A BRANCH-AND-BOUND ALGORITHM
FOR JOB-SHOP PROBLEMS

by 1264

SANGAYYA RACHAYYA HIREMATH

B.E. (Mech.), Karnatak University

Dharwar, Mysore-State, India, 1967

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1970

Approved by:

Said Ashour

Major Professor

ACKNOWLEDGEMENT

I am greatly indebted to my Major Professor, Dr. S. Ashour, for his valuable guidance and the personal interest taken in the preperation of this Master's Thesis. I am also thankful to Dr. F. A. Tillman, Head, Department of Industrial Engineering, Dr. L. E. Grosh, Department of Industrial Engineering and R. O. Turnquist, Department of Mechanical Engineering for their kind patronage.

I appreciate and thank Sudesh Kumar for his assistance in the initial stage of programming. I also acknowledge and thank S. V. Gadad, Pathare and other friends for their help in proof reading, preparing drawings etc. Lastly, I thank Mrs. Jirak for her assistance in typing.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

INTRODUCTION

The scheduling problem arises whenever J jobs have to be processed on M machines in a specified technological requirement. The problem consists of finding the sequence of J jobs on M machines so that a certain criterion is optimized. The formulation of a scheduling problem usually takes the form of a mathematical model whose constituents are: (1) criteria, (2) parameters and variables; and (3) assumptions.

In general there are two types of criteria as stated in [20]. The first type includes those which do not distinguish among individual jobs. This indicates that such a criterion is a function of the sequence of jobs, taken as a whole. Examples of this type are the minimization of the schedule time, i.e., the minimization of the total processing time of all jobs on all machines, the maximization of overall profit. The second type includes those which distinguish among the individual jobs. This indicates that such a criterion is a function of the individual jobs in the sequence. An example of this type is the minimization of the total tardiness of jobs. In this case, the tardiness of an individual job is considered. Tardiness of a job is the positive difference between the completion time of the job and its due-date. The criterion considered in this thesis is the minimization of schedule time.

The formulation of a scheduling model depends, among others, on the behavior of job-arrivals. A deterministic model is applied to a situation in which several jobs arrive simultaneously in a shop that is idle and immediately available for work. However, a stochastic model is applied to a situation in which several jobs arrive continuously at random intervals.

The scheduling models, in practice, are usually of stochastic nature. The stochastic models, therefore, are of practical value. However, the deterministic models have an inherent interest of their own, because they can be considered as a prelude to the stochastic models due to the following reasons as stated in [30]: (1) The deterministic models provide an approach to handle the more complex stochastic models; and (2) The knowledge gained from work on the deterministic models may be directly applicable to the stochastic models. The study of the deterministic models is also interesting as an example of combinatorial problems and the solution techniques may be applicable to other combinatorial problems such as line balancing and travelling salesman problems. It is therefore worthwhile to study the deterministic models.

Most research workers have investigated simple models by imposing several assumptions. Among the assumptions imposed are: (1) Each operation once started must be performed to completion, (2) Each machine can process only one job at a time, (3) There is only one machine of each type; and (4) Processing times include set up and transportation times between machines, if any.

In searching for the optimal solution, one should enumerate and evaluate the possible sequences. However, the number of possible sequences increases very rapidly with the increase in the number of jobs or machines because of the combinatorial nature of the scheduling problem. For a problem of two jobs to be processed on three machines, the number of possible sequences is $(J!)^M$ or $(2!)^3 = 8$. Whereas, for a (6x3) problem, the number of possible sequences is $(6!)^3$ or 373,328,000. Thus it is evident that the complete enumeration method is highly impractical except for trivially small problems. Consequently, other approaches such as combinatorial analysis, mathematical programming, and simulation are used to solve the scheduling problem.

1.1 Problem Formulation*

There are two types of shop, depending on the order in which various machines perform a particular job. They are referred to as flow-shop and job-shop. In flow-shop problems, each job is performed on a certain set of machines in an identical order. Whereas, in job-shop problems, the machine-ordering for each job may be different. This research is concerned with job-shop problems.

In formulating a scheduling problem, a job is designated by an integer j and a machine by an integer m. An operation of job j on machine m is represented by a node (jm).

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 the 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 by (jm_k). Second, the operation of a job j on a machine in the ℓ^{th} order-position is designated by (jm_ℓ). Finally, a specific operation involving a particular job j_k and a

*Adapted from Ashour, S., Introduction to Scheduling, John Wiley & Sons, Inc., in Press particular machine m_ℓ is denoted by $(j_k m_\ell)$.

The machine-ordering for a particular job j is designated by a row vector such that

$$\mathcal{M}j = [jm_1 \; jm_2 \; . \; . \; . \; jm_\ell \; . \; . \; . \; jm_M],$$

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

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

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

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 this machine ordering matrix, operation or node (11) proceeds operations (13) and (12), and operation (11) directly proceeds operation (13).

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. For convenience, the processing times for job j on all machines are designated

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

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

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 . The processing time matrix of the above example is shown below

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

The above processing time matrix indicates that to perform job 1 on machines m_1, m_2 and m_3, it requires 5, 4, 1 units of time respectively. Similarly, job 2 requires 2, 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.

Sometimes it is necessary to determine the completion time of an operation. The completion time of an operation is the sum of the processing times and idle times, if any, of all the preceding operations and those of the operation considered. Similar to the machine-ordering and processing time matrices, the initial completion time matrix of the above example can be shown as below

$$C(jm) = \begin{pmatrix} c_{1m_1} & c_{1m_2} & c_{1m_3} \\ c_{2m_1} & c_{2m_2} & c_{2m_3} \end{pmatrix} = \begin{pmatrix} 2 & 6 & 7 \\ 5 & 6 & 9 \end{pmatrix}$$

This completion time matrix is formed regardless of any conflict between the two jobs.

Whenever two or more jobs have their operations on the same machine during a common time interval, a conflict exists. This conflict can be shown using a Gantt chart, as shown in Figure 1.1. The operations in the conflict set on a certain machine are shown by horizontal hatching. For example, it is obvious from Figure 1.1 that jobs 1 and 2 have conflict on machine 1 during the time interval between 2 and 3. Whenever there is a conflict on a certain machine, it can be resolved in favor of one of the jobs in the conflict set on that machine. The Gantt chart shown in Figure 1.2 shows the resolving of the conflict in favor of node (11).

1.2 Literature Review

Various basic approaches have concentrated on selecting smaller and smaller subsets of schedules from the larger set of possible schedules. One of the approaches is that the set of feasible schedules is obtained by selecting each time an operation at random from the set of schedulable operations. The operations which are available for scheduling immediately without contradicting any precedence relationship are called the schedulable operations. The feasible schedule obtained by selecting the scheduable operations at random does not guarantee optimality. Therefore, in another approach, a certain procedure called left-shift may be used to obtain a better set of schedules, known as active schedules. A left-shift operation consists of jumping to the left of an operation over another operation if there is sufficient idle time to accommodate the processing time of the operation to be shifted. An active schedule is the one in which left-shift is not possible. Clearly, the set of active schedules

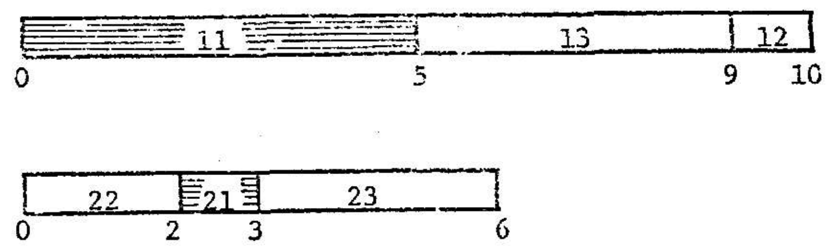

Figure 1.1

A Gantt Chart Depicting the Conflict Between Nodes (11) and (21)

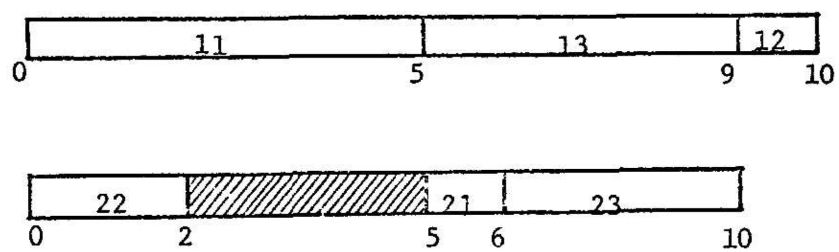

Figure 1.2

A Gantt Chart Depicting the Resolving of Conflict in Favor of Node (11) is smaller than the complete set of feasible schedules. Such a set of active schedules contains the optimal sequence(s). In another approach, a subset of schedules, known as non-delay schedules, is generated from the set of active schedules. Delay is defined as machine idle time incurred while a job is available for processing. It should be pointed out that the set of non-delay schedules may not always contain the optimal solution. It is obvious that in all the above approaches an attempt is made to obtain smaller and smaller subsets of better schedules from a larger set of possible schedules. Such an approach will be referred to as microsimulation approach.

Several investigators have worked on the above approaches. Heller [28] has originally developed an algorithm, based on the linear graph theory, for the construction and evaluation of feasible sequences. In the linear graph theory, an operation of job j on machine m for ith return is represented by node (mji). The algorithm selects one of the schedulable operations at random. When one of the schedulable operations is selected, a new set of schedulable operations is again formed. Since the random procedure in selecting the schedulable operations, one at a time, does not guarantee a good schedule, Heller and Logemann [29] have incorporated in their algorithm the feature that selects one of the schedulable operations using the first-come, first-served rule. If a tie is encountered, it is broken randomly. Ashour [15] has modified the algorithm to suit the assumption that no job is processed more than once by any machine and also for the construction of feasible sequences for job-shop problems. To summarize, this approach produces feasible schedules, sometimes referred to as semi-active schedules.

Giffler and Thompson [23] have developed an algorithm to obtain the set of active schedules. As mentioned earlier, the set of active schedules is relatively smaller than the complete set of feasible schedules and contains the optimal solution. They have obtained the complete set of active schedules by resolving the conflicts in all possible ways. However, the complete set of active schedules also becomes too large even for problems of small size. As an example, for a (6x5) problem, Giffler, Thompson and Van Ness [24] have observed that the complete set of active schedules obtained by using the nonnumerical program consists of 84802 schedules, while the complete set of feasible schedules is $(6!)^5$ or about 8 million. In a nonnumerical program all operation times are assumed to be unity. It is, therefore, obvious that there are about 100 feasible sequences corresponding to each active schedule. The size of the problems solved using this program varies between one and five machines, the number of jobs being fixed at 6. Since they have found that there exists an enormous number of active schedules even for trivially small problems, they have concluded that it is necessary to sample from the set of active schedules. In this sampling procedure, they have resolved conflicts at random. The size of problems solved varies from 4 to 200 jobs and 1 to 25 machines. From these experiments, they have concluded that such a set of optimal solutions, even though a very small subset from the complete set of active schedules, increases very rapidly as the size of the problem increases. They have also computed the probability of obtaining an optimal schedule in a certain number of trials.

Fisher and Thompson [21] have reported their computational experience about the probabilistic learning combinations of local job-shop scheduling rules. The local scheduling rules are the ones that can be applied by machine operator and these require only the knowledge of the work waiting to be processed on his machine. They have selected two rules, the shortest imminent operation rule, SIO, selecting that job with the shortest operation time and the longest remaining time rule, LRT, selecting that job with the maximum remaining processing time. They have modified these two rules such that an operation is not scheduled if a job of higher priority presently being processed on another machine, will arrive prior to the expected completion of the highest priority operation in the queue. If such a situation occurs, the machine is held idle until the new operation arrives. However, an operation is not delayed because of the possible arrival of a higher priority operation now waiting in some other queue. The basic principle of using the learning processes is that the computational experimentation can produce learning of some systematic way in which the frequency of the use of the above two rules is varied. For example, for a particular problem, the computational experience may be such that the SIO rule should be used initially and the LRT rule should be used later. Their experience has showed that the combined rule invariably does much better than any of the local rules taken singly. The criterion used is the minimization of schedule time. They have applied four types of learning using modified SIO and LRT rules. One type of learning is characterized by an unbiased starting position at which the probability of selecting each decision rule is equal, whereas, in another type of learning, the probability of selection may be biased. The sizes of the problems solved vary from 6 to 20 jobs and 5 to 10 machines. From the computational experience, they have concluded that learning is possible and an unbiased combination of scheduling rules is better than any of them taken separately.

Nugent [30] has modified Heller's algorithm to generate the set of non-delay schedules which form a subset of active schedules. As defined earlier, delay is the machine idle time incurred while a job is available for processing. He has generated the non-delay schedules, using the probability dispatching rules such as the first-come, first-served rule, FCFS, the most work remaining rule, MWKR, the shortest operation rule, SHOPN, and the random rule, RANDOM. While using the FCFS, the ties, if any, are broken randomly. However, when other rules such as MWKR and SHOPN are used, the ties are broken using the FCFS. The basic principle in using the probability dispatching rules is that a probability is assigned to each job in a particular set. Such a set consists of jobs available to be processed on a certain machine at a time when the machine is available for processing. He has conducted experiments on two different kinds of sets of problems. One such kind includes 8 sets of jobs generated internally. The sizes of such problems vary from 20 to 100 jobs, with number of machines equal to 9. Another kind includes 7 sets of jobs obtained extenally. The sizes of such problems vary from 6 to 100 jobs and 3 to 10 machines. The purpose of conducting experiments on externally obtained sets of jobs is to compare these results obtained using the probability dispatching rules with those obtained by the previous researchers. The criteria used are minimization of the schedule time and minimization of the mean flow time. In general, he has observed that the non-delay schedules produced using the probability dispatching rules are generally better than those produced by previous methods.

Ashour [15] has developed decomposition approach for scheduling problems. The approach consists of decomposing the original problem into a number of smaller subgroups. This approach attempts to minimize the computational time. The computational experience shows that the mean and minimum of the schedule times obtained by decomposition method is smaller than that obtained by complete or partial enumeration. The mean and minimum schedule time increases as the number of jobs in each subgroup decreases. The size of the problems solved varies from 6 to 40 jobs and 3 to 10 machines.

The branch-and-bound approach generates an optimal solution after the generation of only a small subset of possible sequences. Land and Doig [8] have first developed the basic concepts of this approach. It has been named by Little et. al [10] while solving the travelling salesman problem. Ignall and Schrage [7] have used this approach to the two- and three-machine flow-shop problem using their lower bound. Brown and Lomnicki [4] have extended the branch-and-bound algorithm to any number of machines. McMahon and Burton [12] have developed a new lower bound, referred to as the composite-based bound and have applied the technique to three-machine problem. Ashour and Quraishi [2] have presented a mathematical analysis and comparative evaluation of various lower bounds for the solution of the flow-shop problem.

In dynamic situations, macrosimulation approach is used. Conway et al. [19] have used this approach for the stochastic models to compare the performance of several priority rules.

For the mathematical formulation of the scheduling problem, see Ashour [16].

1.2 Proposed Research

In this paper, a branch-and-bound algorithm for the job-shop problem will be developed. In addition, two new lower bounds, referred to as composite-based bounds LB I and LB II, will be developed and presented in a mathematical form and rigorous notation. For comparison purpose, a mathematical analysis in rigorous notation of some other existing lower bounds will also be presented. One of the existing lower bounds, referred to as bounding procedure LB IV, is modified by incorporating a new feature. The computation of lower bounds by each of the bounding procedures will be illustrated by a sample problem. Furthermore, a more general computational algorithm will be illustrated by the same sample problem, using the composite-based bound LB I.

Many experiments have been conducted using IBM 360/50 computer in order to obtain a fair comparison among the various bounding procedures. The solutions obtained by different bounding procedures are compared with reference to the following: (1) the number of nodes explored, (2) the computational time; and (3) the efficiency of the solution obtained without backtracking. Statistical analysis is carried out to compare the maximum, minimum, mean and standard deviation for the number of nodes explored and the efficiency of the solution obtained without backtracking.

CHAPTER II

DEVELOPMENT OF A BRANCH-AND-BOUND TECHNIQUE

The branch-and-bound technique is an enumerative approach which consists of a systematic generation of a smaller subset of optimal solutions from a larger set of feasible solutions. The basic principle of branch-and-bound technique, when applied to job-shop problems is that it obtains an optimal solution from a set of schedules, known as active schedules. As mentioned in chapter I, Giffler and Thompson [23] have originally developed an algorithm to generate the set of active schedules. Beenhakker [17] has given a mathematical analysis, related to this algorithm, in rigorous notations, especially in checking for a conflict. He has used the algorithm to generate schedules which are optimal with a certain probability with respect to several criteria such as the minimum schedule time, maximum production rate, and minimum total idle time of machines. Brooks and White [4] have modified this algorithm by imbedding a bounding procedure, as a criterion for resolving the conflicts. However, they have reported on their computational experience that this procedure is too long to adopt on medium size computers, even for problems of moderate size. They have compared the results obtained by using lower bound as the criterion for resolving the conflicts with those obtained by using shortest operation time rule and longest remaining time rule. The criteria used for optimality are minimizing lateness and minimizing total schedule time. The size of the problems solved varies between 7 and 10 jobs, and 10 and 18 machines. For minimizing total schedule time, the results obtained by using lower bound have been better than those obtained by using other rules such as shortest operation time and longest remaining time.

This chapter is devoted to the discussion of a branch-and-bound algorithm, in which one of the two lower bounds developed in this thesis, is imbedded. In addition, a mathematical analysis and modification of some other existing lower bounds are presented. The branch-and-bound algorithm is illustrated by a sample problem and is summarized in formal steps.

2.1 Basic Concepts

This branch-and-bound technique is developed on the basis of two principal concepts: (1) the use of a controlled enumeration technique for considering all potential solutions; and (2) the application of a bounding procedure for the identification of a subset containing the optimal solution. The search for the subset of optimal solutions is systematically carried out through branching and bounding processes which may be easily discussed by using a scheduling tree. (Figure 2.9)

The scheduling tree is initialized by a node (ALL) representing the set of all feasible solutions. This node is branched into nodes at the first conflict level, each node representing an operation of a job on a certain machine. As defined earlier, when two or more jobs have their operations on the same machine during a common time interval, a conflict exists. The set of such jobs at a level is called a conflict set. The conflict level index increases by one whenever a conflict is resolved in favor of one of the jobs comprising the conflict set at that level. In other words, one of the jobs at a conflict level is selected for further branching. Consequently, a set of nodes is generated at the next level. This process of generating a new set of nodes at a level from a node at the preceeding level is referred to as the branching process. This process guarantees an optimal solution by generating all nodes of the scheduling tree.

As discussed above, for job-shop problem, each level represents a conflict among a set of jobs on the same machine, and the total number of conflict levels represents the number of conflicts resolved to obtain a schedule time resulting from the corresponding branch. Thus, for a particular problem, the number of conflict levels for different schedule times may be different. In general, the larger the schedule time the higher is the number of conflicts resolved for obtaining that schedule. This is due to the fact that idle time is inserted as a conflict is resolved. Thus, the number of conflicts resolved for obtaining the shortest schedule time should be smaller than that for obtaining a longer one. However, the schedule time is also a function of idle time inserted as a result of resolving a conflict. Furthermore, the amount of the idle time is a function of the processing times. It is worthwhile to note that for job-shop problems, the number of conflict levels varies from one problem to another. However, for flow-shop problems, the total number of levels is equal to the number of jobs in the problem. Another outstanding difference is that for flow-shop problem, the number of new nodes at any level L, emanating from each node at the preceding level, L-1, is equal to (J-L+1) and thus, the number of these new nodes increases as one moves down the scheduling tree along a particular branch. Whereas for job-shop problems, the number of nodes at any level depends on the number of jobs in the conflict set at that level.

As mentioned earlier, whenever a conflict is encountered it can be resolved in favor of one of the jobs in the conflict set. If the conflict is resolved in all possible ways, the size of the scheduling tree increases rapidly. The bounding process therefore helps select a particular node at a level for further branching and thus makes it possible to achieve a reduction in the generation of nodes at each level. In this process, a lower bound on the schedule time is computed for each node at a certain level and the node with the least lower bound, referred to as an active node, is selected for further branching. All other nodes at this level are thus discarded. The lower bound for a node is the sum of the completion time of the scheduled operations and the total processing time of the unscheduled operations for a particular job or machine. It has the property that it does not exceed the schedule time of the associated complete sequence. Thus, the bounding procedure enables one to look for the possibility of recognizing the optimal solution by exploring the least number of nodes. However, in many cases it is necessary to explore more nodes for obtaining an optimal solution. The size of the scheduling tree does not become too large if the lower bounds computed are as high as possible. Therefore the efficiency of the branch-and-bound technique depends greatly on the quality of the bounding procedure.

