA,MACH
      K = J(I)
  190 ITRM = ITRM + IT(K,MR)
      IF(I.GT.KU)                         GO TO 210
      ITRMIN = ITRM
  210 IF(ITRM-ITRMIN)220,200,200
  220 ITRMIN = ITRM
  200 CONTINUE
      ITRMM(M) = ITRMIN
  250 CONTINUE
C
C     PRINT THE THIRD TERM FOR ALL MACHINES , EXCEPT THE LAST.
C
      IF(IPRINT.EQ.0)                     GO TO 110
      WRITE(3,1760)
      DO 111 M = 1,MMM
  111 WRITE(3,1750) ITRMM(M),M
  110 CONTINUE
C
C     COMPUTATION OF LOWER BOUND
```

```
      C
            DO 600 M = 1,MACH
            JL = J(LL)
            IF(M.EQ.MACH)                      GO TO 610
            LB = ICT(JL,M) + ITRR(M) + ITRMM(M)
            IF(M.GT.1)                         GO TO 620
            ILB(LL,N) = LB
            GO TO 112
        610 LB = ICT(JL,M) + ITRR(M)
        620 IF(LB-ILB(LL,N))112,112,630
        630 ILB(LL,N) = LB
      C
      C     PRINT THE LOWER BOUNDS FOR EACH MACHINE AT LEVEL LL.
      C
        112 IF(IPRINT.EQ.0)                    GO TO 600
            WRITE(3,1770) M,LB
        600 CONTINUE
      C
      C     PRINT THE LOWER BOUND AT LEVEL  LL , IF ISKIP = 1.
      C
            IF(ISKIP.EQ.0)                     GO TO 113
            WRITE(3,1780)
            WRITE(3,1790) ILB(LL,N),LL,N
        113 CONTINUE
            RETURN
            END
      SUBROUTINE BOUND2 (NN,LL,JOBS,N)
      DIMENSION JOS(10,20),JPS(10,20),ICT(20,20),ITXMM(10)
      COMMON IT(20,10),MACH,ISTMIN,KOUNT,IPRINT,IB
      COMMON J(20),ILB(20,20),KONTMB,KONTJB,KONTMJ
      COMMON NREPET,JSQ(20,20,20)
 3000 FORMAT(1H ,'SEQUENCE ON MACHINE',I2,'AND',I2,'IS',20I2)
 3010 FORMAT(1H ,'    COMPLETION  TIME  MATRIX')
 3020 FORMAT(1H ,20X,'JOB',15X,'MACHINE',I2,15X,'MACHINE',I3)
 3030 FORMAT(1H ,20X,I2,17X,I4,21X,I4)
 3040 FORMAT(1H ,'  SECOND TERM ON MACHINE',I2,'AND',I3,'=',I5)
 3050 FORMAT(1H ,'LOWER BOUND ON MACHINE',I2,'AND',I3,'IS',I5)
 3060 FORMAT(1H ,'LOWER BOUND AT LEVEL',I2,'FOR NODE',I2,'=',I5)
C
C     FORMING THE SEQUENCE ON MACHINE M AND M+1 BY USING THE
C     JOHNSON'S CRITERION.