At the end of the scheduling tree and for a particular branch, the schedule time is obtained by resolving the last conflict. This solution may be greater than the lower bounds for some of the unexplored nodes, and thus the solution obtained may not be optimal. In order to guarantee optimality, a backtracking process has to be embedded in the branch-and-bound technique. In this process, the scheduling tree is traced back along the same branch until an unexplored node with a lower bound less than the previous solution is found. In a similar manner branching and bounding processes are repeated until a better solution is obtained. The previous solution is, therefore, updated by this solution. However, some branches may be terminated at a level where all nodes have lower bounds equal to or greater than the previous solution. The optimal solution is reached when there is no unexplored node with lower bound less than the updated solution.

2.2 Bounding Procedures.

The basic purpose of using bounding procedures in the branch-and-bound technique is to reduce the number of nodes explored and thus to improve the efficiency of the technique by decreasing the computer time required to solve the scheduling problem. As defined earlier, the lower-bound on the schedule time for a node is defined as the sum of the completion time of the scheduled jobs and the total processing times of the unscheduled jobs. The more powerful a bounding procedure the closer are the lower-bounds produced to the schedule-time. Such a bounding process produces the lower-bounds considering the idle times due to both the scheduled and unscheduled operations. In general, the idle times among the scheduled operations can be considered. However, it is difficult to determine the idle time among the unscheduled operations since their sequence is not known.

This section is devoted to the discussion and analysis of two composite-based bounds LB I, and LB II, developed in this thesis. The composite-based bounds consider the maximum of both machine-based and job-based bounds. The difference in the computation of lower bounds is illustrated by a sample problem. The problem is of job-shop type with four jobs and three machines. The machine ordering matrix and the processing time matrix are shown below.

$$M = \begin{pmatrix} 12 & 13 & 11 \\ 21 & 23 & 22 \\ 33 & 31 & 32 \\ 41 & 42 & 43 \end{pmatrix} \qquad T = \begin{pmatrix} 4 & 2 & 3 \\ 8 & 4 & 5 \\ 6 & 3 & 9 \\ 7 & 6 & 2 \end{pmatrix}$$

In each bounding procedure, the computation of the lower bounds is illustrated for only one node, at each of levels 1 and 2. The lower bound for each node, the minimum lower bound for the unexplored nodes, at each level and the solutions for each bounding procedure for the above sample problem are given in Tables 2.1, 2.2, 2.3, 2.4 and 2.5.

In order to discuss the various bounding procedures, the following common notation is considered.

L	conflict level index
n	set of scheduled operations
$\bar{n}$	set of unscheduled operations
$c^L_{jm_\ell}$	completion time of node (jm_ℓ) at level L
$B^L(jm_\ell)$	lower bound for node (jm_ℓ) at level L
B^L	minimum lower bound on schedule time at level L
S^L	conflict set at level L.

Composite-based Bound LB I

The composite-based bound is expressed as the maximum of the job-based bound and the machine-based bound. In mathematical terms, the lower bound on the schedule time for the node (jm_ℓ) at level L is expressed such that

LB I = max [LB III, LB V]

where

LB III is the job-based bound, and

LB V is the machine-based bound.

First, the bounding procedure LB III has been suggested in [19]. This lower bound will be presented in this thesis in a mathematical form and rigorous notation. This bounding procedure, referred to as the job-based bound, determines the lower bound by the total processing time on each job in the conflict set, at level L, s^L.

The lower bound for node (jm_ℓ) at level L, $B^L(jm_\ell)$, can be stated such that

$$B^L(jm_\ell) = \max \left(\left[c^L_{jm_\ell} + \sum_{s=\ell+1}^{M} t_{jm_s} \right], \max_{\substack{i \\ i \varepsilon s^L \\ i \neq j}} \left[c^L_{jm_\ell} + \sum_{s=r}^{M} t_{im_s} \right] \right) \quad (1)$$

where

for job i, $m_r = m_\ell$

It should also be pointed out that m_ℓ represents a particular machine. The value of this lower bound is the maximum of two expressions. The first expression gives the bound for job j, which consists of two terms:

$c^L_{jm_\ell}$ the completion time of the job j on machine m_ℓ, at level L; and

$\sum_{s=\ell+1}^{M} t_{jm_\ell}$ the sum of the unscheduled operations of job j.

The second expression gives the maximum of the bounds for remaining jobs, i.e., other than job j, in the conflict set, at level L, s^L. It also consists of two terms:

$c^L_{jm_\ell}$ the completion time of the job j on machine m_ℓ, at level L; and

$\sum_{s=r}^{M} t_{jm_s}$ the sum of the unscheduled operations of job i including its operation on machine m_r which is the same machine as m_ℓ.

Second, the bounding procedure LB V has also been suggested in [19]. This lower bound will be presented in this thesis in a mathematical form and rigorous notation. This bounding procedure, referred to as the machine-based bound, determines the lower bound by the total processing time on each machine.

The lower bound on the schedule time for node (jm_ℓ) at level L, $B^L(jm_\ell)$, is expressed such that

$$B^L(jm_\ell) = \max \left(\left(c^L_{jm_\ell} + \sum_{\substack{i \in \bar{n} \\ m=m_\ell}} t_{im} \right), \right.$$

$$\left. \max_{\substack{m \\ m \neq m_\ell}} \left\{ \min_i \left(c^L_{im} - t_{im} \right) + \sum_{\substack{i=1 \\ i \in \bar{n}}}^{J} t_{im} \right\} \right)$$

In this bounding procedure, the earliest time at which an unscheduled operation can be started is found for each machine. The sum of the processing times of the unscheduled operations which require this machine is added to the earliest time at which unscheduled operation can be started on this machine.

The first expression gives the bound on machine m_ℓ. It consists of two terms:

$c^L_{jm_\ell}$	the completion time of job j on machine m_ℓ, at level L. This is also the earliest time at which an unscheduled operation can be started on machine m_ℓ, because the operation of another job in the conflict set, at level L, s^L, can be started on machine m_ℓ immediately after the completion of the operation, jm_ℓ; and
$\sum_{\substack{i\varepsilon\bar{n} \\ m=m_\ell}} t_{im}$	the sum of the processing times of the unscheduled operations of jobs (other than job j) which require the machine m which is the same as machine m_ℓ.

The second expression gives the maximum of the bounds on the machines other than machine m_ℓ. It consists of two terms:

$\min_{i\varepsilon\bar{n}} \left(c^L_{im} - t_{im}\right)$	the earliest time at which an unscheduled job i can be started on machine m; and
$\sum_{i=1}^{J} t_{im}$	the sum of the processing times of unscheduled jobs which require machine m $(m \neq m_\ell)$.

In order to illustrate the composite-based bound LB I we consider the same sample problem presented earlier and compute the lower bounds for only one node at each of levels 1 and 2. First, let us compute the lower-bounds using the job-based bound, LB III.

At level 1, there are two nodes (13) and (33). In other words the conflict set, at level 1, s^1, consists of job 1 and 3. It is interesting to illustrate the conflict among jobs 1 and 3 on machine 3, using the Gantt chart shown in Fig 2.1. The completion time matrix at level 1 temporarily updated for resolving conflict in favor of node (13) is such that

$$C^1(13) = \begin{bmatrix} 4 & 6 & 9 \\ 8 & 12 & 17 \\ 12 & 15 & 24 \\ 7 & 13 & 15 \end{bmatrix}$$

This becomes evident from the Gantt charts shown in Fig 2.2. The lower bound for node (13) at level 1 is computed such that

$$\begin{aligned} B^1(jm_2) &= B^1(13) \\ &= \max\left[\left(c^1_{13} + t_{11}\right), \max\left(c^1_{13} + (t_{33} + t_{31} + t_{32})\right)\right] \\ &= \max\left[(6 + 3), \max\left[6 + (6 + 3 + 9)\right]\right] \\ &= \max\left[9, 24\right] \\ &= 24 \end{aligned}$$

Let us now compute the lower bound for node (41) at level 2 as shown below:

At level 2, the conflict set, s^2, consists of three nodes (21), (31)

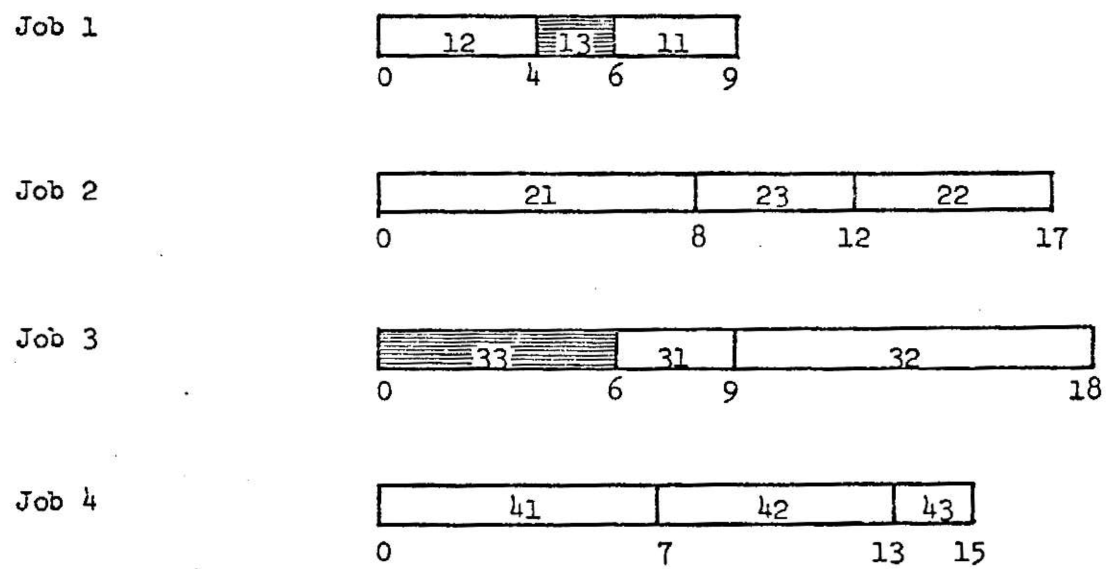

Figure 2.1

A Gantt Chart Depicting the Conflict Between Nodes (13) and (33) at Level 1

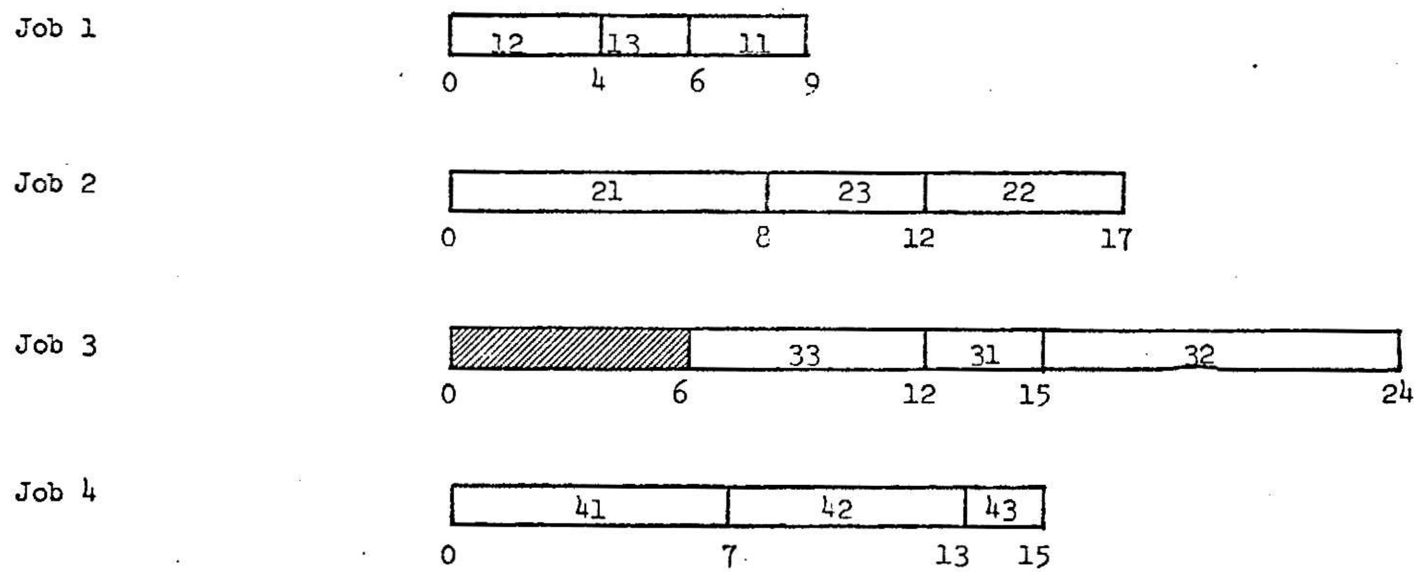

Figure 2.2

A Gantt Chart Depicting the Resolving of Conflict in Favor of Node (33) at Level 1 and (41). This is also illustrated using Gantt chart shown in Fig 2.3.

The completion time matrix temporarily updated for resolving conflict in favor of node (41) is such that

$$C^2(41) = \begin{pmatrix} 4 & 8 & 11 \\ 15 & 19 & 24 \\ 6 & 10 & 19 \\ 7 & 13 & 15 \end{pmatrix}$$

This becomes evident from the Gantt charts shown in Fig 2.4.

The lower bound for node (41) at level 2, $B^2(41)$ is such that

$$B^2(4m_1) = B^2(41)$$

$$= \max \left[c^2_{41} + (t_{42}+t_{43}), \max \begin{bmatrix} c^2_{41} + (t_{21}+t_{23}+t_{22}) \\ c^2_{41} + (t_{31}+t_{32}) \end{bmatrix} \right]$$

$$= \max \left[7 + (6+2), \max \begin{bmatrix} 7 + (8+4+5) \\ 7 + (3+9) \end{bmatrix} \right]$$

$$= \max \left[15, \max (24, 17) \right]$$

$$= \max \left[15, 24 \right]$$

$$= 24$$

Next illustrate the machine-based bound LB V, using the same sample problem for one node at each of levels 1 and 2. At level 1,

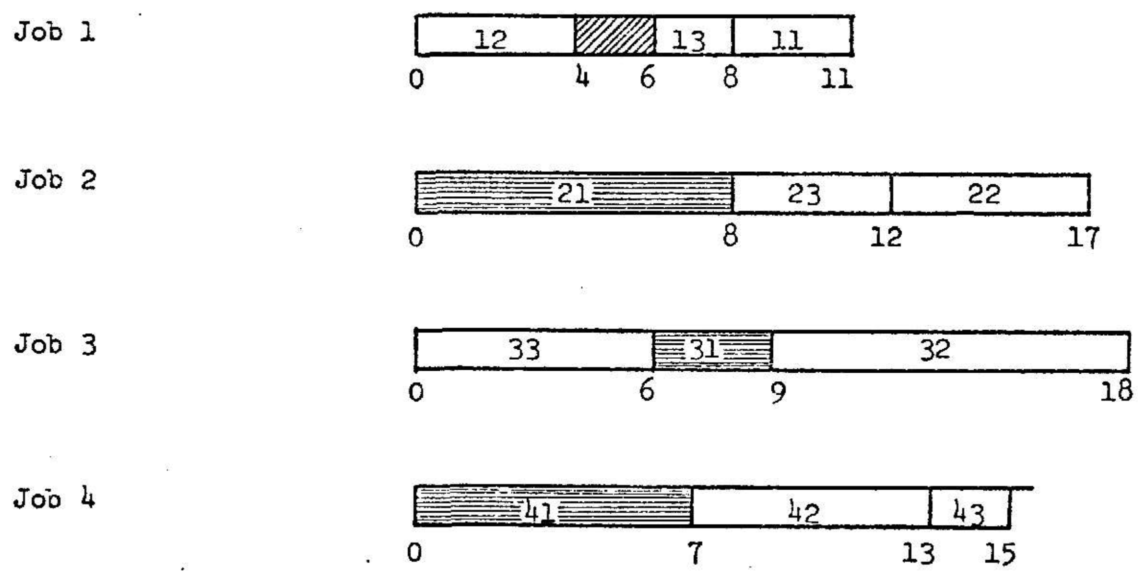

Figure 2.3

A Gantt Chart Depicting the Conflict Among Nodes (21), (31) and (41) at Level 2.

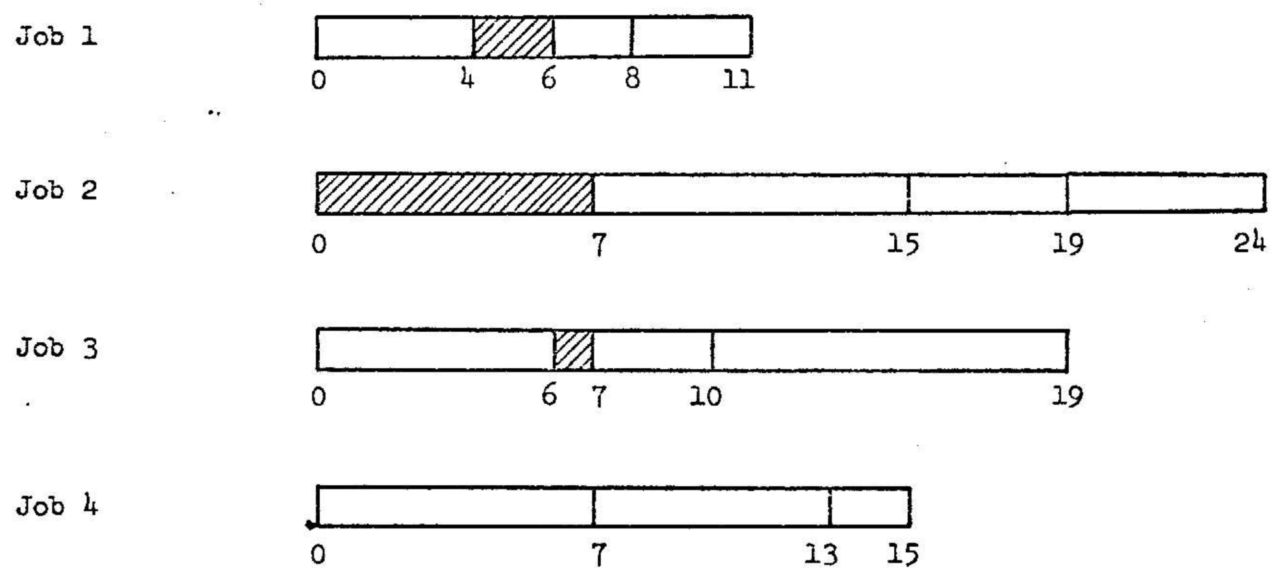

Figure 2.4

A Gantt Chart Depicting the Resolving of Conflict in Favor of Node (41), at Level 2.

the lower bound for node (13) can be computed as shown below. The completion time matrix at level 1, temporarily updated for resolving the conflict in favor of node (13) is such that

$$C^1(13) = \begin{bmatrix} 4 & 6 & 9 \\ 8 & 12 & 17 \\ 12 & 15 & 24 \\ 7 & 13 & 15 \end{bmatrix}$$

This also becomes evident from the Gantt chart shown in Fig 2.2. The conflict set at level 1, s^1, consists of nodes (13) and (33) as illustrated using the Gantt chart shown in Fig 2.1. The unscheduled operations at a conflict level have their completion time equal to or greater than the minimum of the completion times of the jobs in the conflict set. For example, the set of unscheduled jobs at level 1 for machine 3 consists of jobs 2, 3 and 4.