C
      IF(NREPET.EQ.1)                     GO TO 692
      MM = MACH - 1
      DO 690 M = 1,MM
      DO 691 JE = 1,JOBS
  691 JOS(M,JE) = 0
      JX = JOBS + 1
      JJ = 1
      IJUMP = 0
      DO 695 JD = 1,JOBS
      MINT = 9999
C
C     THIS PART COMPUTES THE MINIMUM PROCESSING TIME AND
C     CORRESPONDING JOB ON MACHINE M
C
      DO 700 JA = 1,JOBS
      IF( IJUMP.EQ.0)                     GO TO 685
      DO 680 JR = 1,JOBS
      IF(JA.EQ.JOS(M,JR))                 GO TO 700
  680 CONTINUE
  685 IF(IT(JA,M).LT.MINT)                GO TO 705
      GO TO 700
  705 MINT = IT(JA,M)
      JMTX = JA
  700 CONTINUE
      MIT = 9999
C
C     THIS PART COMPUTES THE MINIMUM PROCESSING TIME AND THE
C     CORRESPONDING JOB ON MACHINE M+1
C
      DO 710 JB = 1,JOBS
      IF( IJUMP.EQ.0)                     GO TO675
      DO 670 JQ = 1,JOBS
      IF(JB.EQ.JOS(M,JQ))                 GO TO 710
  670 CONTINUE
  675 IF(IT(JB,M+1).LT.MIT)               GO TO 715
      GO TO 710
  715 MIT = IT(JB,M+1)
      JMTY = JB
  710 CONTINUE
      IF(MINT.LE.MIT)                     GO TO 720
      JX = JX - 1
      JOS(M,JX) = JMTY
      IJUMP = 1
      IF(JX.EQ.1)                         GO TO 721
      GO TO 695
  720 JOS(M,JJ) = JMTX
      IJUMP = 1
  721 JJ = JJ + 1
  695 CONTINUE
  690 CONTINUE
      NREPET = NREPET + 1
  692 CONTINUE
C
C     PRINT THE SEQUENCE IF IPRINT = 1
C
      IF(IPRINT.EQ.0)                 GO TO 101
      DO 102 M = 1,MM
      MR = M + 1
  102 WRITE(3,3000) M,MR,(JOS(M,JN),JN = 1,JOBS)
C
C     SPLITTING UP THE JOBS INCLUDED IN THE PARTIAL SEQUENCE
C
  101 DO 731 K = 1,LL
      J(K) = JSQ(LL,N,K)
  731 CONTINUE
C
C     SPLITTING UP THE REMAINING JOBS FROM THE SEQUENCE
C     COMPUTED BY THE JOHNSON'S CRITERION.
C
      JU = LL + 1
      DO 733 M = 1,MM
      KN = 1
      DO 732 K = JU,JOBS
  736 JUSK = JOS(M,KN)
      DO 734 I = 1,LL
      IF(JUSK.EQ.J(I))                GO TO 735
  734 CONTINUE
      JPS(M,K) = JUSK
      KN = KN + 1
      IF( KN.GT.JOBS)                 KN = JOBS
      GO TO 732
  735 KN = KN + 1
      IF( KN.GT.JOBS)                 KN = JOBS
      GO TO 736
  732 CONTINUE
  733 CONTINUE
C
C     FORMING THE COMPLETION TIME MATRIX
C
      DO 750 M = 1,MM
      MF = M + 1
      DO 737 K = JU,JOBS
  737 J(K) = JPS(M,K)
      IF(MF.GT.2)                     GO TO 742
      DO 738 I = 1,JOBS
      JJJ = J(I)
      IF(I.GT.1)                      GO TO 739
      DO 740 MA = M,MF
      IF(MA.GT.1)                     GO TO 741
      ICT(JJJ,MA) = IT(JJJ,MA)
      ICTF = ICT(JJJ,MA)
      GO TO 740
  741 ICT(JJJ,MA) = ICTF + IT(JJJ,MA)
  740 CONTINUE
```

```
      GO TO 738
  739 II = I-1
      JP = J(II)
      DO 743 MA = M,MF
      IF(MA.GT.1)                     GO TO 745
      ICT(JJJ,MA) = ICT(JP,MA) + IT(JJJ,MA)
      GO TO 743
  745 MN = MA - 1
      IF(ICT(JJJ,MN).GE.ICT(JP,MA))    GO TO 746
      ICT(JJJ,MA) = ICT(JP,MA) + IT(JJJ,MA)
      GO TO 743
  746 ICT(JJJ,MA) = ICT(JJJ,MN) + IT(JJJ,MA)
  743 CONTINUE
  738 CONTINUE
      GO TO 104
  758 CONTINUE
      GO TO 602
  742 DO 747 IR = 1,LL
      JJJ = J(IR)
      IF(IR.GT.1)                     GO TO 748
      ICT(JJJ,MF) = ICT(JJJ,M) + IT(JJJ,MF)
      GO TO 747
  748 II = IR - 1
      JJP = J(II)
      IF(ICT(JJP,MF).GE.ICT(JJJ,M))    GO TO 751
      ICT(JJJ,MF) = ICT(JJJ,M) + IT(JJJ,MF)
      GO TO 747
  751 ICT(JJJ,MF) = ICT(JJP,MF) + IT(JJJ,MF)
  747 CONTINUE
      DO 752 K = JU,JOBS
      JJ = J(K)
      JR = J(K-1)
  752 ICT(JJ,M) = ICT(JR,M) + IT(JJ,M)
      DO 753 K = JU,JOBS
      KR = J(K)
      JN = J(K-1)
      IF(ICT(KR,M).GE.ICT(JN,MF))      GO TO 754
      ICT(KR,MF) = ICT(JN,MF) + IT(KR,MF)
      GO TO 753
  754 ICT(KR,MF) = ICT(KR,M) + IT(KR,MF)
  753 CONTINUE
C
C     PRINT THE COMPLETION TIME MATRIX IF IPRINT = 1
C
  104 IF(IPRINT.EQ.0)                 GO TO 759
      WRITE (3,3010)
      WRITE(3,3020) M,MF
      DO 601 JM = 1,JOBS
      JK = J(JM)
  601 WRITE(3,3030) JK,ICT(JK,M),ICT(JK,MF)
C
C     FINDING THE MINIMUM OF THE SUMS OF PROCESSING TIMES
C     FOR JOBS NOT INCLUDED IN THE PARTIAL SEQUENCE ON
C     SUCCEEDING MACHINES.
C
  759 MH = MF + 1
      IF(M.NE.MM)                     GO TO 523
      JJJ = J(JOBS)
      LB = ICT(JJJ,MF)
      IF(LB - ILB(LL,N))521,521,522
  522 ILB(LL,N) = LB
  521 GO TO 602
  523 DO 760 KA = JU,JOBS
      ITXM = 0
      DO 761 MG = MH,MACH
      KJ = J(KA)
  761 ITXM = ITXM + IT(KJ,MG)
      IF(KA.GT.JU)                            GO TO 762
      ITXMIN = ITXM
  762 IF(ITXM - ITXMIN)763,760,760
  763 ITXMIN = ITXM
  760 CONTINUE
      ITXMM(M) = ITXMIN
      IF(IPRINT.EQ.0)                         GO TO 524
      WRITE(3,3040) M,MF,ITXMM(M)
C
C     COMPUTATION OF LOWER BOUND
C
  524 JJJ = J(JOBS)
      LB = ICT(JJJ,MF) + ITXMM(M)
      ILB(LL,N) = LB
      IF(M.EQ.1)                              GO TO 758
  757 IF(LB - ILB(LL,N))602,602,765
  765 ILB(LL,N) = LB
  602 IF(IPRINT.EQ.0)                         GO TO 750
      WRITE(3,3050) M,MF,LB
  750 CONTINUE
      IF(IPRINT.EQ.0)                         GO TO 603
      WRITE(3,3060) LL,N,ILB(LL,N)
  603 RETURN
      END
      SUBROUTINE BOUND3 (NN,LL,JOBS,N)
      DIMENSION ICT(20,20),ITRR(10),ITRMM(10),IDT(20,20)
      COMMON IT(20,10),MACH,ISTMIN,KOUNT,IPRINT,IB
      COMMON J(20),ILB(20,20),KONTMB,KONTJB,KONTMJ
      COMMON NREPET,JSQ(20,20,20)
 3200 FORMAT(1H ,10X,'JOB',10X,'M1      M2      M3      M4      M5',
     1'      M6      M7      M8      M9      M10 ')