The lower bound for the node (13) at level 1, $B^1(13)$, is computed such that

$B^1(jm_2) = B^1(13)$

$$= \max \left\{ \begin{array}{l} \left[c_{13}^1 + (t_{23}+t_{33}+t_{43})\right], \\ \max \left\{ \begin{array}{l} \min\left[(c_{11}^1-t_{11}),(c_{21}^1-t_{21}),(c_{31}^1-t_{31}),(c_{41}^1-t_{41})\right] \\ \qquad + (t_{11}+t_{21}+t_{31}+t_{41}) \\ \min\left[(c_{22}^1-t_{22}),(c_{32}^1-t_{32}),(c_{42}^1-t_{42})\right] \\ \qquad + (t_{22}+t_{32}+t_{42}) \end{array} \right\} \end{array} \right\}$$

$$= \max \left\{ \left\{6 + (4+6+2)\right\}, \max \left\{ \begin{array}{l} \min \left\{(11-3),(8-8),(9-3),(7-7)\right\} + (3+8+3+7) \\ \\ \min \left\{(17-5),(18-9),(13-6)\right\} + (5+9+6) \end{array} \right\} \right\}$$

$$= \max \left\{18, \max \left\{21,27\right\}\right\}$$

$$= \max \left\{18,27\right\}$$

$$= 27$$

At level 2, the conflict set, s^2, consists of three nodes (11), (21), and (41), as illustrated using Gantt chart shown in Fig 2.5. The completion time matrix at level 2, temporarily updated for resolving the conflict in favor of node (41) is such that

$$C^2(41) = \begin{bmatrix} 4 & 6 & 10 \\ 15 & 19 & 24 \\ 12 & 15 & 24 \\ 7 & 13 & 15 \end{bmatrix}$$

This is also evident from the Gantt chart shown in Fig 2.6.

The lower bound for node (41) at level 2, $B^2(41)$, is computed such that

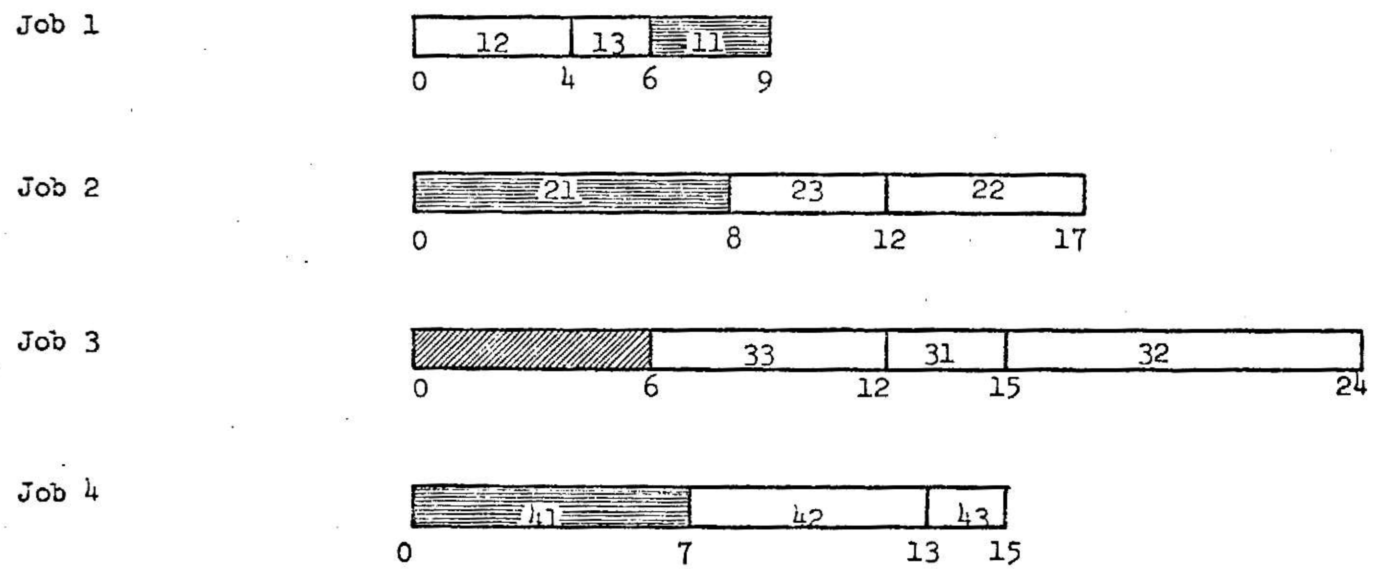

Figure 2.5

A Gantt Chart Depicting the Conflict Among Nodes (11), (21) and (41) at Level 2

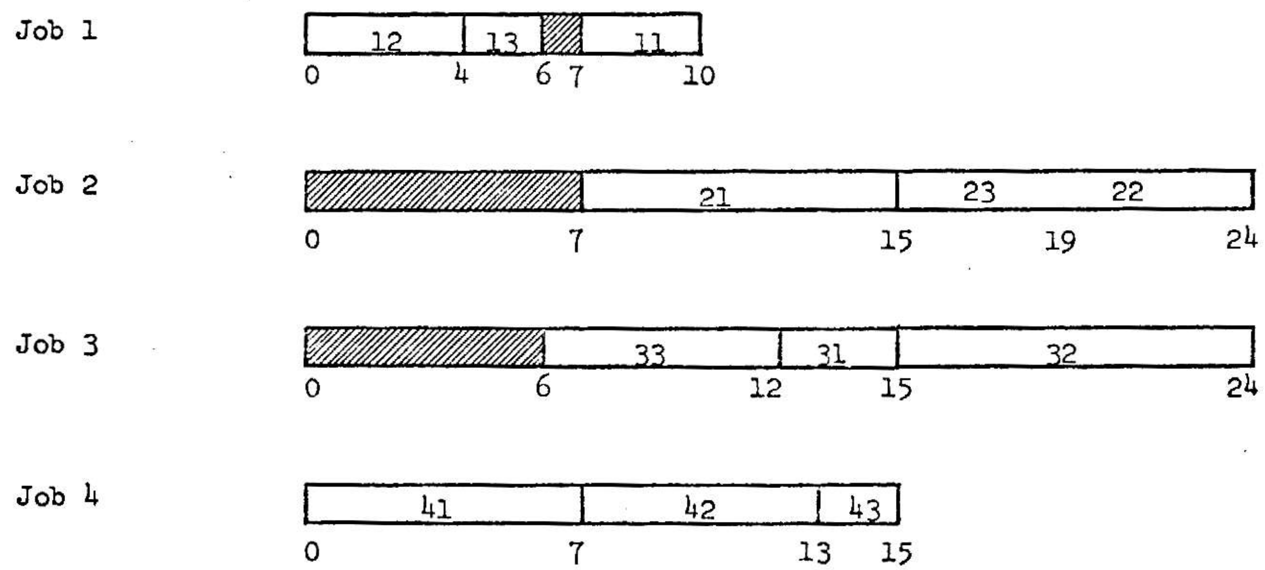

Figure 2.6

A Gantt Chart Depicting the Resolving of Conflict in Favor of Node (41) at Level 2.

$$B^2(4m_1) = B^2\ (41)$$

$$= \max \left[\left(c^2_{41} + (t_{11}+t_{21}+t_{31}) \right), \max \left[\begin{array}{l} \min \left[(c^2_{22}-t_{22}),(c^2_{32}-t_{32}),(c^2_{42}-t_{42}) \right] + (t_{22}+t_{32}+t_{42}) \\ \min \left[(c^2_{23}-t_{23}),(c^2_{33}-t_{33}),(c^2_{43}-t_{43}) \right] + (t_{23}+t_{33}+t_{43}) \end{array} \right] \right]$$

$$= \max \left[\left(7 + (3+8+3) \right), \max \left[\begin{array}{l} \min \left[(24-5),(24-9),(13-6) \right] + (5+9+6) \\ \min \left[(19-4),(12-6),(15-2) \right] + (4+6+2) \end{array} \right] \right]$$

$$= \max \left[21, \max \left[27,18 \right] \right]$$

$$= \max \left[21,27 \right]$$

Now let us compute the lower bounds for nodes (13) and (41) at levels 1 and 2 respectively, using the composite-based bound LB I. We can compute the lower bounds by using job-based bound LB III and machine-based bound LB V, as illustrated earlier and take the maximum of them as the composite-based bound. At level 1, the lower bound for node (13) can be computed using composite-based bound LB I such that

$$B^1(13) = \max\ [\text{LB III, LB V}]$$

$$= \max\ [24,\ 27]$$

$$= 27$$

Similarly at level 2, the lower bound for node (41) is computed such that

$$B^2(41) = \max\,[\text{LB III}, \text{LB V}]$$
$$= \max\,[24, 27]$$
$$= 27$$

Composite-based bound LB II

This bounding procedure is expressed as the maximum of the job-based bound and the machine-based bound. In mathematical terms, the lower bound on the schedule time is expressed such that

LB II = max [LB IV, LB V]

where

LB IV is the job-based bound; and

LB V is the machine-based bound.

First, the bounding procedure LB IV has been developed in [4]. This lower bound is presented in this thesis in a mathematical form and rigorous notation. This is also referred to as the job-based bound since the lower bound is determined by considering the total processing time on each job.

In this bounding procedure, the conflict among the last operation of all jobs is resolved. In other words, the idle time created by some of the unscheduled operations is considered. As mentioned earlier, a powerful bounding procedure considers the idle time due to the unscheduled operations and thus, produces the lower bounds as high as possible. Therefore it can be expected that this bounding procedure will produce more realistic lower bounds.

In order to consider the idle time created by the last operation for each job, it is necessary to know the machine on which each job has its last operation. In other words, we can check all machines from 1 to M to know how many jobs have their last operations on a particular machine.

It is necessary to know the completion time of the operation just preceding the last operation in order to resolve the conflict among the last operations of all jobs. The completion time of the operation just preceding the last operation can be computed as shown below.

For job i in the conflict set at level L, s^L, except the job j around which the conflict is resolved, the completion time of the operation just preceding the last operation, $c^L_{im_{M-1}}$, is computed such that

$$c^L_{im_{M-1}} = c_{jm_\ell} + \sum_{s=r}^{M-1} t_{im_s}$$

where

$$m_r = m_\ell$$

For all other jobs not in the conflict set at that level, and the job j around which the conflict is resolved, the completion time of the operation just preceding the last operation remains the same as in the previous completion time matrix. Thus, these completion times are known.

Let r be the number of jobs which have the last operation on a particular machine. Arrange the completion times, $c^L_{jm_{M-1}}$, of such r jobs in ascending order, and store them temporarily in a vector U such that

$$U = [U_1, U_2, \ldots, U_r]$$

Also, store temporarily, the corresponding times on the last machine,

t_{jm_M}, in a vector V such that

$$V = [V_1, V_2, \ldots, V_r]$$

Let $D_i = c^L_{im_M}$

$$D_1 = U_1 + V_1$$

$$D_2 = \max\,[D_1, U_2] + V_2$$

$$\vdots$$

$$D_{r-1} = \max\,[D_{r-2}, U_{r-1}] + V_{r-1}$$

$$D_r = \max\,[D_{r-1}, U_r] + V_r$$

There are two special cases of the above situation: (1) when there is no job having its last operation on a particular machine; and (2) when there is only one job having its last operation on a particular machine. It is not necessary to consider the former case since there is no job having its last operation on a particular machine. In the latter case, however, the completion time of the last operation of the only job i is computed such that

$$c^L_{im_M} = \max\left(c^L_{im_{M-1}}, \max_p \left(c^L_{pm_{M-1}}\right)\right) + t_{im_M}$$

where

m_M for job i is the same machine as m_{M-1} for job p.

This special case is considered in this thesis. The lower bound developed in [4] has been modified in this thesis by incorporating this feature. Thus, we know the completion time of the last operation of all jobs, $c^L_{im_M}$, obtained by resolving the conflict among the last operations for each job.

The lower bound on the schedule time for the node (jm_ℓ) at level L, $B^L(jm_\ell)$ is computed such that

$$B^L(jm_\ell) = \max_i \left\{ c^L_{im_M} \right\} \qquad i = 1, 2, \ldots, J$$

Second, the bounding procedure LB V has already been discussed under composite-based bound LB I.

In order to illustrate the composite-based bound LB II, let us consider the same sample problem presented earlier and compute the lower bounds for only one node at each of levels 1 and 2. First, let us compute the lower bounds using the job-based bound, LB IV.

At level 1, the conflict set, s^1, consists of two nodes (13) and (33). This conflict is shown using the Gantt chart in Figure 2.1. The completion time matrix at level 1 temporarily updated for resolving in favor of node (13) is such that

$$C^1(13) = \begin{bmatrix} 4 & 6 & 9 \\ 8 & 12 & 17 \\ 12 & 15 & 24 \\ 7 & 13 & 15 \end{bmatrix}$$

This becomes evident from the Gantt chart shown in Figure 2.1. The completion time of the operation just preceding the last operation for job 3 in the conflict set can also be computed such that

$$\begin{aligned} c^1_{3m_2} &= c^1_{31} \\ &= c^1_{13} + (t_{33} + t_{31}) \\ &= 6 + (6 + 3) \\ &= 15 \end{aligned}$$

For other jobs, which are not the in the conflict set, and for job 1, around which the conflict is resolved, the completion time of the operation just preceding the last operation remain the same as in the previous completion time matrix.

Let us check machines from 1 to 3.

For machine 1, the only job which has its last operation on this particular machine is job 1. The completion time of last operation of job 1 is computed such that

$$
\begin{aligned}
c^1_{1m_3} &= c^1_{11} \\
&= \max\left(c^1_{13},\ \max\left(c^1_{31}\right)\right) + t_{11} \\
&= \max\ [6,\ 15] + 3 \\
&= 15 + 3 \\
&= 18
\end{aligned}
$$

The number of jobs having their last operation on machine 2 is 2, i.e., $r = 2$.

The vectors U and V are formed such that

$$
\begin{aligned}
U &= \left(c^1_{23},\ c^1_{31}\right) \\
&= [12,\ 15] \qquad ; \qquad \text{and} \\
V &= \left(t_{23},\ t_{32}\right) \\
&= [5,\ 9]\,.
\end{aligned}
$$

$$
\begin{aligned}
D_1 &= c^1_{22} \\
&= U_1 + V_1 \\
&= 12 + 5 \\
&= 17 \\
D_2 &= c^1_{32} \\
&= \max\left(D_1,\ U_2\right) + V_2 \\
&= \max\ [17,\ 15] + 9
\end{aligned}
$$

$$= 17 + 9$$

$$= 26$$

The only job having its last operation on machine 3 is job 4. The completion time of the last operation of job 4 is computed such that

$$c^1_{43} = \max\left\{c^1_{42}, \max\left\{c^1_{13}, c^1_{13}\right\}\right\} + t_{43}$$

$$= \max\left\{13, \max[6, 12]\right\} + 2$$

$$= \max[13, 12] + 2$$

$$= 13 + 2$$

$$= 15$$

The conflict among the last operation for each job can be resolved using the Gantt chart as shown in Figure 2.7.

The lower bound for node (13) at level 1, $B^1(13)$, is computed such that

$$B^1(13) = \max\left\{c^1_{11}, c^1_{22}, c^1_{32}, c^1_{43}\right\}$$

$$= \max[18, 17, 26, 14]$$

$$= 26$$

At level 2, the conflict set, s^2, consists of nodes (21), (31) and (41). The completion time matrix at level 2, temporarily updated in favor of node (41), is such that

$$C^2(41) = \begin{bmatrix} 4 & 8 & 11 \\ 15 & 19 & 24 \\ 6 & 10 & 19 \\ 7 & 13 & 15 \end{bmatrix}$$

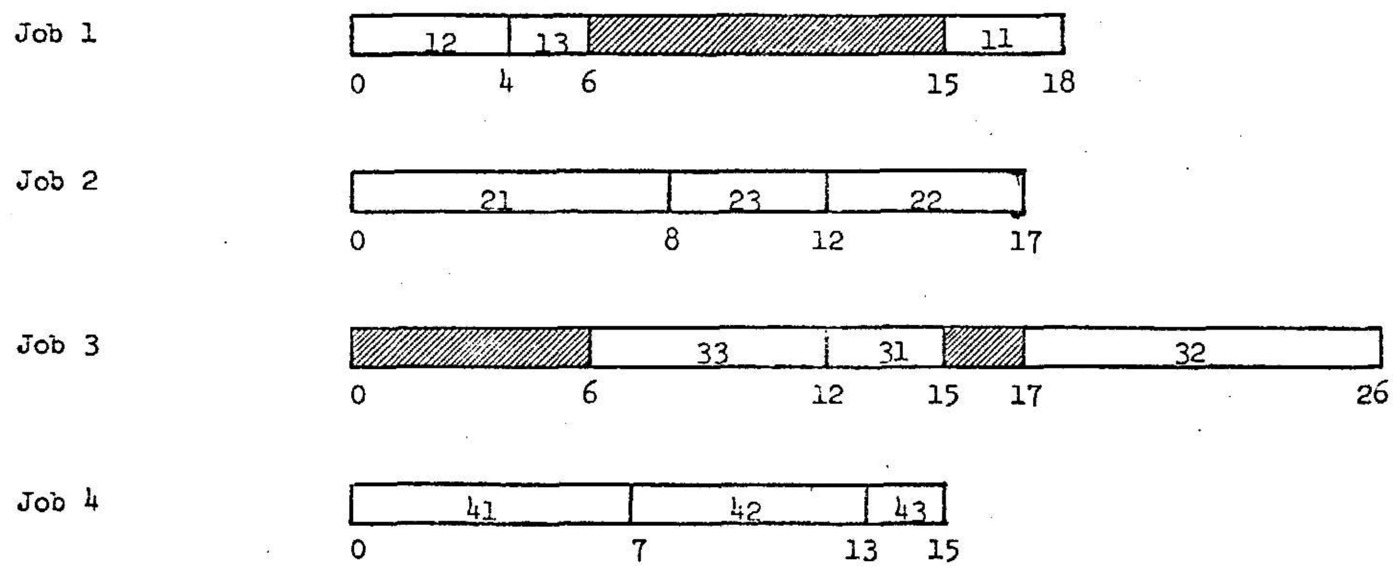

Figure 2.7

A Gantt Chart Depicting the Resolving of Conflict Among the Last Operation for Each Job at Level 1

This becomes evident from the Gantt chart , shown in Figure 2.3.

From the above completion time matrix, we get the updated completion time of the operation, just preceding the last operation, for each job.

The completion time of the operation, just preceding the last operation, for job 2 and 3 can also be computed such that

$$
\begin{aligned}
c^2_{2m_2} &= c^2_{23} \\
&= c^2_{41} + (t_{21} + t_{23}) \\
&= 7 + (8 + 4) \\
&= 19 \qquad ; \text{ and}
\end{aligned}
$$

$$
\begin{aligned}
c^2_{3m_2} &= c^2_{31} \\
&= c^2_{41} + t_{31} \\
&= 7 + 3 \\
&= 10
\end{aligned}
$$

For job 1, which is not in the conflict set and for job 4, around which the conflict is resolved, the completion time of the operation just preceding the last operation remain the same as in the previous completion time matrix.

Let us check machines from 1 to 3.

For machine 1, the only job having its last operation on this particular machine is job 1. The completion time of the last operation of job 1 is computed such that

$$
\begin{aligned}
c^2_{11} &= \max\left[c^2_{13}, \max\left(c^2_{31}\right)\right] + t_{11} \\
&= \max\,[8, 10] + 3
\end{aligned}
$$

$= 10 + 3$

$= 13$

The number of jobs, having the last operation on machine 2 is 2 i.e. $r = 2$.

The vectors U and V are formed such that

$$U = \left[c^2_{31}, c^2_{23}\right]$$

$$= [10, 19] \qquad ; \quad \text{and}$$

$$V = \left[t_{32}, t_{22}\right]$$

$$= [9, 5].$$

$$D_1 = c^2_{32}$$

$$= U_1 + V_1$$

$$= 10 + 9$$

$$= 19$$

$$D_2 = c^2_{22}$$

$$= \max\left[D_1, U_2\right] + V_2$$

$$= \max [19, 19] + 5$$

$$= 24$$

The only job having the last operation on machine 3 is job 4. The completion time of the last operation of job 4 is computed such that

$$c^2_{43} = \max\left[c^2_{42}, \max\left[c^2_{13}, c^2_{13}\right]\right] + t_{43}$$

$$= \max\left[13, \max [8, 19]\right] + 2$$

$$= \max [13, 19] + 2$$

$$= 19 + 2$$

$$= 21$$

The conflict among the last operation for each job can be resolved using the Gantt charts shown in Figure 2.8.

We have already illustrated the machine-based bound LB V, using the same sample problem.

Now let us compute the lower bounds for nodes (13) and (41) at levels 1 and 2 respectively, using the composite-based bound LB II. As illustrated earlier, we can compute the lower bounds by using job-based bound LB IV and machine-based bound LB V and take the maximum of the two, as the composite-based bound.