 3210 FORMAT(1H ,10X,I2,6X,10I7)
 3220 FORMAT(1H ,'      *****  COMPLETION  TIME  MATRIX  *****')
 3230 FORMAT(1H ,' SOPHISTICATED COMPLETION TIME FOR THE LAST JOB IN THE
     1 PARTIAL SEQUENCE.')
 3240 FORMAT(1H ,10X,'SECOND TERM',10X,'MACHINE')
 3250 FORMAT(1H ,13X,I4,14X,I4)
 3260 FORMAT(1H ,11X,'THIRD TERM',10X,'MACHINE')
 3270 FORMAT(1H ,10X,'ON MACHINE ',I2,'  LOWER BOUND IS  ',I4)
 3280 FORMAT(1H ,11HLOWER BOUND,15X,5HLEVEL,15X,4HNODE)
 3290 FORMAT(1H ,4X,I5,19X,I2,18X,I2)
C
C     SPLITTING THE JOBS INCLUDED IN THE PARTIAL SEQUENCE
C
      DO 120 K = 1,LL
      J(K) = JSQ(LL,N,K)
  120 CONTINUE
C
C     SPLITTING THE JOBS WHICH ARE NOT INCLUDED IN THE
C     PARTIAL SEQUENCE
C
      JU = LL+1
      DO 160 K = JU,JOBS
      KK = K-1
      DO 150 JUSK = 1,JOBS
      DO 130 I = 1,KK
      IF(JUSK.EQ.J(I))                    GO TO 150
  130 CONTINUE
      J(K) = JUSK
      GO TO 160
  150 CONTINUE
  160 CONTINUE
C
C     FORMING THE COMPLETION TIME MATRIX FOR JOBS INCLUDED IN
C     THE PARTIAL SEQUENCE
C
      DO 400 I=1,LL
      JJ = J(I)
      IF(I.GT.1)                          GO TO 430
      DO 410  M = 1,MACH
      IF(M.GT.1)                          GO TO 420
      ICT(JJ,M) = IT(JJ,M)
      ICTF = ICT(JJ,M)
      GO TO 410
  420 ICT(JJ,M) = ICTF + IT(JJ,M)
      ICTF = ICT(JJ,M)
  410 CONTINUE
      GO TO 400
  430 II = I-1
      JP = J(II)
      DO 500 M = 1,MACH
      IF(M.GT.1)                          GO TO 510
      ICT(JJ,M) = ICT(JP,M)+IT(JJ,M)
      GO TO 500
  510 MM = M-1
      IF(ICT(JJ,MM).GE.ICT(JP,M))     GO TO 520
      ICT(JJ,M) = ICT(JP,M)+IT(JJ,M)
      GO TO 500
  520 ICT(JJ,M) = ICT(JJ,MM)+IT(JJ,M)
  500 CONTINUE
  400 CONTINUE
      IF(IPRINT.EQ.0)                 GO TO 105
      WRITE(3,3220)
      WRITE(3,3200)
      DO 106 K = 1,LL
      JK = J(K)
  106 WRITE(3,3210) JK,(ICT(JK,M),M = 1,MACH)
  105 CONTINUE
C
C     FINDING THE SOPHISTICATED COMPLETION TIME
C
      DO 855 M = 1,MACH
      JL = J(LL)
      IF(M.GT.1)                      GO TO 860
      IDT(JL,M) = ICT(JL,M)
      GO TO 855
  860 MG = M-1
      IDT(JL,M) = ICT(JL,M)
      DO 865 MS = 1,MG
      MINT = 9999
      DO 875 K = JU,JOBS
      JK = J(K)
      ITIME = 0
      DO 870 MX = MS,MG
  870 ITIME = ITIME + IT(JK,MX)
      IF(ITIME.GE.MINT)               GO TO 875
      MINT = ITIME
  875 CONTINUE
      IDTX = ICT(JL,MS) + MINT
      IF(IDT(JL,M).LT.IDTX)           GO TO 880
      GO TO 865
  880 IDT(JL,M) = IDTX
  865 CONTINUE
  855 CONTINUE
C
C     PRINT THE SOPHISTICATED COMPLETION TIME FOR THE LAST
C     JOB IN THE PARTIAL SEQUENCE.
C
      IF(IPRINT.EQ.0)                 GO TO 107
      WRITE(3,3230)
      WRITE(3,3200)
      WRITE(3,3210) JL,(IDT(JL,M),M = 1,MACH)
  107 CONTINUE
C
C     SECOND TERM IN THE IGNALL & SCHRAGE LOWER BOUND
C
      KU = LL + 1
      DO 185  M = 1,MACH
      ITR = 0
      DO 180  I = KU,JOBS
      K = J(I)
  180 ITR = ITR + IT(K,M)
      ITRR(M) = ITR
  185 CONTINUE
```

```
C
C     PRINT THE SECOND TERM FOR EACH MACHINE, IF IPRINT = 1.
C
      IF(IPRINT.EQ.0)                           GO TO 108
      WRITE(3,3240)
      DO 109 M = 1,MACH
  109 WRITE(3,3250) ITRR(M),M
  108 CONTINUE
C
C     THIRD TERM IN THE IGNALL & SCHRAGE LOWER BOUND
C
      MMM = MACH - 1
      DO 250  M = 1,MMM
      MA = M+1
      DO 200  I = KU,JOBS
      ITRM = 0
      DO 190  MR = MA,MACH
      K = J(I)
  190 ITRM = ITRM + IT(K,MR)
      IF(I.GT.KU)                               GO TO 210
      ITRMIN = ITRM
  210 IF(ITRM-ITRMIN)220,200,200
  220 ITRMIN = ITRM
  200 CONTINUE
      ITRMM(M) = ITRMIN
  250 CONTINUE
C
C     PRINT THE THIRD TERM FOR ALL MACHINES , EXCEPT THE LAST.
C
      IF(IPRINT.EQ.0)                           GO TO 110
      WRITE(3,3260)
      DO 111 M = 1,MMM
  111 WRITE(3,3250) ITRMM(M),M
  110 CONTINUE
C
C     COMPUTATION OF SOPHISTICATED LOWER BOUND
C
      DO 600  M = 1,MACH
      JL = J(LL)
      IF(M.EQ.MACH)                             GO TO 610
      LB = IDT(JL,M) + ITRR(M) + ITRMM(M)
      IF(M.GT.1)                                GO TO 620
      ILB(LL,N) = LB
      GO TO 112
  610 LB = IDT(JL,M) + ITRR(M)
  620 IF(LB-ILB(LL,N))112,112,630
  630 ILB(LL,N) = LB
C
C     PRINT THE LOWER BOUNDS FOR EACH MACHINE AT LEVEL LL.
C
  112 IF(IPRINT.EQ.0)                           GO TO 600
      WRITE(3,3270) M,LB
  600 CONTINUE
C
C     PRINT THE LOWER BOUND AT LEVEL  LL .
C
      IF(IPRINT.EQ.0)                           GO TO 113
      WRITE(3,3280)
      WRITE(3,3290) ILB(LL,N),LL,N
  113 CONTINUE
```

```
      RETURN
      END
      SUBROUTINE BOUND4 (NN,LL,JOBS,N)
      DIMENSION ICT(20,20)
      COMMON IT(20,10),MACH,ISTMIN,KOUNT,IPRINT,IB
      COMMON J(20),ILB(20,20),KONTMB,KONTJB,KONTMJ
      COMMON NREPET,JSQ(20,20,20)
 3500 FORMAT(1H ,10X,'LOWER BOUND ON MACHINE ',I2,' IS ',I4)
 3510 FORMAT(1H ,10X,'LOWER BOUND AT LEVEL ',I2,' FOR NODE ',I2,' IS ',I
     15)
 3520 FORMAT(1H ,'     *****  COMPLETION  TIME  MATRIX  *****')
 3530 FORMAT(1H ,10X,'JOB',10X,'M1      M2      M3      M4      M5',
    1'      M6      M7      M8      M9      M10 ')