At level 1, the lower bound for node (13), $B^1(13)$ can be computed using the composite based bound LB II such that

$$\begin{aligned} B^1(13) &= \max\ [\text{LB IV, LB V}] \\ &= \max\ [26,\ 27] \\ &= 27 \end{aligned}$$

Similarly at level 2, the lower bound for node (41), $B^2(41)$ can be computed using the composite-based bound LB II such that

$$\begin{aligned} B^2(41) &= \max\ [\text{LB IV, LB V}] \\ &= \max\ [29,\ 27] \\ &= 29 \end{aligned}$$

2.3 Sample Problem

In order to demostrate the branch-and-bound technique, the same sample problem, consisting of four jobs and three machines presented earlier, is solved using the computational algorithm that shall be described in formal steps in section 2.4. For convenience, the machine ordering and processing time matrices are reproduced below.

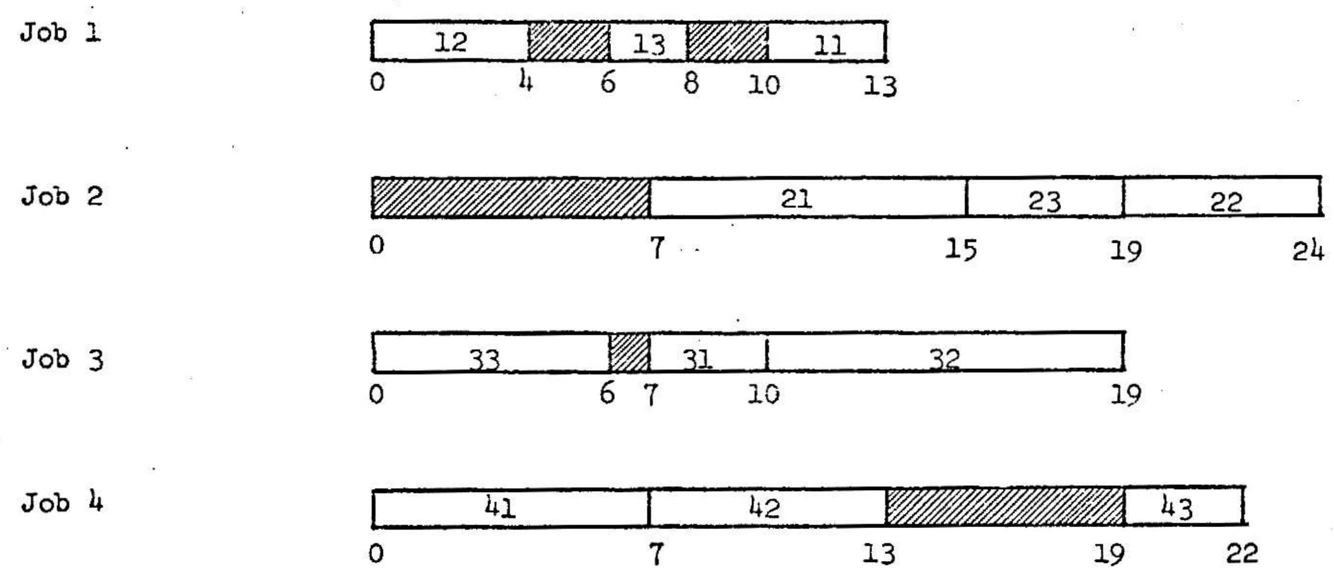

Figure 2.8

A Gantt Chart Depicting the Resolving of Conflict Among the Last Operations for Each Job at Level 2

$$M = \begin{bmatrix} 12 & 13 & 11 \\ 21 & 23 & 22 \\ 33 & 31 & 32 \\ 41 & 42 & 43 \end{bmatrix} \qquad T = \begin{bmatrix} 4 & 2 & 3 \\ 8 & 4 & 5 \\ 6 & 3 & 9 \\ 7 & 6 & 2 \end{bmatrix}$$

Refer the scheduling tree shown in Figure 2.9 and the scheduling table shown in Table 2.6 throughout all the steps in order to follow the solution easily.

Step 1. Set conflict level L = 1 and initial schedule time $T_0(s) = \infty$. The initial completion time matrix, $C^1(jm)$, regardless of any conflict is constructed as follows:

$$C^1(jm) = \begin{bmatrix} 4 & 6 & 9 \\ 8 & 12 & 17 \\ 6 & 9 & 18 \\ 7 & 13 & 15 \end{bmatrix}$$

Construct the scheduling table as shown in Table 2.6. Enter the completion time of the first operation of each job, i.e., c^1_{12}, c^1_{21}, c^1_{33} and c^1_{41} equal 4, 8, 6 and 7, respectively under the appropriate nodes. According to step 1.5, find τ such that

$$\tau = \min_{jm} \left\{c^1_{12}, c^1_{21}, c^1_{33}, c^1_{41}\right\}$$

$$= \min\,[4, 8, 6, 7]$$

$$= 4.$$

Step 2. In machine blocks 1 and 3, there is no completion time equal to τ. In machine block 2, there is only one completion time, i.e., c^1_{12}, equal to τ. Therefore there is no conflict existing. According to step 2.2, go to step 7.

Step 7. According to step 7.1, enter the next operation of job 1, whose completion time, c_{13}^{1}, is 6.

τ, i.e., 4, is not the highest number at level 1 in the scheduling Table 2.6. Therefore according to step 7.2.1, set $\tau = 6$ where 6 is the next higher value at level 1 in the scheduling Table. Go to step 2.

Step 2. In machine blocks 1 and 2, there is no entry equal to τ. However, in machine block 3, the completion time of node (13) and node (33) are equal to τ.

Check for conflict:

$$\tau + t_{33} > c_{33}^{1}, \text{ i.e., } 6 + 6 = 12 > 6$$

According to step 2.1, a conflict exists and therefore go to step 3

Step 3. Compute the lower bounds for nodes (13) and (33) in the conflict set at level 1, s^{1}, using the bounding procedure LB I such that

Node	(13)	(33)
Lower-bound	27	27

Step 4. Search for the unexplored node(s) with the minimum lower bound at level 1. The minimum lower bound at level 1, B^{1}, is such that

$$B^{1} = \min\,[27, 27]$$
$$= 27 \text{ for nodes (13) and (33)}$$

Step 5. B^{1} is less than T_0 i.e. 27 is less than ∞. Therefore, according to step 5.1, go to step 6.

Step 6. Since a tie exists for the minimum lower-bound, according to step 6.1, break the tie by Left Hand Rule in favor of node (13). Set $L = 1 + 1 = 2$ and update the completion time matrix such that

$$C^2(13) = \begin{bmatrix} 4 & 6 & 9 \\ 8 & 12 & 17 \\ 12 & 15 & 24 \\ 7 & 13 & 15 \end{bmatrix}$$

Step 7. The completion-time of node (13) is equal τ, i.e. $c^2_{13} = 6$. According to step 7.1, enter the completion time of the next operation of job 1, i.e., c^2_{11}.

According to step 7.2.1, set $\tau = 7$ since previous τ is not the highest entry at level 2 and go to step 2.

Step 2. In machine block 1, there is one job with completion time equal to τ and two jobs with completion time greater than τ. Check for conflict:

$$\tau + t_{11} > c^2_{11} \quad \text{i.e.,} \quad 7 + 3 = 10 > 9$$

$$\tau + t_{21} > c^2_{21} \quad \text{i.e.,} \quad 7 + 8 = 15 > 8$$

According to step 2.1, conflict exists and therefore, go to step 3.

Step 3. The lower-bounds are computed for each node in the conflict set at level 1, s^1, by bounding procedure LB I such that

Node	(11)	(21)	(41)
Lower-bound	35	32	27

Step 4. At level 2, search for minimum unexplored node(s). The minimum lower-bound at level 2, B^2, is such that

$B^2 = 27$ for node (41)

Step 5. Since B^2 is less than T_0 i.e. 27 is less than ∞, according to step 5.1, go to step 6.

Step 6. A tie does not exist for the minimum lower-bound. Set $L = 2 + 1 = 3$ and update the completion time matrix at level 3, $C^3(41)$, such that

$$C^3(41) = \begin{bmatrix} 4 & 6 & 10 \\ 15 & 19 & 24 \\ 12 & 15 & 24 \\ 7 & 13 & 15 \end{bmatrix}$$

Step 7. The completion time of node (41) is equal to τ. Therefore enter the completion time of the next operation of job 4 i.e. c^3_{42}. According to step 7.2.1, set $\tau = 10$ and go to step 2.

Step 2. Under machine block 1, there are two nodes, one with completion time equal to τ and the other with a completion time higher than τ.

Check for conflict:

$$\tau + t_{21} > c^3_{21} \quad \text{i.e.} \quad 10 + 8 = 18 > 15$$

According to step 2.1, the conflict exists and go to step 3.

Step 3. Compute the lower-bounds for each node in the conflict set at level 3, s^3, using bounding procedure LB I such that

Node	(11)	(21)
Lower-bounds	27	27

Step 4. Searching for the minimum unexplored node(s) at level 3, we find that the minimum lower-bound, B^3, is such that

$B^3 = 27$ for nodes (11) and (21)

Step 5. Since B^3 is less than T_0, i.e. 27 is less than ∞, according to step 5.1, go to step 6.

Step 6. A tie exists for the minimum lower-bound. According to step 6.1, break the tie by Left Hand Rule in favor of node (11) Set $L = 3 + 1 = 4$ and update the completion time matrix such that

$$c^4(11) = \begin{bmatrix} 4 & 6 & 10 \\ 18 & 22 & 27 \\ 12 & 15 & 24 \\ 7 & 13 & 15 \end{bmatrix}$$

Step 7. Since node (11) is the last operation of job 1, go to step 7.2.1

According to step 7.2.1, set τ = 12 and go to step 2

Step 2. Since there is only one job with completion time equal to τ in machine block 3, there is no conflict existing. Therefore according to step 2.2, go to step 7

Step 7. According to step 7.1, enter the completion time of the next operation of job 3, i.e., c^4_{31}

According to step 7.2.1, set τ = 13 and go to step 2.

Step 2. Since there is only one job with $c^4_{jm_\ell} = \tau$ in machine block 2 and no other job with has $c^4_{jm_\ell} \geq \tau$, there is no conflict existing. According to step 2.2, go to step 7.

Step 7. According to step 7.1, enter the completion time of the next operation of job 4, i.e. c^4_{43}.

According to step 7.2.1, set τ = 15 and go to step 2.

Step 2. In machine block 1, there are two entries with $c^4_{jm_\ell} \geq \tau$

Check for conflict:

$$\tau + t_{21} > c^4_{21} \quad \text{i.e.} \quad (5 + 8 = 23 > 18$$

According to step 2.1, conflict exists and therefore go to step 3.

Step 3. The lower bounds for each node in the conflict set at level 4, s^4, are computed by using bounding procedure LB I such that

Node	(21)	(31)
Lower bounds	35	32

Step 4. Search for the minimum unexplored node(s) at level 4. Node (31) has the minimum lower bound such that

$$B^4 = \min_{jm_\ell} [35, 32] = 32$$

Step 5. Since B^4 is less than $T_0(s)$ i.e., 32 is less than ∞, go to step 6

Step 6. For the minimum lower bound, there is no tie existing. Therefore branch from node (31) and set L = L + 1 = 4 + 1 = 5. Update the completion time matrix at level 5 such that

$$C^5(31) = \begin{bmatrix} 4 & 6 & 10 \\ 23 & 27 & 32 \\ 12 & 15 & 24 \\ 7 & 13 & 15 \end{bmatrix}$$

Step 7. The completion time of node (31) is equal to τ. According to step 7.1, enter the completion time of the next operation of node (31) i.e. c^5_{32}

According to step 7.2.1, set τ = 23 since 15 is not the highest entry at level 5. Go to step 2

Step 2. There is only one node in machine bock 1 with completion time equal to τ, no conflict exists.

According to step 2.2, go to step 7.

Step 7. The completion time of node (21) is equal to τ. Therefore according to step 7.1, enter the completion time of the next operation of node (21) i.e. c^5_{23}

According to step 7.2.1, set $\tau = 24$.

Go to step 2

Step 2. In machine block 2, there is only one entry with completion time equal to τ. Since there is no other entry with $c^5_{jm_\ell} \geq \tau$, no conflict exists. Therefore according to step 2.2, go to step 7.

Step 7. The completion time of node (23) is equal to τ. Therefore according to step 7.1, enter the completion time of the next operation of node (23) i.e. c^5_{22}.

According to step 7.2.1, set $\tau = 32$ since 27 is not the highest entry at level 5. Go to step 2.

Step 2. In machine block 2, there is one node (22) with completion time equal to τ. Since there is no other node with $c^5_{jm_\ell} \geq \tau$, no conflict exists. Therefore, according to step 2.2, go to step 7.

Step 7. The completion time of node (22) is equal to τ. There is no next operation of node (22).

τ is the highest number in the scheduling table at level 5. Therefore according to step 7.2.2, set $T_0(s) = \tau = 32$ and go to step 8.

Step 8. Back-track along the same branch of the scheduling tree, setting $L = L - 1 = 5 - 1 = 4$. Compare the lower-bounds of unexplored node(s) with the updated solution $T_0(s)$. There is no unexplored node at this level with lower-bound less than $T_0(s)$. Therefore, according to step 8.2, go to step 9

Step 9. Since $L > 1$ i.e. $4 > 1$, according to step 9.1, go to step 8.

Step 8. Backtrack along the same branch of the scheduling tree by setting $L = L - 1 = 4 - 1 = 3$, and compare the lower bound of the unexplored node(s) with the updated solution $T_0(s)$. The unexplored node (21) has lower bound 27 which is less than $T_0(s)$ or 32. Therefore, according to step 8.1, set $\tau = \min_{j \epsilon s^3} [c^3_{jm_\ell}] = \min [10, 15] = 10$; and go to step 4.

Step 4. At level 3, the minimum lower bound for unexplored node(s), 3, is 27 for node (21)

Step 5. According to step 5.1, go to step 6 since B^3 is less than $T_o(s)$ i.e. 27 is less than 32.

Step 6. Since there is no tie existing, according to step 6.2, set $L = L + 1 = 3 + 1 = 4$ and update the completion time matrix such that

$$c^4(21) = \begin{bmatrix} 4 & 6 & 18 \\ 15 & 19 & 24 \\ 12 & 15 & 24 \\ 7 & 13 & 15 \end{bmatrix}$$

Go to step 7.

Step 7. There is no node with completion time equal to τ. According to step 7.2.1, set $\tau = 12$ and go to step 2.

Step 2. The node (33) is the only node with completion time equal to τ. There is no other node in machine block 3, with $c^4_{jm_\ell} \geq \tau$. According to step 2.2, go to step 7 since no conflict exists.

Step 7. The completion time of node (33) is equal to τ. Enter the completion time of the next operation of node (33) i.e. c^4_{31} According to step 7.2.1, set $\tau = 13$ since previous τ is not the highest number at level 4 and go to step 2

Step 2. The node (42) in machine block 2 has the completion time equal to τ. There is no other node in machine block 2 with $c^4_{jm_\ell} \geq \tau$.

According to step 2.2, there is no conflict existing. Therefore, go to step 7.

Step 7. The completion time of node (42) is equal to τ. Enter the completion time of the next operation of node (42) i.e. c_{43}^{4} According to step 7.2.1, set τ = 15 since previous τ is not the highest number at level 4 and go to step 2.

Step 2. In machine block 1, there are three operations with $c_{jm_\ell}^{4} \geq \tau$.

Check for conflict:

$\tau + t_{21} > c_{21}^{4}$ i.e., 15 + 8 = 23 > 15

$\tau + t_{31} > c_{31}^{4}$ i.e., 15 + 3 = 18 > 15

According to step 2.1, conflict exists and therefore go to step 3.

Step 3. The lower bounds for each node in the conflict set at level 4, s^4, are computed by using bounding procedure LB I such that

Node	(11)	(21)	(31)
Lower bounds	35	32	32

Step 4. Search for the minimum unexplored node(s) at level 4. The minimum lower bound at level 4 is such that

B^4 = 32 for nodes (21) and (31)

Step 5. B^4 is equal to $T_0(s)$ i.e., 32. Therefore, according to step 5.2, go to step 8.

According to step 8, the backtracking process is continued along the same branch of the scheduling tree, by setting L = L - 1 = 4 - 1 = 3. At level 1, the unexplored node (33) has lower bound less than the updated schedule time $T_0(s)$. The branching, bounding and backtracking processes are carried out till an updated solution $T_1(s)$, i.e., 27 is obtained. The backtracking process is again continued to find an unexplored node with lower bound less than the updated solution, $T_1(s)$. It is found that there is no unexplored node with lower bound less than the updated solution, $T_1(s)$. Hence, the schedule time, $T_1(s)$ or 27, is the minimum schedule time. The number of nodes explored is 24. Figure 2-10 shows the Gantt chart of the solution.

2.4 Computational Algorithm

The branch-and-bound algorithm discussed above is stated in formal steps below:

Step 1: Initialize the scheduling table.

1.1. Set level index $L = 1$, and schedule time $T(s) = \infty$.

1.2. Compute the initial completion time matrix regardless of any conflict, $C^1(jm)$.

1.3. Construct the scheduling table.

1.4. Enter the completion time of the first operation of each job at level L, c^L_{jm}

1.5. Find the minimum completion time at level L such that

$$\tau = \min_{jm} [c^L_{jm}]$$

Step 2: Check for conflict, within each machine block, between job ending at time τ and those with $c^L_{jm_\ell} \geq \tau$

2.1. If a conflict exists such that

$$\tau + t_{jm_\ell} > c^L_{jm_\ell}, \text{ go to step 3.}$$

2.2. If there is no conflict such that

$$\tau + t_{jm_\ell} \leq c^L_{jm_\ell}, \text{ go to step 7.}$$

Step 3: Compute the lower-bounds at level L, for each node under conflict, $B^L(jm_\ell)$, by a particular bounding procedure.

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

$$B^L = \min_{jm_\ell} [B^L(jm_\ell)]$$

Step 5: Check the minimum lower bound at any level L:

5.1. If $B^L < T(s)$, go to step 6.

5.2. If $B^L \geq T(s)$, go to step 8.

Step 6: Branch from an unexplored node with the minimum lower bound:

6.1. If a tie exists, break it by a particular rule. Set $L = L + 1$ and update the completion time matrix $C^L_{(jm_\ell)}$. Go to step 7.

6.2. If a tie does not exist, branch from that node. Set $L = L + 1$ and update the completion time matrix, $C^L(jm_\ell)$. Go to step 7.

Step 7: Update the scheduling table

7.1. Enter the completion time, $c^L_{jm_\ell}$ of next operation of the jobs with completion time equal to τ.

7.2. Check τ:

7.2.1. If τ is not the highest number at level L, set $\tau = \tau'$ where τ' is the next higher number at level L and go to step 2.

7.2.2. If τ is the highest number at level L, set $T(s) = \tau$ and go to step 8.

Step 8: Backtrack along the same branch of the scheduling table by setting $L = L - 1$. Compare the lower bounds for all unexplored nodes at this level:

8.1. If there exist one or more nodes with a lower bound such

$$B^L(jm_\ell) < T(s),$$

set $\tau = \min\limits_{j\varepsilon s^L} (c^L_{jm_\ell})$ and go to step 4.