 3540 FORMAT(1H ,10X,I2,6X,10I7)
C
C     SPLITTING THE JOBS INCLUDED IN THE PARTIAL SEQUENCE
C
      IF(IB.EQ.5)                         GO TO 155
      DO 120 K = 1,LL
      J(K) = JSQ(LL,N,K)
  120 CONTINUE
C
C     SPLITTING THE JOBS WHICH ARE NOT INCLUDED IN THE
C     PARTIAL SEQUENCE
C
      JU = LL+1
      DO 160 K = JU,JOBS
      KK = K-1
      DO 150 JUSK = 1,JOBS
      DO 130 I = 1,KK
      IF(JUSK.EQ.J(I))                    GO TO 150
  130 CONTINUE
      J(K) = JUSK
      GO TO 160
  150 CONTINUE
  160 CONTINUE
  155 CONTINUE
C
C     FORMING THE COMPLETION TIME MATRIX FOR JOBS INCLUDED IN
C     THE PARTIAL SEQUENCE
C
      DO 400 I=1,LL
      JJ = J(I)
      IF(I.GT.1)                          GO TO 430
      DO 410  M = 1,MACH
      IF(M.GT.1)                          GO TO 420
      ICT(JJ,M) = IT(JJ,M)
      ICTF = ICT(JJ,M)
      GO TO 410
  420 ICT(JJ,M) = ICTF + IT(JJ,M)
      ICTF = ICT(JJ,M)
  410 CONTINUE
      GO TO 400
  430 II = I-1
      JP = J(II)
      DO 500 M = 1,MACH
      IF(M.GT.1)                          GO TO 510
      ICT(JJ,M) = ICT(JP,M)+IT(JJ,M)
      GO TO 500
  510 MM = M-1
      IF(ICT(JJ,MM).GE.ICT(JP,M))         GO TO 520
      ICT(JJ,M) = ICT(JP,M)+IT(JJ,M)
      GO TO 500
  520 ICT(JJ,M) = ICT(JJ,MM)+IT(JJ,M)
  500 CONTINUE
  400 CONTINUE
C
C     PRINT THE COMPLETION TIME MATRIX.
C
      IF(IPRINT.EQ.0)                   GO TO 205
      WRITE(3,3520)
      WRITE(3,3530)
      DO 206  K = 1,LL
      JK = J(K)
  206 WRITE(3,3540) JK,(ICT(JK,M),M = 1,MACH)
  205 CONTINUE
C
C     SECOND TERM IN THE JOB BASED LOWER BOUND.
C
      JP = LL + 1
      DO 903 MJ = 1,MACH
      MAX = 0
      DO 900 K = JP,JOBS
C
C     PROCESSING TIME ON MACHINE 'MJ' AND ALL MACHINES
C     FOLLOWING MACHINE 'MJ' , FOR A UNSCHEDULED JOB .
C
      IF(MJ.NE.MACH)                    GO TO 905
      DO 906 KQ = JP,JOBS
      JQ = J(KQ)
  906 MAX = MAX + IT(JQ,MACH)
      GO TO 907
  905 JH = J(K)
      IPROC = 0
      DO 901 M = MJ,MACH
  901 IPROC = IPROC + IT(JH,M)
C
C     MINIMUM RUNNING TIME FOR THE REMAINING JOBSEXCLUDING
C     THE JOB CONSIDERED FOR NEXT SEQUENCE POSITION
C
      IRUN = 0
      DO 902 KS = JP,JOBS
      JR = J(KS)
      IF(JR.EQ.JH)                      GO TO 902
      IF(IT(JR,MJ).LE.IT(JR,MACH))      GO TO 904
      IRUN = IRUN + IT(JR,MACH)
      GO TO 902
  904 IRUN = IRUN + IT(JR,MJ)
  902 CONTINUE
      LTERM = IPROC + IRUN
      IF(LTERM.GT.MAX)                  MAX = LTERM
  900 CONTINUE
C
C     LOWER BOUND ON MACHINE 'MJ' .