8.2. If all unexplored nodes have lower bounds such that

$$B^L(jm_\ell) \geq T(s), \text{ go to step 9.}$$

Step 9: Check for an optimal solution:

9.1. If $L > 1$, go to step 8

9.2. If $L = 1$, $T(s)$ is an optimal schedule time.

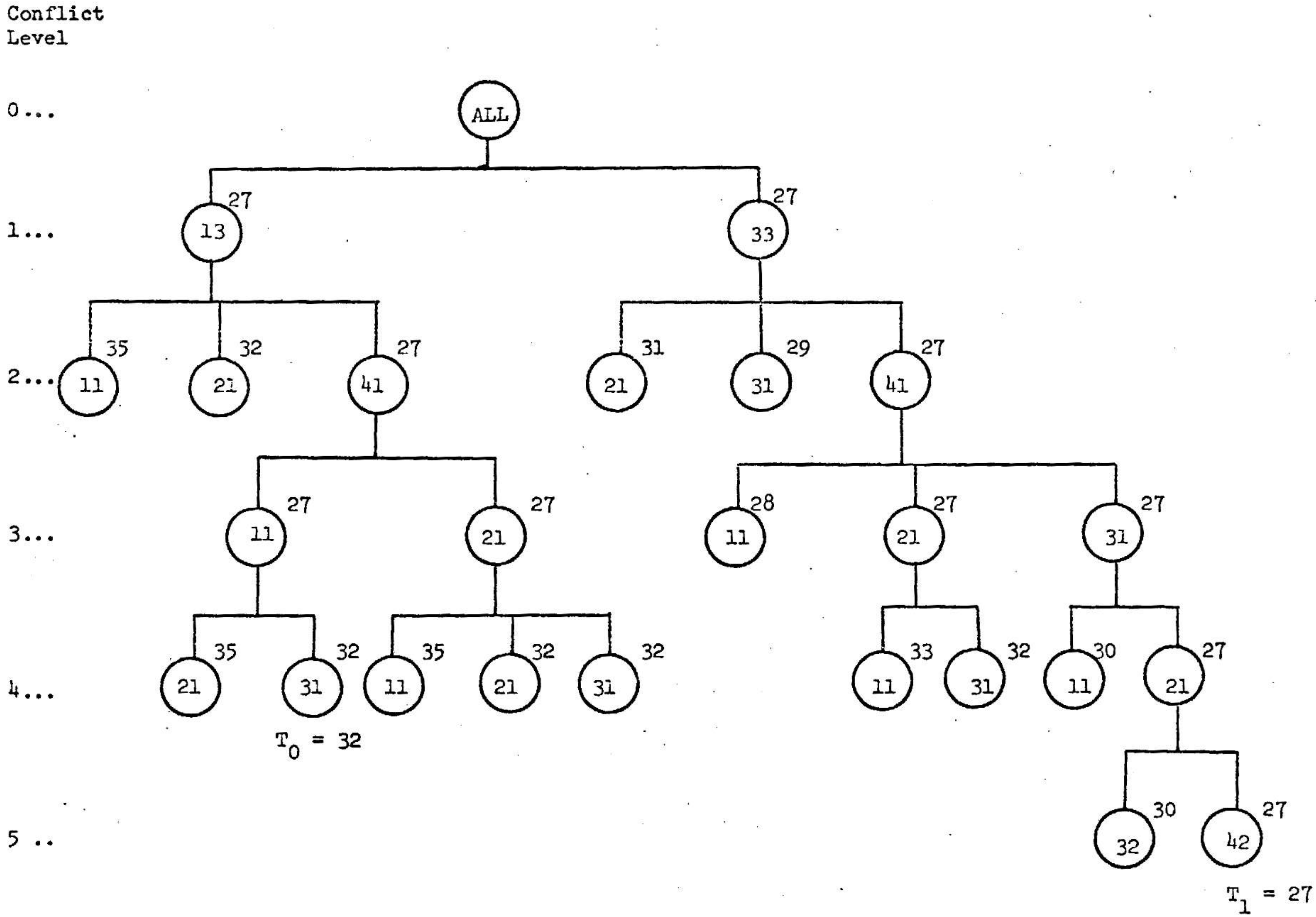

Figure 2.9

The Scheduling Tree for the Sample Problem Using Composite-Based Bound LB I

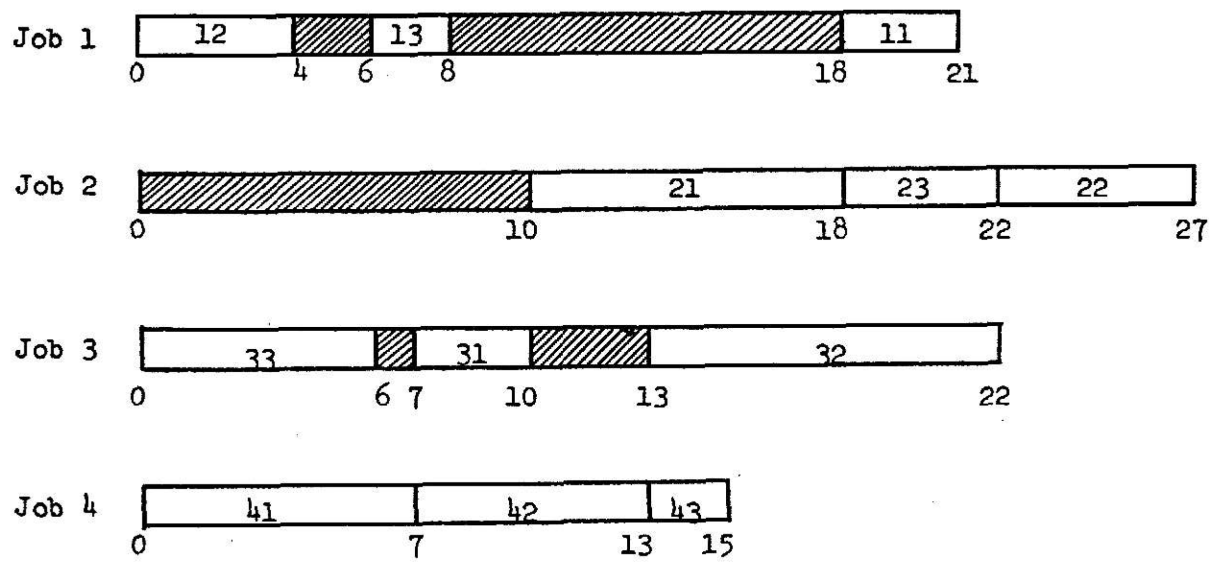

Figure 2.10

A Gantt Chart Depicting an Optimal Schedule of the Sample Problem

TABLE 2.1 . SOLUTION OF THE SAMPLE PROBLEM USING COMPOSITE-BASED BOUNDING LB I

Back-track i	Conflict Level L	Node (jm_ℓ)	Lower Bounds LB III	Lower Bounds LB V	Lower Bound LB II $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	1	(13)	24	27	27	27	
		(33)	18	27	27		
	2	(11)	26	35	35		
		(21)	23	32	32		
		(41)	24	27	27	27	
	3	(11)	27	27	27	27	
		(21)	24	27	27		
	4	(21)	30	35	35		
0		(31)	32	29	32	32	32
	3	(11)	27	27	27		
		(21)	24	27	27	27	
	4	(11)	35	35	35		
		(21)	27	32	32		
		(31)	32	29	32		
	1	(13)	24	27	27		
		(33)	18	27	27	27	
	2	(21)	23	31	31		
		(31)	26	29	29		
		(41)	24	27	27	27	
	3	(11)	28	27	28		
		(21)	27	27	27	27	
		(31)	27	27	27		
	4	(11)	30	33	33		
		(31)	27	32	32		
	3	(11)	28	27	28		
		(21)	27	27	27		
		(31)	27	27	27	27	

TABLE 2.1. SOLUTION OF THE SAMPLE PROBLEM USING COMPOSITE-BASED BOUNDING LB I (continued)

Back-track i	Conflict Level L	Node (jm_ℓ)	Lower Bounds LB III	Lower Bounds LB V	Lower Bound LB II $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	4	(11)	30	27	30		
		(21)	27	27	27	27	
	5	(32)	27	30	30		
1		(42)	22	27	27	27	27

TABLE 2.2. SOLUTION OF THE SAMPLE PROBLEM USING COMPOSITE-BASED BOUNDING LB II

Back-track	Conflict Level	Node	Lower Bounds		Lower Bound LB II	Minimum Lower Bound	Schedule Time
			LB IV	LB V			
i	L	(jm_ℓ)			$B^L(jm_\ell)$	B^L	T_i
	1	(13)	26	27	27	27	
		(33)	23	27	27		
	2	(11)	29	35	35		
		(21)	26	32	32		
		(41)	29	27	29	29	
	3	(11)	29	27	29	29	
		(21)	29	27	29		
	4	(21)	35	35	35		
0		(31)	32	29	32	32	32
	3	(11)	29	27	29		
		(21)	29	27	29	29	
	4	(11)	35	35	35		
		(21)	32	32	32		
		(31)	32	29	32		
	1	(13)	26	27	27		
		(33)	23	27	27	27	
	2	(21)	25	31	31		
		(31)	26	29	29		
		(41)	24	27	27	27	
	3	(11)	28	27	28		
		(21)	32	27	32		
		(31)	27	27	27	27	
	4	(11)	30	27	30		
		(21)	27	27	27	27	
	5	(32)	27	30	30		
1		(42)	27	27	27	27	27

TABLE 2.3. SOLUTION OF THE SAMPLE PROBLEM USING LB III

No. of Backtrack i	Conflict Level L	Node (jm_ℓ)	Lower Bounds $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	1	(13)	24		
		(33)	18	18	
	2	(21)	23	23	
		(31)	26		
		(41)	24		
	3	(11)	26	26	
		(31)	26		
		(41)	27		
	4	(31)	29	29	
		(41)	30		
	5	(22)	26	26	
		(32)	28		
	6	(32)	34		
0		(42)	36	34	34
	5	(22)	26		
		(32)	28	28	
	6	(22)	37		
		(32)	31	31	
		(42)	36		
	7	(22)	36		
		(42)	34		
	4	(31)	29		
		(41)	30	30	
	5	(32)	38		
1		(42)	33		33
	3	(11)	26		
		(31)	26	26	
		(41)	27		

TABLE 2.3. SOLUTION OF THE SAMPLE PROBLEM USING LB III (continued)

No. of Backtrack i	Conflict Level L	Node (jm_ℓ)	Lower Bounds $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	4	(11)	29		
		(41)	26	26	
	5	(22)	26		
		(32)	25	25	
	6	(22)	34		
		(32)	28	28	
		(42)	33		
	7	(22)	33		
2		(32)	31	31	31
	5	(22)	26	26	
		(32)	25		
	6	(32)	34		
		(42)	33		
	4	(11)	29	29	
		(41)	26		
	5	(22)	26		
		(32)	25	25	
	6	(22)	33		
		(42)	32		
	5	(22)	26	26	
		(32)	25		
	6	(32)	34		
		(42)	36		
	3	(11)	26		
		(31)	26		
		(41)	27	27	
	4	(22)	25	25	
		(42)	26		

TABLE 2.3. SOLUTION OF THE SAMPLE PROBLEM USING LB III (continued)

No. of Backtrack i	Conflict Level L	Node (jm_ℓ)	Lower Bounds $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	5	(11)	30		
		(31)	27	27	
	6	(32)	35		
		(42)	32		
	5	(11)	30	30	
		(31)	27		
	6	(32)	38		
		(42)	32		
	4	(22)	25		
		(42)	26	26	
	5	(11)	30		
		(31)	27	27	
	6	(22)	35		
		(32)	35		
		(42)	30	30	
	7	(22)	35		
		(32)	35		
	5	(11)	30	30	
		(31)	27		
	6	(22)	35		
		(32)	35		
	2	(21)	23		
		(31)	26		
		(41)	24	24	
	3	(11)	28		
		(21)	27	27	
		(31)	27	27	

TABLE 2.3. SOLUTION OF THE SAMPLE PROBLEM USING LB III (continued)

No. of Backtrack i	Conflict Level L	Node (jm_{ℓ})	Lower Bounds $B^{L}(jm_{\ell})$	Minimum Lower Bound B^{L}	Schedule Time T_i
	4	(11)	30		
		(31)	27	27	
	5	(22)	33		
		(32)	32		
	4	(11)	30	30	
		(31)	27		
	5	(22)	33		
		(32)	35		
	3	(11)	28		
		(21)	27		
		(31)	27	27	
	4	(11)	30		
		(21)	27	27	
	5	(32)	27		
3		(42)	22	22	27
	2	(21)	23		
		(31)	26	26	
		(41)	24		
	3	(11)	28		
		(21)	32		
		(31)	26	26	
		(41)	33		
	4	(11)	29		
		(21)	32		
		(41)	33		
	1	(13)	24	24	
		(33)	18		
	2	(11)	26		
		(21)	23	23	
		(41)	24		

TABLE 2.3. SOLUTION OF THE SAMPLE PROBLEM USING LB III (continued)

No. of Backtrack i	Conflict Level L	Node (jm_ℓ)	Lower Bounds $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	3	(11)	26		
		(41)	23	23	
	4	(23)	30		
		(43)	24	24	
	5	(11)	33		
		(31)	30		
		(41)	27		
	3	(11)	26	26	
		(41)	23		
	4	(23)	30		
		(33)	24	24	
	5	(31)	30		
		(41)	30		
	2	(11)	26		
		(21)	23		
		(41)	24	24	
	3	(11)	27		
		(21)	24	24	
	4	(11)	35		
		(21)	27		
		(31)	32		
	2	(11)	26	26	
		(21)	23		
		(41)	24		
	3	(21)	32		
		(31)	32		
		(41)	33		

TABLE 2.4 . SOLUTION OF THE SAMPLE PROBLEM USING LB IV

No. of Backtrack i	Conflict Level L	Node (jm_ℓ)	Lower Bounds $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	1	(13)	26		
		(33)	23	23	
	2	(21)	25		
		(31)	26		
		(41)	24	24	
	3	(11)	28		
		(21)	32		
		(31)	27	27	
	4	(11)	30		
		(21)	27	27	
	5	(32)	27	27	
		(42)	27		
	6	(22)	35		
0		(42)	30	30	30
	5	(32)	27		
1		(42)	27	27	27
	2	(21)	25	25	
		(31)	26		
		(41)	24		
	3	(11)	26	26	
		(31)	26		
		(41)	27		
	4	(31)	29		
		(41)	30		
	3	(11)	26		
		(31)	26	26	
		(41)	27		
	4	(11)	29		
		(41)	26	26	

TABLE 2.4. SOLUTION OF THE SAMPLE PROBLEM USING LB IV (continued)

No. of Backtrack i	Conflict Level L	Node (jm_ℓ)	Lower Bounds $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	5	(22)	26	26	
		(32)	26		
	6	(32)	34		
		(42)	27		
	5	(22)	26		
		(32)	26	26	
	6	(22)	34		
		(32)	28		
		(42)	33		
	2	(21)	25		
		(31)	26	26	
		(41)	24		
	3	(11)	28		
		(21)	34		
		(31)	26	26	
		(41)	33		
	4	(11)	29		
		(21)	32		
		(41)	33		
	1	(13)	26	26	
		(33)	23		
	2	(11)	29		
		(21)	26	26	
		(41)	29		
	3	(11)	26	26	
		(41)	26	26	
	4	(23)	30		
		(43)	29		
	3	(11)	26		
		(41)	26	26	

TABLE 2.4. SOLUTION OF THE SAMPLE PROBLEM USING LB IV (continued)

No. of Backtrack i	Conflict Level L	Node (jm_ℓ)	Lower Bounds $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	4	(23)	30		
		(33)	29		

TABLE 2.5. SOLUTION OF THE SAMPLE PROBLEM USING LB V

No. of Backtrack i	Conflict Level L	Node (jm_ℓ)	Lower Bounds $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	1	(13)	27	27	
		(33)	27		
	2	(11)	35		
		(21)	32		
		(41)	27	27	
	3	(11)	27	27	
		(21)	27		
	4	(21)	35		
0		(31)	29	29	32
	3	(11)	27		
		(21)	27	27	
	4	(11)	35		
		(21)	32		
		(31)	29	29	
	5	(11)	30		
		(21)	29	29	
	5	(11)	30	30	
		(21)	29		
	1	(13)	27		
		(33)	27	27	
	2	(21)	31		
		(31)	29		
		(41)	27	27	
	3	(11)	27	27	
		(21)	27		
		(31)	27		
	4	(21)	36		
1		(31)	28	28	31

TABLE 2.5. SOLUTION OF THE SAMPLE PROBLEM USING LB V (continued)

No. of Backtrack i	Conflict Level L	Node (jm_ℓ)	Lower Bounds $B^L(jm_\ell)$	Minimum Lower Bound B^L	Schedule Time T_i
	3	(11)	27		
		(21)	27	27	
		(31)	27		
	4	(11)	33		
		(31)	32		
	3	(11)	27		
		(21)	27		
		(31)	27	27	
	4	(11)	27	27	
2		(21)	27		30
	4	(32)	30		
3		(42)	27	27	27

TABLE 2.6 SCHEDULING TABLE FOR SAMPLE PROBLEM USING COMPOSITE-BASED BOUND LB I.

Back-track i	Conflict Level L	Machine 1				Machine 2				Machine 3				Solu-tion T_i
		(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)	
	1		8		7	4				6*		6*		
	2	9*	8*		7*	4				6		12		
	3	10*	15*		7	4			13	6		12		
	4	10	18*	15*	7	4			13	6		12	15	
0	5	10	23	15	7	4	32	24	13	6	27	12	15	32
	3	10*	15*		7	4			13	6		12		
	4	18	15	15	7	4			13	6		12	15	
	1		8		7	4				6*		6*		
	2		8*	9*	7*	4				8		6		
	3	11*	15*	10*	7	4			13	8		6		
	4	18	15	18	7	4			13	8	19	6	15	
	3	11*	15*	10*	7	4			13	8		6		
	4	13*	18*	10	7	4		19	13	8		6		
	5	21	18	10	7	4		19*	13*	8		6		
1	6	21	18	10	7	4	27	22	13	8	22	6	15	27

* The jobs in the conflict set at a level.

* The job around which the conflict is resolved.

CHAPTER III

COMPUTATIONAL EXPERIMENTS

The branch-and-bound algorithm for job-shop problems, discussed in Section 2.4, has been programmed in FORTRAN IV language. The two composite-based bounds LB I and LB II, developed in this thesis, are imbedded as subroutines. Three other lower bounds, referred to as LB III, LB IV and LB V, are also imbedded as subroutines for comparison purpose. In order to compare the performance of the various bounding procedures, a considerable number of experiments have been conducted on IBM 360/50 computer.

The performance of these lower bounds is compared on the basis of the following factors: (1) the number of nodes explored, (2) the computational time required to obtain the optimal solution; and (3) the efficiency of the solution obtained without backtracking. In addition, various statistics such as the minimum, maximum, mean and standard deviation for all the above three factors are computed.

The sizes of the problems vary between 3 to 12 jobs and 3 to 5 machines. The number of experiments conducted is 18. The number of problems in each experiments is 25. However, due to computation time limitations, in some experiments it has not been possible to solve all the 25 problems. The objective in selecting the problems of above sizes is to investigate the effects of changes in both the number of jobs and the number of machines. The processing times are generated randomly from a uniform distribution with interval 1 and 30, both inclusive. The entries of the machine ordering matrices are also generated randomly.

The results of experiments I through XVII in terms of the number of nodes explored, the computational time required to obtain the optimal solution and the efficiency of solution without backtracking are shown in Tables 3.1 through 3.3. The performance of the various bounding procedures will be evaluated and compared with the help of these results. A number of significant observations, obtained from the analysis of these results for different bounding procedures, are discussed below.

1. Number of Nodes Explored. The number of nodes explored to obtain the optimal solution increases very rapidly as the number of jobs increases. The obvious reason for this rapid increase is: the higher the number of jobs the larger the number of conflicts to be resolved, and consequently, the greater the number of nodes to be explored. However, this factor varies greatly for different bounding procedures for the same experiment. As observed in Table 3.1, in almost all experiments, the number of nodes explored for the composite-based bounds LB I, LB II is relatively very small as compared to other lower bounds LB III, LB IV and LB V. In general, a powerful bounding procedure produces lower bounds as high as possible and recognizes the optimal solution by exploring a small number of nodes. Otherwise, the optimal solution will be reached after a number of backtrackings which, in turn, increase the number of nodes explored. Thus, it is obvious that the composite-based bounds LB I and LB II are more powerful than any of the other lower bounds LB III, LB IV and LB V. Although the results obtained using the composite-based bounds LB I and LB II are fairly close to each other, LB II gives slightly better

results than LB I. This is because the job-based bound LB IV imbedded in composite-based bound LB II is more powerful than the job-based bound LB III imbedded in composite-based bound LB I.

The number of nodes explored to obtain an optimal solution increases as the number of machines increases. However, in some experiments such as II and III for all bounding procedures, and VII and VIII for LB II and LB V as observed in Table 3.1, the number of nodes decreases as the number of machines increases. The decrease in the number of nodes explored also depends on the quality of the bounding procedure. A decrease in the number of nodes may be expected by a reasoning similar to that for the increase in the number of nodes. For different problems of the same size for a particular bounding procedure, the number of nodes explored varies greatly. This variation is due to the random generation of the elements in processing time and machine ordering matrices and the lower bound on the schedule time depends on these elements. Table 3.1 shows another significant observation that the change in the number of nodes explored is due more to the change in the number of jobs than the change in the number of machines. For example, as observed in experiments I, IV and II for LB I in Table 3.1, the mean number of nodes explored changes from 9.32 to 33.32 when the number of jobs changes from 3 to 4, whereas, it changes from 9.32 to 13.44 when the number of machines changes from 3 to 4.