C
  907 JL = J(LL)
      LB = ICT(JL,MJ) + MAX
      IF(IPRINT.EQ.0 )                  GO TO 908
      WRITE(3,3500)MJ,LB
  908 IF(MJ.GT.1)                       GO TO 909
      ILB(LL,N) = LB
      GO TO 903
  909 IF(LB - ILB(LL,N))903,903,910
  910 ILB(LL,N) = LB
  903 CONTINUE
      IF(IPRINT.EQ.0)                    GO TO 911
      WRITE(3,3510) LL,N,ILB(LL,N)
  911 RETURN
      END
      SUBROUTINE BOUND5(NN,LL,JOBS,N)
      COMMON IT(20,10),MACH,ISTMIN,KOUNT,IPRINT,IB
      COMMON J(20),ILB(20,20),KONTMB,KONTJB,KONTMJ
      COMMON NREPET,JSQ(20,20,20)
C
C     SPLITTING THE JOBS INCLUDED IN THE PARTIAL SEQUENCE
C
      DO 120 K = 1,LL
      J(K) = JSQ(LL,N,K)
  120 CONTINUE
C
C     SPLITTING THE JOBS WHICH ARE NOT INCLUDED IN THE
C     PARTIAL SEQUENCE
C
      JU = LL+1
      DO 160 K = JU,JOBS
      KK = K-1
      DO 150 JUSK = 1,JOBS
      DO 130 I = 1,KK
      IF(JUSK.EQ.J(I))                      GO TO 150
  130 CONTINUE
      J(K) = JUSK
      GO TO 160
  150 CONTINUE
  160 CONTINUE
      CALL BOUND1(NN,LL,JOBS,N)
      MBASED = ILB(LL,N)
      CALL BOUND4(NN,LL,JOBS,N)
      IF(MBASED.EQ.ILB(LL,N))               GO TO 983
      IF(MBASED.GT.ILB(LL,N))               GO TO 980
      KONTJB = KONTJB + 1
      GO TO 981
  980 ILB(LL,N) = MBASED
      KONTMB = KONTMB + 1
      GO TO 981
  983 KONTMJ = KONTMJ + 1
  981 RETURN
      END
      FUNCTION BANDNO(IY)
      IY = IY*65627
      IF(IY) 5,6,6
    5 IY = IY + 2147483647 + 1
    6 BANDNO = IY*.4656613E-9
      RETURN
      END
```

REFERENCES

1. Agin, N., "Optimum Seeking with Branch and Bound," Management Science, Vol. 13, No. 4, 1966, pp 176 - 185.

2. Ashour, S., "Comparison of Different Approaches for Solving Sequencing Problems," submitted to Operations Research for publication, 1969.

3. Brown, A.P.G., and Z.A. Lomnicki, "Some Applications of the Branch-and-Bound Algorithm to the Machine Scheduling Problem," Operational Research Quarterly, Vol. 17, No. 2, 1966, pp 173 - 186.

4. Churchman, C.W., R.L. Ackoff, and E.L. Arnoff, Introduction to Operations Research, John Wiley & Sons, New York, 1961, pp 450 - 476.

5. Conway, R.W., W.L. Maxwell, and L.W. Miller, Theory of Scheduling, Addison-Wesley Publishing Company, Reading, Massachusetts, 1967.

5a. Eastman, W.L., S. Ever, and I.M. Isaacs, "Bounds for Optimal Scheduling of Jobs," Management Science, Vol. 11, No. 2, 1964, pp. 268 - 279.

6. Elmaghraby, S.E., "The Machine Sequencing Problem - Review and Extension," Naval Research Logistics Quarterly, Vol. 15, No. 2, 1968, pp 205 - 232.

7. Giglio, R.J., and H.M. Wagner, "Approximate Solutions to the Three-Machine Scheduling Problem," Operations Research, Vol. 12, No. 2, 1964, pp 305 - 324.

8. Ignall, E.J., and L.E. Schrage, "Application of the Branch and Bound Technique to Some Flow-Shop Scheduling Problems," Operations Research, Vol. 13, No. 3, 1965, pp. 400 - 412.

9. Johnson, S.M., "Optimal Two and Three Stage Production Schedules with Setup Times Included," Naval Research Logistics Quarterly, Vol. 1, No. 1, 1954, pp. 61 - 68.