2. Computational Time. Since the computational time depends on the nodes explored, the computational time required to obtain an optimal solution increases rapidly with the increase in the number of jobs. As explained earlier, this is because the increase in the number of

jobs leads to more number of conflicts and consequently, more number of nodes have to be explored. As observed for all experiments except II, III, V and VI in Table 3.2, the computational time required to obtain the optimal solution, using composite-based bounds LB I and LB II, is less than that, using the other lower bounds LB III, LB IV and LB V. For small problems like (3x3) and (4x3), the computational time for composite-based bounds is almost the same as that for some of the other lower bounds, even though the number of nodes explored using the former is less than that using the latter. For example, Table 3.2 shows that, on the average 1.87 seconds are required by LB I and LB IV to explore 9.32 and 10.60 nodes respectively. Thus LB IV spends more computational time in computing a lower bound for each node than that by LB II. However, in all experiments except the above, the composite-based bounds give better results since they recognize the optimal solution by exploring a small number of nodes.

The computational time increases as the number of machines increases. However, in some experiments such as V and VI for LB III and LB IV, the computational time decreases as the number of machines increases. This is because there is a decrease in the number of nodes explored with the increase in the number of machines for the above experiments, as observed in Table 3.1. It is interesting to note that the change in the computational time due to change in the number of machines is relatively more for the composite-based bounds LB I and LB II, and also for the machine-based bound LB V than that for the job-based bounds LB III and LB IV. This is because the increase in the number of machines causes an increase in the number of bounds since the lower bound for each node using any of the former lower bounds is computed as the maximum value of the bounds for all machines.

The variation in the computational time from one problem to another with the same size and for a particular bounding procedure is due to the random generation of the entries in processing time and machine ordering matrices. On the average, the composite-based bounds LB I and LB II take less computational time. Therefore, these bounding procedures are more efficient than any of the job-based or machine-based bounds LB III, LB IV and LB V. However, in general, the composite-based bound LB I gives slightly better results than LB II because the computational time required to compute the lower bound using the former is more than that using the latter.

3. Efficiency of Solution Obtained Without Backtracking. It is interesting to find out how close the solution obtained without backtracking is to the optimal solution. In experiments I through XVIII, the efficiency of such solution is fairly good as observed in Table 3.3. The overall variation in this factor is about 10 percent. Also, unlike the number of nodes explored and the computational time, this factor does not vary much with the increase in the size of the problem. For some experiments such as IV and V, Table 3.3 shows an increase in the efficiency for an increase in the number of machines for all bounding procedures except LB V. Whereas, for some other experiments such as VII and VIII, there is a decrease in the efficiency for an increase in the number of machines for all bounding procedures except LB III. The efficiency of the solution obtained without backtracking depends on the quality of the bounding procedure: the more powerful a bounding procedure the higher the

efficiency of the solution obtained without backtracking. From Table 3.3, it becomes evident that the composite-based bounds LB I and LB II give a higher efficiency than any of the bounding procedures LB III, LB IV and LB V and are, therefore, more powerful. Although, the results obtained using the composite-based bounds LB I and LB II are fairly close to each other, LB II gives slightly better results than LB I. The reason for this slight variation is that the job-based bound LB IV imbedded in composite-based bound LB I is more powerful than the job-based bound LB III imbedded in composite-based bound LB II.

Table 3.1 Mean Number of Nodes Explored to Obtain the Optimal Solution

Exp.	Prob.	LB I		LB II		LB III		LB IV		LB V	
No.	Size	N	NE	N	NE	N	NE	N	NE	N	NE
I	(3x3)	25	9.32	25	8.92	25	13.84	25	10.60	25	13.00
II	(3x4)	25	13.44	25	12.72	25	19.92	25	14.12	25	20.40
III	(3x5)	25	11.28	25	10.52	25	16.60	25	13.40	25	18.68
IV	(4x3)	25	33.32	25	28.44	25	92.00	25	164.47	25	43.68
V	(4x4)	25	49.48	25	46.20	25	99.84	25	80.04	25	79.56
VI	(4x5)	25	58.16	25	50.16	25	94.44	25	56.16	25	141.68
VII	(5x3)	25	196.04	25	202.79	25	1451.76	25	777.52	25	1451.76
VIII	(5x4)	25	266.96	25	188.50	25	1925.43	25	917.24	25	620.56
IX	(5x5)	10	292.90	10	196.00	10	806.20	10	375.80	10	316.22
X	(6x3)	25	385.84	25	366.28	8	1125.35	4	1007.88	25	960.32
XI	(6x4)	24	1210.21	12	1289.50		*		*	5	4961.59
XII	(6x5)	10	1955.40	10	1634.70		*	5	13202.60	2	17829.00
XIII	(8x3)	3	453.35	3	414.33		*		*	3	4200.85
XIV	(8x4)	4	1965.50	4	2211.75		*		*	4	5800.35
XV	(10x3)	4	1242.00	4	1239.75		*		*	4	2314.35
XVI	(10x4)	1	7639.00	1	4781.00		*		*		*
XVII	(12x3)	11	4307.32	3	6202.00		*		*	7	3936.57
XVIII	(12x4)		*		*		*		*		*

N - Number of Problems Solved
NE - Mean Number of Nodes Explored
* - Computer spent 3600 seconds without obtaining the optimal solution

Table 3.2 Mean Computational Time Required to Obtain the Optimal Solution

Exp.	Prob.	LB I		LB II		LB III		LB IV		LB V	
No.	Size	N	CT	N	CT	N	CT	N	CT	N	CT
I	(3x3)	25	1.87	25	2.02	25	2.16	25	1.87	25	2.16
II	(3x4)	25	2.59	25	2.59	25	2.45	25	2.45	25	3.60
III	(3x5)	25	2.73	25	3.02	25	2.59	25	2.74	25	4.18
IV	(4x3)	25	4.18	25	3.89	25	6.76	25	6.34	25	4.90
V	(4x4)	25	6.34	25	7.05	25	6.05	25	7.05	25	10.20
VI	(4x5)	25	9.34	25	9.65	25	5.76	25	6.34	25	22.90
VII	(5x3)	25	17.50	25	20.90	25	54.30	25	47.50	25	33.70
VIII	(5x4)	25	31.70	25	29.65	25	82.00	25	65.50	25	72.00
IX	(5x5)	10	27.40	10	36.00	10	32.40	10	28.15	10	46.50
X	(6x3)	25	33.80	25	42.80	8	378.00	4	823.00	25	94.10
XI	(6x4)	24	148.70	12	193.00		*		*	5	655.00
XII	(6x5)	10	199.50	10	218.50		*	5	942.50	2	423.40
XIII	(8x3)	11	356.50	3	1030.00		*		*	3	1020.00
XIV	(8x4)	4	934.00	4	965.00		*		*	4	800.00
XV	(10x3)	4	678.00	4	838.00		*		*	4	812.00
XVI	(10x4)	1	2880.00	1	2880.00		*		*		*
XVII	(12x3)	11	787.00	3	1620.00		*		*	7	481.00
XVIII	(12x4)		*		*		*		*		*

N - Number of Problems Solved
CT - Mean Computational Time in Seconds
* - Computer spent 3600 seconds without obtaining the optimal solution

Table 3.3 Efficiency of Solution Obtained Without Backtracking

Exp.	Prob.	LB I		LB II		LB III		LB IV		LB V	
No.	Size	N	ES	N	ES	N	ES	N	ES	N	ES
I	(3x3)	25	99.52	25	99.32	25	96.56	25	98.08	25	91.60
II	(3x4)	25	97.84	25	97.88	25	97.72	25	97.60	25	91.92
III	(3x5)	25	98.43	25	99.00	25	97.48	25	97.68	25	93.20
IV	(4x3)	25	96.87	25	97.48	25	95.48	25	93.87	25	93.68
V	(4x4)	25	97.64	25	98.31	25	96.72	25	95.96	25	91.52
VI	(4x5)	25	97.56	25	98.60	25	95.59	25	97.04	25	87.44
VII	(5x3)	25	96.25	25	97.20	25	92.30	25	95.44	25	91.25
VIII	(5x4)	25	93.28	25	92.68	25	92.68	25	92.92	25	82.87
IX	(5x5)	10	95.19	10	93.66	10	93.10	10	94.20	10	94.67
X	(6x3)	25	92.64	25	93.12	8	85.39	4	87.75	25	88.32
XI	(6x4)	24	91.50	12	91.38	1	73.20	1	84.00	5	85.39
XII	(6x5)	10	93.10	10	93.46	1	85.30	5	89.30	2	87.50
XIII	(8x3)	11	89.73	3	90.67	1	83.10	1	92.30	3	86.33
XIV	(8x4)	4	88.75	4	91.50	1	89.20	1	94.50	4	82.00
XV	(10x3)	4	91.00	4	92.75	1	78.80	1	95.00	4	84.50
XVI	(10x4)	1	74.80	1	71.20	1	84.20	1	78.80	1	67.50
XVII	(12x3)	11	97.00	3	95.38	1	89.32	1	92.70	7	94.42
XVIII	(12x4)		*		*		*		*		*

N - Number of Problems Solved
ES - Efficiency of Solution Obtained Without Backtracking
* - The efficiency could not be computed because the optimal solution was not obtained

Table 3.4 Results Obtained by Branch-and-Bound With and Without Backtracking Using LB I*

Exp. No.	Size of Prob.	Branch-and-Bound with Backtracking: Number of Nodes Explored: Max.	Min.	μ	σ	Branch-and-Bound without Backtracking: Efficiency of Solution: Range	μ	σ
I	(3x3)	22	6	9.32	3.85	0.90 - 1.00	99.52	1.98
II	(3x4)	39	4	13.44	7.81	0.89 - 1.00	97.84	3.96
III	(3x5)	42	4	11.28	7.88	0.90 - 1.00	98.44	2.76
IV	(4x3)	92	11	33.32	20.64	0.87 - 1.00	96.87	3.64
V	(4x4)	315	15	49.48	58.12	0.89 - 1.00	97.64	3.48
VI	(4x5)	276	12	58.16	58.54	0.85 - 1.00	97.56	4.33
VII	(5x3)	2224	20	196.04	435.48	0.81 - 1.00	96.25	4.99
VIII	(5x4)	1110	29	266.95	252.25	0.75 - 1.00	93.28	7.13
IX	(5x5)	1438	83	292.9	383.94	0.87 - 1.00	95.19	3.00
X	(6x3)	3487	42	385.84	709.17	0.76 - 1.00	92.65	7.65
XI	(6x4)	6160	46	1210.21	1449.33	0.75 - 1.00	91.58	7.10
XII	(6x5)	4907	195	1955.40	1400.47	0.86 - 1.00	93.10	4.44
XIII	(8x3)	21322	103	3171.55	6022.66	0.76 - 1.00	89.73	7.10
XIV	(8x4)	4526	570	1965.50	1610.93	0.83 - 0.98	88.75	5.58
XV	(10x3)	2602	279	1242.00	927.95	0.77 - 0.98	91.00	8.22
XVII	(12x3)	7781	537	4307.82	3260.01	0.86 - 1.00	97.00	5.00

*The results for the remaining experiments are not tabulated as only one or no optimal solution was obtained

μ - Mean

σ - Standard deviation

Table 3.5 Results Obtained by Branch-and-Bound With and Without Backtracking Using LB II*

Exp. No.	Size of Prob.	Branch-and-Bound with Backtracking Number of Nodes Explored Max.	Min.	μ	σ	Branch-and-Bound without Backtracking Efficiency of Solution Range	μ	σ
I	(3x3)	22	5	8.92	3.95	0.90 - 1.00	99.32	2.34
II	(3x4)	33	4	12.72	6.53	0.90 - 1.00	97.88	3.55
III	(3x5)	42	4	10.52	7.51	0.90 - 1.00	99.00	2.50
IV	(4x3)	72	11	28.44	16.14	0.82 - 1.00	97.48	4.68
V	(4x4)	305	13	46.2	57.41	0.88 - 1.00	98.31	3.09
VI	(4x5)	252	12	50.16	53.49	0.85 - 1.00	98.60	3.37
VII	(5x3)	1939	20	97.20	396.95	0.81 - 1.00	97.20	4.75
VIII	(5x4)	434	46	188.50	116.74	0.75 - 1.00	92.69	6.37
IX	(5x5)	413	98	196.00	94.76	0.79 - 1.00	93.67	6.43
X	(6x3)	2809	41	360.28	628.75	0.79 - 1.00	93.12	8.10
XI	(6x4)	2397	50	1098.00	775.33	0.75 - 1.00	92.00	7.00
XII	(6x5)	4561	160	1634.7	1359.36	0.86 - 1.00	93.46	4.31
XIII	(8x3)	506	319	414.33	76.39	0.82 - 1.00	90.66	6.94
XIV	(8x4)	4526	509	2211.75	1541.51	0.84 - 1.00	91.00	7.00
XV	(10x3)	2293	283	1239.75	863.56	0.84 - 1.00	92.75	5.26
XVI	(10x4)	4781	2635	4781	0.00			
XVII	(12x3)	10809	2169	6202	3550.54	0.97 - 1.00	98.00	2.00

*The results for the remaining experiments are not tabulated as only one or no optimal solution was obtained

μ - Mean

σ - Standard deviation

Table 3.6 Results Obtained by Branch-and-Bound With and Without Backtracking, Using LB III*

Exp. No.	Size of Prob.	Branch-and-Bound with Backtracking Number of Nodes Explored Max.	Min.	μ	σ	Branch-and-Bound without Backtracking Efficiency of Solution Range	μ	σ
I	(3x3)	33	6	13.84	7.06	0.83 - 1.00	96.56	5.43
II	(3x4)	69	4	19.92	17.33	0.85 - 1.00	97.72	4.28
III	(3x5)	52	4	16.60	13.45	0.81 - 1.00	97.48	4.75
IV	(4x3)	391	15	92	93.21	0.79 - 1.00	95.48	5.45
V	(4x4)	437	20	99.84	85.54	0.85 - 1.00	96.72	3.94
VI	(4x5)	336	12	94.44	90.45	0.86 - 1.00	95.56	4.94
VII	(5x3)	52	4	16.60	13.45	0.81 - 1.00	97.48	4.75
VIII	(5x4)	7645	246	1925.40	2064.40	0.82 - 1.00	92.48	5.17
IX	(5x5)	3360	120	806.20	894.19	0.86 - 1.00	93.10	4.11
X	(6x3)	2809	41	366.28	628.75	0.79 - 1.00	93.12	8.10

*The results for the remaining experiments are not tabulated as only one or no optimal solution was obtained

μ - Mean

σ - Standard deviation

Table 3.7 Results Obtained by Branch-and-Bound With and Without Backtracking Using LB IV*

Exp. No.	Size of Prob.	Branch-and-Bound with Backtracking Number of Nodes Explored Max.	Min.	μ	σ	Branch-and-Bound without Backtracking Efficiency of Solution Range	μ	σ
I	(3x3)	27	6	10.6	5.37	0.90 - 1.00	98.08	3.42
II	(3x4)	39	4	14.12	9.16	0.83 - 1.00	97.60	4.35
III	(3x5)	44	4	13.4	10.18	0.81 - 1.00	97.68	4.56
IV	(4x3)	603	28	164.47	151.96	0.78 - 1.00	93.87	5.06
V	(4x4)	372	16	80.04	92.65	0.79 - 1.00	95.96	5.81
VI	(4x5)	252	12	56.16	51.74	0.86 - 1.00	97.04	4.42
VII	(5x3)	3077	97	777.52	704.77	0.83 - 1.00	95.44	4.83
VIII	(5x4)	4599	137	917.24	1109.26	0.81 - 1.00	92.2	5.35
IX	(5x5)	3360	120	806.20	894.19	0.86 - 1.00	93.10	4.11
X	(6x3)	28116	1807	10078.75	10532.70	0.74 - 0.98	87.75	9.23

*The results for the remaining experiments are not tabulated as only one or no optimal solution was obtained

μ - Mean

σ - Standard deviation

Table 3.8 Results Obtained by Branch-and-Bound With and Without Backtracking Using LB V*

Exp. No.	Size of Prob.	Branch-and-Bound with Backtracking: Number of Nodes Explored				Branch-and-Bound without Backtracking: Efficiency of Solution		
		Max.	Min.	μ	σ	Range	μ	σ
I	(3x3)	35	6	13.0	6.68	0.80 - 1.00	91.60	7.09
II	(3x4)	60	4	20.4	12.00	0.63 - 1.00	91.92	8.99
III	(3x5)	62	4	18.68	14.81	0.79 - 1.00	93.20	7.41
IV	(4x3)	118	11	43.68	30.15	0.82 - 1.00	93.68	5.96
V	(4x4)	525	15	79.56	96.08	0.67 - 1.00	91.52	9.81
VI	(4x5)	727	28	141.68	154.97	0.63 - 1.00	87.44	9.68
VII	(5x3)	2696	20	303.96	542.41	0.68 - 1.00	91.25	8.82
VIII	(5x4)	2661	50	620.56	612.72	0.68 - 1.00	82.88	7.66
IX	(5x5)	1438	137	316.22	397.93	0.67 - 1.00	94.67	3.89
X	(6x3)	8343	42	960.32	1947.24	0.75 - 1.00	88.32	7.26
XI	(6x4)	8773	2342	4961.60	2241.25	0.77 - 0.96	85.40	6.47
XII	(6x5)	23643	12015	17829.0	5814.0	0.93 - 0.94	94.20	4.00
XIII	(8x3)	916	637	844.0	146.73	0.82 - 0.91	86.33	3.68
XIV	(8x4)	12008	771	5227.0	4166.13	0.76 - 0.87	82.00	3.94
XV	(10x3)	3371	264	2010.25	1182.54	0.77 - 0.89	84.50	4.56
XVII	(12x3)	11278	567	3936.57	3401.70	0.86 - 1.00	90.43	5.29

*The results for the remaining experiments are not tabulated as only one or no optimal solution was obtained

μ - Mean

σ - Standard deviation

CHAPTER IV

SUMMARY AND CONCLUSIONS

The basic objective of this thesis is to develop a branch-and-bound algorithm for job-shop problems. The branch-and-bound approach generates an optimal solution after the generation of only a small subset of possible sequences. The basic concepts of this approach which consists of the branching, bounding and backtracking processes are discussed, using a scheduling tree. The process of generating a new set of nodes at a level from a node at the preceeding level is referred to as the branching process. This process guarantees an optimal solution by generating all nodes of the scheduling tree. The bounding process helps select a particular node at a level for further branching and thus makes it possible to achieve a reduction in the generation of nodes at each level. A backtracking process has to be embedded in the branch-and-bound technique to guarantee optimality. The efficiency of the branch-and-bound technique depends on the quality of the bounding procedure.

A mathematical analysis, in rigorous notation, of the five bounding procedures is presented. The composite-based bounds, referred to as LB I and LB II, are developed in this thesis. The other three bounding procedures, referred to as LB III, LB IV and LB V, are analyzed for comparison purposes. The computation of the lower bounds, using these bounding procedures, is illustrated with the help of a sample problem. The computational algorithm for the branch-and-bound technique is summarized in formal steps. The sample problem, presented earlier, is solved to illustrate the computational algorithm.

In order to study the performance of the various bounding procedures, a considerable number of experiments has been conducted on IBM 360/50. The sizes of the problems vary between 3 to 12 jobs and 3 to 5 machines. The total number of experiments conducted is 18 and the number of problems in each experiment is 25. The elements in both processing time and machine ordering matrices are generated randomly. The performance of the various bounding procedures is compared on the basis of the number of nodes explored, the computation time and the efficiency of solution without backtracking. Also, various statistics such as the minimum, maximum, mean and standard deviation for all the above three factors are computed.

The most significant results obtained from the computational experiments are as follows:

1. The number of nodes explored increases with the increase in the size of the problem. This is because the increase in the size of the problem leads to more number of conflicts.

2. The computational time required to obtain the optimal solution depends on the number of nodes explored. For composite-based bounds LB I and LB II and machine-based bound LB V, the computational time to explore a node using the machine-based bound depends on the number of machines because the lower bound for each node is computed as the maximum value of the bounds for all machines.

3. Unlike the number of nodes explored and the computational time required to obtain the optimal solution, the efficiency of solution obtained without backtracking does not vary much with the increase in the size of the problem. However, it depends on the quality of the bounding procedure: the more powerful a bounding procedure the higher the

efficiency of the solution obtained without backtracking. It is observed that the composite-based bounds LB I and LB II, on the average, give a higher efficiency than any of the bounding procedures LB III, LB IV and LB V.

4. The composite-based bounds LB I and LB II are, on the average, more powerful in terms of the number of nodes explored and the computational time required to obtain the optimal solution and the efficiency of the solution obtained without backtracking than any of the lower bounds LB III, LB IV and LB V. The results obtained using the composite-based bounds LB I and LB II are fairly close to each other. However, the composite-based bound LB I gives slightly better results in terms of the number of nodes explored and the efficiency of the solution obtained without backtracking. This is because the job-based bound LB IV, embedded in composite-based bound LB II, is more powerful than the job-based bound LB III, embedded in the composite-based bound LB I. The composite-based bound LB I gives better results in terms of the computer time required to obtain the optimal solution than LB II because the computational time required to explore a node using the former is less than that using the latter.

5. In ranking the five bounding procedures, as shown in Tables 4.1, 4.2 and 4.3, it appears, on the average, that the composite-based bound LB I ranks first from the point of view of computational time. Whereas, the composite-based bound LB II ranks first according to the number of nodes explored to obtain the optimal solution, and the efficiency of solution obtained without backtracking.

In conclusion, the composite-based bound LB I, which consists of the job-based bound LB III and the machine-based bound LB V, is recommended as the powerful lower bound.

Table 4.1 Rank of Bounding Procedures Based on the Number of Nodes Explored.

Exp. No.	Size of Problem	RANK 1	2	3	4	5
I	(3x3)	LB II	LB I	LB IV	LB V	LB III
II	(3x4)	LB II	LB I	LB IV	LB III	LB V
III	(3x5)	LB II	LB I	LB IV	LB III	LB V
IV	(4x3)	LB II	LB I	LB V	LB III	LB IV
V	(4x4)	LB II	LB I	LB V	LB IV	LB III
VI	(4x5)	LB II	LB IV	LB I	LB III	LB V
VII	(5x3)	LB I	LB II	LB IV	LB III,V	–
VIII	(5x4)	LB II	LB I	LB V	LB IV	LB III
IX	(5x5)	LB II	LB I	LB V	LB IV	LB III
X	(6x3)	LB II	LB I	LB V	LB IV	LB III
XI	(6x4)	LB I	LB II	LB V	*	*
XII	(6x5)	LB II	LB I	LB V	LB IV	*
XIII	(8x3)	LB II	LB I	LB V	*	*
XIV	(8x4)	LB I	LB II	LB V	*	*
XV	(10x3)	LB II	LB I	LB V	*	*
XVI	(10x4)	LB II	LB I	LB V	*	*
XVII	(12x3)	LB I	LB II	LB V	*	*

*Due to Computer time limitations, the optimal solution was not obtained for the remaining bounding procedures.

Table 4.2 Rank of Bounding Procedures Based on Computational Time

Exp. No.	Size of Problem	RANK 1	2	3	4	5
I	(3x3)	LB I,IV	LB II	LB III,V	—	—
II	(3x4)	LB III,IV	LB I,II	LB V	—	—
III	(3x5)	LB III	LB I	LB IV	LB II	LB V
IV	(4x3)	LB II	LB I	LB V	LB IV	LB III
V	(4x4)	LB III	LB I	LB II,IV	LB V	—
VI	(4x5)	LB III	LB IV	LB I	LB II	LB V
VII	(5x3)	LB I	LB II	LB V	LB IV	LB III
VIII	(5x4)	LB II	LB I	LB IV	LB V	LB III
IX	(5x5)	LB I	LB IV	LB III	LB II	LB V
X	(6x3)	LB I	LB II	LB V	LB III	LB IV
XI	(6x4)	LB I	LB II	LB V	*	*
XII	(6x5)	LB I	LB II	LB V	LB IV	*
XIII	(8x3)	LB I	LB V	LB II	*	*
XIV	(8x4)	LB V	LB I	LB II	*	*
XV	(10x3)	LB I	LB V	LB II	*	*
XVI	(10x4)	LB I,II	LB V	*	*	*
XVII	(12x3)	LB V	LB I	LB II	*	*

*Due to Computer time limitations, the optimal solution was not obtained for the remaining bounding procedures.

Table 4.3 Rank of Bounding Procedures Based on Efficiency of Solution (Without Backtracking)

Exp. No.	Size of Problem	RANK 1	2	3	4	5
I	(3x3)	LB I	LB II	LB IV	LB III	LB V
II	(3x4)	LB II	LB I	LB III	LB IV	LB V
III	(3x5)	LB II	LB I	LB IV	LB III	LB V
IV	(4x3)	LB II	LB I	LB III	LB IV	LB V
V	(4x4)	LB II	LB I	LB III	LB IV	LB V
VI	(4x5)	LB II	LB I	LB IV	LB III	LB V
VII	(5x3)	LB II	LB I	LB IV	LB III	LB V
VIII	(5x4)	LB I	LB IV	LB II	LB III	LB V
IX	(5x5)	LB I	LB V	LB IV	LB II	LB III
X	(6x3)	LB II	LB I	LB V	LB IV	LB III
XI	(6x4)	LB I	LB II	LB V	LB IV	LB III
XII	(6x5)	LB II	LB I	LB IV	LB V	LB III
XIII	(8x3)	LB IV	LB II	LB I	LB V	LB III
XIV	(8x4)	LB IV	LB II	LB III	LB I	LB V
XV	(10x3)	LB IV	LB II	LB I	LB V	LB III
XVI	(10x4)	LB III	LB IV	LB I	LB II	LB V
XVII	(12x3)	LB I	LB II	LB IV	LB V	LB III

BIBLIOGRAPHY

Branch-and-Bound Approach

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

2. Ashour, S. and M. N. Quraishi, "Investigation of Various Bounding Procedures for Production Scheduling Problems", The International Journal of Production Research, Vol. 7, No. 3, 1969, pp. 1 - 4.

3. Ashour, S., "An Experimental Investigation and Comparative Evaluation of Flow-shop Scheduling Techniques", submitted to Operations Research, 1969.

4. Brooks, G. H. and C. White, "An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem", The Journal of Industrial Engineering, Vol. 16, Jan. 1965, pp. 34 - 40.

5. 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.

6. Eastman, W. L., "Linear Programming with Pattern Constraints", Ph.D. Dissertation, Harvard University, Cambridge, Mass., July 1958.

7. 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.

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

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

10. 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.

11. 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.

12. 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.

13. 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.

14. White, C., "An Algorithm for Finding Optimal or Near Optimal Solutions to the Production Scheduling Problem," Ph.D. Thesis, Purdue University, Lafayette, Indiana, 1963.

Related Approaches

15. Ashour, S., "A Decomposition Approach for the Machine Scheduling Problem", The International Journal of Production Research, Vol. 6, No. 2, 1967, pp. 109 - 122.

16. Ashour, S., Introduction to Scheduling: Concepts, Analyses, and Performances, John Wiley & Sons, Inc., in press.

17. Beenhakker, H. L., "Mathematical Analysis of Facility-Commodity Scheduling Problems", The International Journal of Production Research, Vol. 2, No. 4, 1963, pp. 313 - 321.

18. Beenhakker, H. L., "The Development of Alternatie Criteria for Optimality in the Machine Sequenceing Problem", Ph.D. Thesis, Purdue University, Lafayette, Indiana, 1963.

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

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

21. Fisher, H., and G. L. Thompson, "Probabilistic Learning Combinations of Local Job-Shop Scheduling Rules," Chapter 15 in reference 31.

22. Giffler, B. and Thompson, G., "Algorithms for Solving Production Scheduling Problem", IBM Research Report RC-118, Yorktown Heights, New York, June 1959.

23. _______ and _______, "Algorithms for Solving Production Scheduling Problems", Operations Research, Vol. 8, July-Aug. 1960, pp. 487 - 503.

24. _______, _______ and Van Ness, V., "Numerical Experience with the Linear and Monte Carlo Algorithms for solving Production Scheduling Problems", Chapter 3 in reference 31.

25. 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.

26. Heller, J., "Combinatorial, Probabilistic and Statistical Aspects of an M x J Scheduling Problem", Report NYO-2540, Atomic Energy Commission Computed and Applied Mathematics Center, Institute of Mathematical Science, New York University, New York, Feb. 1959.

27. _______, "Combinatorial Properties of Machine Shop Scheduling", Report NYO-2879, Atomic Energy Commission Computed and Applied Mathematics Center, Institute of Mathematical Sciences, New York University, New York, July 1959.

28. _______, "Some Problems in Linear Graph Theory that Arise in the Analysis of the Sequencing of Jobs through Machines", Report NYO-9847, Atomic Energy Commission Computed and Applied Mathematics Center, Institute of Mathematical Science, New York University, New York, Oct. 1960.

29. _______ and Logemann, G., "An Algorithm for the construction and Evaluation of Possible Schedules", Management Science, Vol. 8, Jan. 1962, pp. 168 - 183.

30. 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.

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

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

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

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

APPENDIX A

THE
FOLLOWING
DOCUMENT HAS
PRINTING THAT
EXTENDS INTO
THE BINDING.

THIS IS AS
RECEIVED FROM
CUSTOMER.