9a. Land, A.H., and A. Doig, "An Automatic Method of Solving Discrete Programming Problems," Econometrica, Vol. 28, No. 3, 1960, pp. 497 - 520.

10. Lawler, E.L., and D.E. Wood, "Branch-and-Bound Methods: A Survey," . Operations Research, Vol. 14, No. 4, 1966, pp. 699 - 719.

11. Little, J.D.C., K.G. Murty, D.W. Sweeney, and C. Karel, "An Algorithm for the Travelling Salesman Problem," Operations Research, Vol. 11, No. 6, 1963, pp. 972 - 989.

12. Lomnicki, Z.A., "A Branch-and-Bound Algorithm for the Exact Solution of the Three Machine Scheduling Problem," Operational Research Quarterly, Vol. 16, No. 1, 1965, pp. 89 - 100.

13. McMahon, G.B., and P.G. Burton, "Flowshop Scheduling with the Branch-and-Bound Method," Operations Research, Vol. 15, No. 3, 1967, pp. 473 - 481.

14. Muth, J.F., and G.L. Thompson, eds., Industrial Scheduling, Prentice-Hall, Englewood Cliffs, New Jersey, 1963.

15. Nabeshima, I., "On the Bound of Makespans and its Application in M Machine Scheduling Problem," Journal of the Operations Research Society of Japan, Vol. 9, Nos. 3&4, 1967, pp. 98 - 135.

16. Nugent, C.E., "On Sampling Approaches to the Solution of the n - By - m Static Sequencing Problem," Ph.D. Thesis, Cornell University, Ithaca, New York, 1964.

17. Ryser, H.J., Combinatorial Mathematics, The carus Mathematical Monographs, No. 14, The Mathematical Association of America, 1963, Dist. John Wiley and Sons, Inc.

18. Sisson, R., "Sequencing Theory," Chapter 7, Progress in Operations Research, Vol. 1, Ackoff, R.L., ed., John Wiley, New York, 1961.

19. Spinner, A.H., "Sequencing Theory - Development to Date," Naval Research Logistics Quarterly, Vol. 15, No. 2, 1968, pp. 319 - 324.

ANALYSIS AND COMPARISON OF VARIOUS LOWER-BOUNDS ON SCHEDULE TIMES FOR THE SOLUTION OF FLOW-SHOP PROBLEMS

by

MOHAMMED NAWA-ULLAH QURAISHI

B.E. (Mech.), Maulana Azad College of Technology

Bhopal, (M.P.), India, 1966

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1969

There exist several production situations in which a certain number of jobs have to be processed on various machines according to a specified technological requirement. The production scheduling problem consists of finding the sequence of jobs to be processed on all machines so as to optimize a certain criterion. This research is concerned with the solution to the flow shop problem in which all jobs have the same machine ordering. Various approaches to this problem are available; however, due to the combinatorial nature of the problem, these techniques are inefficient even for relatively small size problems.

The branch-and-bound technique is used in this research. This procedure involves generation of nodes that indicate partial sequences. A lower-bound on the schedule time is evaluated for each node in order to determine which node is to be explored. The number of explored nodes can be curtailed by a powerful bounding process which is imbedded in the branch-and-bound algorithm. Back-tracking process of the algorithm guarantees optimality of the solution.

The basic objective of this research is to analyze mathematically and empirically the five bounding procedures developed by several investigators. The basic concept of the branch-and-bound technique including branching, bounding and back-tracking processes are discussed. In order to study the merits of the various bounding procedures, several computational experiments were conducted. The comparison was based on both the number of nodes explored and the computation time spent in obtaining the optimal solution. Based on problems solved, the number of nodes explored and the computation time increase rapidly as the number of jobs increases. The ranks of the bounding procedures are tabulated according to the number of nodes explored and the computation time spent. It appears that LB I

ranks first in the computation time; however, its rank in the number of nodes is second.