ILLEGIBLE DOCUMENT

THE FOLLOWING DOCUMENT(S) IS OF POOR LEGIBILITY IN THE ORIGINAL

THIS IS THE BEST COPY AVAILABLE

```
C
C      ***   BRANCH-AND-BOUND ALGORITHM   ***
C
C      ***       FOR JOB-SHOP PROBLEMS     ***
C
C      PROGRAMMED BY
C                     S. R. HIREMATH
C
C.....................................................................
C
C.....................................................................
C      THE BRANCH AND BOUND ALGORITHM DESCRIBED IN SECTION 2.4
C      IS PROGRAMMED IN FORTRAN IV
C      THIS PROGRAM CONSISTS OF MAIN PROGAM AND FIVE BOUNDING
C      PROCEDURES AS SUBROUTINES. IN ADDISON IT ALSO CONSISTS
C      OF THREE MORE SUBROUTINES.
C
C.....................................................................
C
C.....................................................................
C
C      *****      VARIABLES     *****
C
C      IT             PROCESSING TIME
C      MM             MACHINE ORDERING
C      JCT            COMPLETION TIME
C      MACH           TOTAL NUMBER OF MACHINES OR OPERATIONS FOR
C      JOBS           TOTAL NO. OF JOBS
C                     A JOB
C      LA             ENTRY IN THE SCHEDULING TABLE
C      IOP            OPERATION
C      JJ             JOB IN THE CONFLICT SET
C      N              NO. OF JOBS IN CONFLICT SET AT A LEVEL
C      ILB            LOWER BOUND FOR A NODE
C      NILB           MIN. LOWER BOUND AT A LEVEL
C      ISTMIN         SCHEDULE TIME
C      JACTIV         ACTIVE NODE AT A LEVEL
C
C.....................................................................
C
C
C      IREAD.EQ.O      GENERATING DATA (MACHINE-ORDERING AND
C                      PROCESSING TIME MATRICES)
C      IREAD.NE.O      READ DATA CARDS FOR  BOTH MATRICES
C
C      IF IPRINT.EQ.O   PRINT DETAILS
C      IF IPRINT.NE.O   DO NOT PRINT DETAILS
C
C      IF ICARD.EQ.O    NO CARD OUTPUT DESIRED
C-     IF ICARD.NE.O    CARD OUTPUT DESIRED
C
C      LIMIT1&LIMIT2    THE LIMITS OF INTERVAL FOR PROCESS-
C                       ING TIMES
C
C.....................................................................
C
C       MAIN PROGRAM
C
```

```
                 COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
                 COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
                 COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
                 DIMENSION IRAND2(50)
                 READ(1,1) MACH,JOBS,LIMIT1,LIMIT2,NPROB,NFLB,NLLB,IREAD,ISKIP,
                1IPRINT,ICARD,IX,IY,IB
               1 FORMAT(11I4,2I8,I4)
                 DO 32 NP=1,NPROB
                 WRITE (3,33) NP
              33 FORMAT(1H0,10X,' PROBLEM NUMBER = ',I3)
                 IF(IREAD.EQ.0) GO TO 600
C
C                READ PROCESSING TIME MATRIX
C
                 DO 9 J=1,JOBS
               9 READ(1,2) (IT(J,I),I=1,MACH)
               2 FORMAT(10I5)
                 GO TO 77
C
C                GENERATE PROCESSING TIME MATRIX
C
             600 DO 211 M=1,MACH
                 DO 211 J=1,JOBS
             211 IT(J,M)=RANDNO(IY)*(LIMIT2-LIMIT1+1)+LIMIT1
              77 WRITE (3,82)
              82 FORMAT(1H ,10X,         'PROCESSING TIME MATRIX')
                 DO 11 J=1,JOBS
              11 WRITE (3,4) (IT(J,I),I=1,MACH)
               4 FORMAT(1H ,10X,12I4)
                 IF(IREAD.EQ.0) GO TO 698
C
C                READ MACHINE-ORDERING MATRIX
C
                 DO 3 J=1,JOBS
               3 READ(1,2) (MM(J,I),I=1,MACH)
                 GO TO 76
C
C                GENERATE MACHINE ORDERING MATRIX
C
             698 DO 235 J=1,JOBS
                 DO 231 M=1,MACH
             231 IRAND2(M)=M
                 M1=MACH
             232 IRAN=RANDNO(IY)*M1+1
                 MM(J,M1)=IRAND2(IRAN)+100*J
                 IF (IRAN .EQ. M1) GO TO 234
                 M1=M1-1
                 IF (M1 .EQ. 0) GO TO 235
                 DO 233 M2=IRAN,M1
             233 IRAND2(M2)=IRAND2(M2+1)
                 GO TO 232
             234 IF (M1 .EQ. 1) GO TO 235
                 M1=M1-1
                 GO TO 232
             235 CONTINUE
              76 DO 96 J=1,JOBS
                 DO 96 I=1,MACH
              96 MM(J,I)=MM(J,I)-J*100
```

```
               WRITE (3,81)
            81 FORMAT(1H ,10X,        'MACHINE ORDERING MATRIX')
               DO 8 J=1,JOBS
             8 WRITE (3,4) (MM(J,I),I=1,MACH)
               WRITE (3,35) IB
            35 FORMAT(1H0,10X,' BOUNDING PROCEDURE',I3)
      C
      C        FORM COMPLETION TIME MATRIX
      C
               DO 10 J=1,JOBS
               JCT(J,1)=IT(J,1)
               DO 10 I=2,MACH
               JCT(J,I)=JCT(J,I-1)+IT(J,I)
            10 CONTINUE
               IF(IPRINT.EQ.0) GO TO 21
               WRITE (3,83)
            83 FORMAT(1H ,10X,        'COMPLETION TIME MATRIX')
               DO 13 J=1,JOBS
            13 WRITE (3,2) (JCT(J,I),I=1,MACH)
      C
      C        INITIALIZE
      C        SET UP SCHEDULING TABLE
      C
            21 DO 40 LV=1,90
               DO 40 I=1,MACH
               DO 40 J=1,JOBS
            40 LA(LV,I,J)=0
               DO 41 LV=1,90
               DO 41 J=1,JOBS
            41 IOP(LV,J)=1
      C
      C        ENTER FIRST OPERATIONS OF EACH JOB
      C
               T=0.
               ISWTCH=0
               ISTMIN=99999
               LV=1
               NBKTRK=0
               NNODES=0
               NCNFLT=0
               CALL TIME(NT1)
               DO 50 J=1,JOBS
               M=MM(J,1)
            50 LA(1,M,J)=JCT(J,1)
      C
      C        FIND SMALLEST T AND NEXT HIGHER T
      C
            51 CALL SMALLT(T,LV)
               IF(IPRINT.EQ.0) GO TO 22
               WRITE (3,84)
            84 FORMAT(1H ,10X,        'SCHEDULING TABLE')
               WRITE (3,7) ((LA(LV,K,J),J=1,JOBS),K=1,MACH)
             7 FORMAT(1H ,10X,30I4)
      C
      C        CHECK FOR CONFLICT
      C
           551 WRITE (3,85)
            85 FORMAT(1H ,10X,' JOB MC    LV  JCT')
```

```
   22 DO 70 K=1,MACH
      DO 72 J=1,JOBS
      IF(LA(LV,K,J).NE.T) GO TO 72
      N(LV)=0
      DO 69 JM=1,JOBS
   99 IF(LA(LV,K,JM).GE.T) GO TO 68
      GO TO 69
   68 N(LV)=N(LV)+1
      JJ(LV,N(LV))=JM
      IF(IPRINT.EQ.0) GO TO 69
      WRITE (3,65) JJ(LV,N(LV)),K,LV,LA(LV,K,JM)
   65 FORMAT(1H ,10X,4I4)
   69 CONTINUE
C
C     DETERMINE LOWER BOUNDS AND RESOLVE CONFLICT
C
      IF(N(LV).GT.1) CALL CONFLT(T,K,LV,NNODES,NCNFLT,&110)
      GO TO 70
   72 CONTINUE
   70 CONTINUE
C
C     UPDATE THE ARRAY AND ENTER NEXT OPERATION
C
   89 DO 80 K=1,MACH
      DO 80 J=1,JOBS
      IF(LA(LV,K,J).NE.T) GO TO 80
      IOP(LV,J)=IOP(LV,J)+1
      IF(IOP(LV,J).GT.MACH) GO TO 75
      KK=MM(J,IOP(LV,J))
      LA(LV,KK,J)=JCT(J,IOP(LV,J))
      GO TO 80
   75 IOP(LV,J)=IOP(LV,J)-1
   80 CONTINUE
C
C     CHECK FOR T
C     IF T IS THE HIGHEST ENTRY A SOLUTION HAS BEEN FOUND
C     OTHERWISE FIND NEXT HIGHER T
C
   79 DO 16 M=1,MACH
      DO 16 J=1,JOBS
      IF(T.LT.LA(LV,M,J)) GO TO 51
   16 CONTINUE
      IF(T.GE.ISTMIN) GO TO 110
      ISTMIN=T
      WRITE (3,6) ISTMIN
    6 FORMAT(1H ,' A SOLUTION  ',I6)
  980 FORMAT(1H0,10X,' COMPUTATION TIME =',F12.4)
      LEVEL=LV-1
      WRITE(3,1001) LEVEL
 1001 FORMAT(1H ,' NO OF CONFLICT LEVELS FOR SOLN',I6)
      ISWTCH=ISWTCH+1
      NBKTRK=NBKTRK+1
      IF(ISWTCH.EQ.1) NBKTRK=0
      IF(ISWTCH.NE.1) GO TO 110
      IF(ICARD.EQ.0) GO TO 110
      WRITE (2,1003) ISTMIN
 1003 FORMAT(I8)
C
```

```
C     BACKTRACKING
C
  110 DO 95 I=1,MACH
      DO 95 J=1,JOBS
   95 LA(LV,MM(J,I),J)=0
      LV=LV-1
      IF(IPRINT.EQ.0) GO TO 23
      WRITE (3,501) LV
  501 FORMAT(1H ,' LEVEL',I5)
   23 NLV=N(LV)
  120 DO 300 NL=1,NLV
      IF(JJ(LV,NL).NE.JACTIV(LV)) GO TO 300
      NK=NL
  300 CONTINUE
      DO 360 NL=1,NLV
      IF(JJ(LV,NL).EQ.JACTIV(LV)) GO TO 360
      IF(ISTMIN.LE.ILB(LV,NL)) GO TO 360
      IF(ILB(LV,NL).LT.NILB(LV)) GO TO 360
      IF(ILB(LV,NL).GT.NILB(LV)) GO TO 310
      IF(NL.GT.NK) GO TO 310
  360 CONTINUE
      IF(LV-1)400,400,110
  310 JL=JJ(LV,1)
      IL=IOP(LV,JL)
      ML=MM(JL,IL)
      T=LA(LV,ML,JL)
      DO 350 NL=2,NLV
      J2=JJ(LV,NL)
      I2=IOP(LV,J2)
      M2=MM(J2,I2)
      IF(LA(LV,M2,J2).LT.T) T=LA(LV,M2,J2)
  350 CONTINUE
      IF(IPRINT.EQ.0) GO TO 24
      WRITE (3,48) T
   48 FORMAT(1H ,' T*',F8.1)
   24 DO 583 J=1,JOBS
      DO 580 I=1,MACH
      M=MM(J,I)
      IF(LA(LV,M,J).EQ.0) GO TO 585
      JCT(J,I)=LA(LV,M,J)
      GO TO 580
  585 JCT(J,I)=JCT(J,I-1)+IT(J,I)
  580 CONTINUE
  583 CONTINUE
      CALL SMLILB(LV)
      GO TO 22
  400 WRITE (3,509) ISTMIN
  509 FORMAT(1H0,10X,' OPTIMAL SCHEDULE TIME = ',I6)
      WRITE (3,73) NNODES
      WRITE (3,74) NCNFLT
      WRITE (3,78) NBKTRK
      CALL TIME(NT2)
      COTIME=(NT2-NT1)/100.
      WRITE(3,980) COTIME
      IF(ICARD.EQ.0) GO TO 32
      WRITE(2,902) NNODES,NCNFLT,NBKTRK,COTIME,ISTMIN
   78 FORMAT(1H0,10X,' NUMBER OF BACKTRACKS =',I12)
   73 FORMAT(1H0,10X,' NUMBER OF NODES EXPLORED =',I12)
```

```
      74 FORMAT(1H0,10X,' NUMBER OF CONFLTS =',I12)
     902 FORMAT(3I12,F11.3,I10)
      32 CONTINUE
     100 STOP
         END
```

```
      FUNCTION RANDNO(IY)
      IY=IY*65627
      IF(IY)5,6,6
    5 IY=IY+2147483647+1
    6 RANDNO=IY*.4656613E-9
      RETURN
      END
```

```
      SUBROUTINE SMALLT(T,LV)
C
C.......................................................................
C
C     THIS SUBROUTINE FINDS THE SMALLEST NUMBER IN THE SCHE-
C     DULING TABLE AT LEVEL 1 IN THE BEGINNING. EVERYTIME,
C     IT UPDATES THE VALUE OF T TO THE NEXT HIGHER VALUE IN
C     THE SCHEDULING TABLE AT A LEVEL LV.
C
C.......................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      ISWTCH=0
      DO 20 M=1,MACH
      DO 20 J=1,JOBS
      IF(LA(LV,M,J).LE.T) GO TO 20
      ISWTCH=ISWTCH+1
      IF(ISWTCH.EQ.1) TS=LA(LV,M,J)
   10 IF(LA(LV,M,J).LT.TS) TS=LA(LV,M,J)
   20 CONTINUE
      T=TS
      IF(IPRINT.EQ.0) GO TO 25
      WRITE (3,502) T
  502 FORMAT(1H ,' T  ',F8.1)
   25 RETURN
      END
```

```
      SUBROUTINE CONFLT (T,K,LV,NNODES,NCNFLT,*)
C
C.....................................................................
C
C     THIS SUBROUTINE CHECKS FOR CONFLICT. IF A CONFLICT
C     EXISTS, THE LOWER BOUNDS FOR THE NODES IN THE CONFLICT
C     SET ARE COMPUTED USING ONE OF THE LOWER BOUNDS.
C
C.....................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      NLV=N(LV)
      L=0
C
C     CHECKING FOR CONFLICT USING BEENHAKKERS FORMULA.
C
      DO 40 KL=1,NLV
      J2=JJ(LV,KL)
      I2=IOP(LV,J2)
      M2=MM(J2,I2)
      IF(LA(LV,M2,J2).EQ.T) GO TO 39
      IF((T+IT(J2,I2)).LE.JCT(J2,I2)) GO TO 40
   39 L=L+1
   40 CONTINUE
      IF(L.LE.1) GO TO 35 ← Return
C
C     IF CONFLICT EXISTS, ONE OF THE BOUND SUBROUTINE IS
C     CALLED TO COMPUTE THE LOWER BOUNDS FOR THE NODES IN
C     THE CONFLICT SET AT A LEVEL.
C
      GO TO(91,92,93,94,95),IB
   91 CALL BOUND1(T,K,LV)
      GO TO 96
   92 CALL BOUND2(T,K,LV)
      GO TO 96
   93 CALL BOUND3(T,K,LV)
      GO TO 96
   94 CALL BOUND4(T,K,LV)
      GO TO 96
   95 CALL BOUND5(T,K,LV)
   96 NILB(LV)=ILB(LV,1)
C
C     DETERMINE THE NODE WITH MINIMUM LOWER BOUND. IF A TIE
C     EXISTS, IT IS BROKEN USING THE LEFT HAND RULE.
C
      NNODES=NNODES+N(LV)
      JACTIV(LV)=JJ(LV,1)
      DO 10 NL=2,NLV
      IF(NILB(LV).LE.ILB(LV,NL)) GO TO 10
      NILB(LV)=ILB(LV,NL)
      JACTIV(LV)=JJ(LV,NL)
   10 CONTINUE
      IF(NILB(LV)-ISTMIN)21,22,22
   21 LV=LV+1
      NCNFLT=NCNFLT+1
      DO 20 M=1,MACH
```

```
      DO 20 J=1,JOBS
   20 LA(LV,M,J)=LA(LV-1,M,J)
      DO 30 J=1,JOBS
   30 IOP(LV,J)=IOP(LV-1,J)
      IA=IOP(LV,JACTIV(LV-1))
      L=N(LV-1)
      DO 15 NL=1,L
      IF(JJ(LV-1,NL).EQ.JACTIV(LV-1)) GO TO 15
      J1=JJ(LV-1,NL)
C
C     UPDATE THE COMPLETION TIME MATRIX IN FAVOR OF THE NODE
C
      I1=IOP(LV,J1)
      JCT(J1,I1)=JCT(JACTIV(LV-1),IA)+IT(J1,I1)
      M=MM(J1,I1)
      LA(LV,M,J1)=JCT(J1,I1)
      IK=I1+1
      IF(IK.GT.MACH) GO TO 15
      DO 14 IC=IK,MACH
   14 JCT(J1,IC)=JCT(J1,IC-1)+IT(J1,IC)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 35
      WRITE (3,83)
   83 FORMAT(1H0,10X,        'COMPLETION TIME MATRIX*')
      DO 16 J=1,JOBS
   16 WRITE (3,4) (JCT(J,I),I=1,MACH)
    4 FORMAT(1H ,10X,12I4)
      GO TO 35
   22 NCNFLT=NCNFLT+1
      RETURN 1
   35 RETURN
      END
```

```
      SUBROUTINE BOUND1 (T,K,LV)
C
C.......................................................................
C
C     THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
C     COMPOSITE-BASED BOUND LB I. THE LOWER BOUND FOR A NODE
C     IS COMPUTED AS THE MAXIMUM OF THE JOB-BASED BOUND
C     LB III (BOUND 3)&THE MACHINE-BASED BOUND LBV (BOUND 5)
C
C.......................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      DIMENSION GREAT(10),GRAT(10),GRT(10),DIF(10)
      IF(IPRINT.EQ.0) GO TO 60
      WRITE (3,86)
   86 FORMAT(1H0,10X,'  ILB   N')
   60 NLV=N(LV)
      DO 20 N1=1,NLV
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      M1=MM(J1,I1)
      DO 11 J=1,JOBS
   11 DIF(J)=0.
      DO 10 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
      IF(N2.EQ.N1) DIF(J2)=0
      IF(N2.EQ.N1) GO TO 10
      DIF(J2)=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
   10 CONTINUE
      JGRT=JCT(J1,I1)
      DO 9 J=1,JOBS
      IF(J1.EQ.J) GO TO 9
      DO 8 I=1,MACH
      KC=MM(J,I)
      IF(KC.NE.K) GO TO 8
      IF(LA(LV,K,J).NE.0.AND.LA(LV,K,J).LT.T) GO TO 8
    6 JGRT=JGRT+IT(J,I)
    8 CONTINUE
    9 CONTINUE
      GREAT(K)=JGRT
      DO 40 L=1,MACH
      LL=0
      IF(K.EQ.L) GO TO 40
      DO 39 J=1,JOBS
      DO 38 I=1,MACH
      KE=MM(J,I)
      IF(KE.NE.L) GO TO 38
      IP=I-1
      IF(IP.EQ.0) GO TO 35
      IF(JCT(J,I).LT.T) GO TO 38
      LL=LL+1
      GRAT(LL)=JCT(J,IP)+DIF(J)
      GO TO 38
   35 JGPP=JCT(J,I)
      IF(JGPP.LT.T) GO TO 38
```

```
          LL=LL+1
          GRAT(LL)=0.
       38 CONTINUE
       39 CONTINUE
          IF(LL-1)26,28,29
       29 JGP=GRAT(1)
          DO 27 LR=2,LL
          IF(GRAT(LR).LT.JGP) JGP=GRAT(LR)
       27 CONTINUE
          JGPR=JGP
          GO TO 7
       28 JGPR=GRAT(1)
        7 DO 50 J=1,JOBS
          DO 48 I=1,MACH
          KG=MM(J,I)
          IF(KG.NE.L) GO TO 48
          IF(LA(LV,L,J).NE.0.AND.LA(LV,L,J).LT.T) GO TO 48
        5 JGPR=JGPR+IT(J,I)
       48 CONTINUE
       50 CONTINUE
          GREAT(L)=JGPR
          GO TO 40
       26 NT=LA(LV,L,1)
          DO 30 J=2,JOBS
          IF(LA(LV,L,J).GT.NT) NT=LA(LV,L,J)
       30 CONTINUE
          GREAT(L)=NT
       40 CONTINUE
          IF(IPRINT.EQ.0) GO TO 12
          WRITE(3,900) (GREAT(MQ),MQ=1,MACH)
      900 FORMAT(1H ,4F8.1)
       12 ILB(LV,N1)=GREAT(1)
          DO 15 I=2,MACH
          IF(ILB(LV,N1).GE.GREAT(I)) GO TO 15
          ILB(LV,N1)=GREAT(I)
       15 CONTINUE
          IF(IPRINT.EQ.0) GO TO 65
          WRITE (3,59) ILB(LV,N1),N1
       59 FORMAT(1H0,10X,2I5)
       65 JXP1=ILB(LV,N1)
          DO 80 MI=1,NLV
       80 GREAT(MI)=0.
          J1=JJ(LV,N1)
          I1=IOP(LV,J1)
          DO 70 N2=1,NLV
          J2=JJ(LV,N2)
          I2=IOP(LV,J2)
          IF(N2.EQ.N1) GREAT(N2)=JCT(J2,MACH)
          IF(N2.EQ.N1) GO TO 70
          KIF=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
          GREAT(N2)=JCT(J2,MACH)+KIF
       70 CONTINUE
          ILB(LV,N1)=GREAT(1)
          DO 75 I=2,NLV
          IF(ILB(LV,N1).GE.GREAT(I)) GO TO 75
          ILB(LV,N1)=GREAT(I)
       75 CONTINUE
          IF(IPRINT.EQ.0) GO TO 66
```

```
      WRITE (3,59) ILB(LV,N1),N1
   66 JXP2=ILB(LV,N1)
      IF(JXP2-JXP1)61,61,62
   61 ILB(LV,N1)=JXP1
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
      GO TO 20
   62 ILB(LV,N1)=JXP2
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
   20 CONTINUE
      RETURN
      END
```

```
 0001                 SUBROUTINE BOUND2 (T,K,LV)
            C
            C.................................................................
            C
            C         THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
            C         COMPOSITE-BASED BOUND LB II. THE LOWER BOUND FOR A
            C         NODE IS COMPUTED AS THE MAXIMUM OF THE JOB-BASED BOUND
            C         LB IVI (BOUND 4)&THE MACHINE-BASED BOUND LBV (BOUND 5)
            C
            C.................................................................
            C
 0002                 COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
 0003                 COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
 0004                 COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
 0005                 DIMENSION GREAT(10),GRAT(10),GRT(10),DIF(10),IZ(10),SMALL(10),
                     1GRETE(10)
 0006                 IF(IPRINT.EQ.0) GO TO 999
 0007                 WRITE (3,86)
 0008              86 FORMAT(1H0,10X,'  ILB   N')
 0009             999 NLV=N(LV)
 0010                 DO 20 N1=1,NLV
 0011                 J1=JJ(LV,N1)
 0012                 I1=IOP(LV,J1)
 0013                 DO 10 N2=1,NLV
 0014                 J2=JJ(LV,N2)
 0015                 I2=IOP(LV,J2)
 0016                 IF(N2.EQ.N1) GREAT(N2)=JCT(J2,MACH-1)
 0017                 IF(N2.EQ.N1) GO TO 10
 0018                 KIF=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
 0019                 GREAT(N2)=JCT(J2,MACH-1)+KIF
 0020              10 CONTINUE
 0021                 N2=NLV
 0022                 DO 30 J=1,JOBS
 0023                 NN=1
 0024               9 IF(JJ(LV,NN).EQ.J) GO TO 30
 0025                 NN=NN+1
 0026                 IF(NN.GT.NLV) GO TO 40
 0027                 GO TO 9
 0028              40 N2=N2+1
 0029                 GREAT(N2)=JCT(J,MACH-1)
 0030                 JJ(LV,N2)=J
 0031              30 CONTINUE
 0032                 DO 600 KX=1,MACH
 0033                 L=0
 0034                 DO 502 NN=1,N2
 0035                 J3=JJ(LV,NN)
 0036                 M3=MM(J3,MACH)
 0037                 IF(M3.NE.KX) GO TO 502
 0038                 L=L+1
 0039                 GRAT(L)=GREAT(NN)
 0040                 IZ(L)=NN
 0041                 MR=NN
 0042             502 CONTINUE
 0043                 IF(L.EQ.0) GO TO 600
 0044                 IF(L.GT.1) GO TO 510
 0045                 LL=0
 0046                 DO 210 IK=1,N2
 0047                 J4=JJ(LV,IK)
```

```
      IF(MM(J4,MACH-1).NE.KX) GO TO 210
      LL=LL+1
      IF(LL.EQ.1) JGRT=GREAT(IK)
  210 CONTINUE
      IF(LL.EQ.0) GO TO 580
      J5=JJ(LV,MR)
      IF(JGRT.GE.GREAT(MR)) GO TO 220
      GRT(MR)=GREAT(MR)+IT(J5,MACH)
      GO TO 600
  220 GRT(MR)=JGRT+IT(J5,MACH)
      GO TO 600
  510 DO 700 M=1,L
      SMALL(M)=GRAT(1)
      MN=IZ(1)
      KP=1
      DO 690 KS=2,L
      IF(GRAT(KS).GE.SMALL(M)) GO TO 690
      SMALL(M)=GRAT(KS)
      MN=IZ(KS)
      KP=KS
  690 CONTINUE
      GRAT(KP)=9999
      J6=JJ(LV,MN)
      IF(M.GT.1) GO TO 691
      IF(M.EQ.1) GRT(MN)=SMALL(M)+IT(J6,MACH)
      IF(M.EQ.1) GRETE(M)=SMALL(M)+IT(J6,MACH)
      GO TO 700
  691 IF(SMALL(M).LE.GRETE(M-1)) GRT(MN)=GRETE(M-1)+IT(J6,MACH)
      IF(SMALL(M).LE.GRETE(M-1)) GRETE(M)=GRETE(M-1)+IT(J6,MACH)
      IF(SMALL(M).GT.GRETE(M-1)) GRT(MN)=SMALL(M)+IT(J6,MACH)
      IF(SMALL(M).GT.GRETE(M-1)) GRETE(M)=SMALL(M)+IT(J6,MACH)
  700 CONTINUE
      GO TO 600
  580 J7=JJ(LV,MR)
      GRT(MR)=GREAT(MR)+IT(J7,MACH)
  600 CONTINUE
      ILB(LV,N1)=GRT(1)
      DO 15 IX=2,N2
      IF(ILB(LV,N1).GE.GRT(IX)) GO TO 15
      ILB(LV,N1)=GRT(IX)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 65
      WRITE (3,59) ILB(LV,N1),N1
   59 FORMAT(1H0,10X,2I5)
   65 JXP1=ILB(LV,N1)
   21 DO 47 L=1,10
      GREAT(L)=0.
      GRT(L)=0.
   47 GRAT(L)=0.
      NLV=N(LV)
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      M1=MM(J1,I1)
      DO 11 J=1,JOBS
   11 DIF(J)=0.
      DO 41 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
```

```
      IF(N2.EQ.N1) DIF(J2)=0
      IF(N2.EQ.N1) GO TO 41
      DIF(J2)=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
   41 CONTINUE
      JGRT=JCT(J1,I1)
      DO 42 J=1,JOBS
      IF(J1.EQ.J) GO TO 42
      DO 8 I=1,MACH
      KC=MM(J,I)
      IF(KC.NE.K) GO TO 8
      IF(LA(LV,K,J).NE.0.AND.LA(LV,K,J).LT.T) GO TO 8
    6 JGRT=JGRT+IT(J,I)
    8 CONTINUE
   42 CONTINUE
      GREAT(K)=JGRT
      DO 43 L=1,MACH
      LL=0
      IF(K.EQ.L) GO TO 43
      DO 39 J=1,JOBS
      DO 38 I=1,MACH
      KE=MM(J,I)
      IF(KE.NE.L) GO TO 38
      IP=I-1
      IF(IP.EQ.0) GO TO 35
      IF(JCT(J,I).LT.T) GO TO 38
      LL=LL+1
      GRAT(LL)=JCT(J,IP)+DIF(J)
      GO TO 38
   35 JGPP=JCT(J,I)
      IF(JGPP.LT.T) GO TO 38
      LL=LL+1
      GRAT(LL)=0.
   38 CONTINUE
   39 CONTINUE
      IF(LL-1)26,28,29
   29 JGP=GRAT(1)
      DO 27 LR=2,LL
      IF(GRAT(LR).LT.JGP) JGP=GRAT(LR)
   27 CONTINUE
      JGPR=JGP
      GO TO 7
   28 JGPR=GRAT(1)
    7 DO 44 J=1,JOBS
      DO 48 I=1,MACH
      KG=MM(J,I)
      IF(KG.NE.L) GO TO 48
      IF(LA(LV,L,J).NE.0.AND.LA(LV,L,J).LT.T) GO TO 48
    5 JGPR=JGPR+IT(J,I)
   48 CONTINUE
   44 CONTINUE
      GREAT(L)=JGPR
      GO TO 43
   26 NT=LA(LV,L,1)
      DO 45 J=2,JOBS
      IF(LA(LV,L,J).GT.NT) NT=LA(LV,L,J)
   45 CONTINUE
      GREAT(L)=NT
   43 CONTINUE
```

```
      IF(IPRINT.EQ.0) GO TO 52
      WRITE(3,900) (GREAT(MQ),MQ=1,MACH)
  900 FORMAT(1H ,4F8.1)
   52 ILB(LV,N1)=GREAT(1)
      DO 46 I=2,MACH
      IF(ILB(LV,N1).GE.GREAT(I)) GO TO 46
      ILB(LV,N1)=GREAT(I)
   46 CONTINUE
      IF(IPRINT.EQ.0) GO TO 66
      WRITE (3,59) ILB(LV,N1),N1
   66 JXP2=ILB(LV,N1)
      IF(JXP2-JXP1)61,61,62
   61 ILB(LV,N1)=JXP1
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
      GO TO 20
   62 ILB(LV,N1)=JXP2
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BOUND3 (T,K,LV)
C
C.....................................................................
C     THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
C     JOB-BASED BOUND III, SUGGESTED BY CONWAY ET AL.
C.....................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      DIMENSION GREAT(10)
      IF(IPRINT.EQ.0) GO TO 60
      WRITE (3,86)
   86 FORMAT(1H ,10X,'  ILB    N')
   60 NLV=N(LV)
      DO 20 N1=1,NLV
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      DO 10 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
      IF(N2.EQ.N1) GREAT(N2)=JCT(J2,MACH)
      IF(N2.EQ.N1) GO TO 10
      DIF=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
      GREAT(N2)=JCT(J2,MACH)+DIF
   10 CONTINUE
      ILB(LV,N1)=GREAT(1)
      DO 15 I=2,NLV
      IF(ILB(LV,N1).GE.GREAT(I)) GO TO 15
      ILB(LV,N1)=GREAT(I)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,50) ILB(LV,N1),N1
   50 FORMAT(1H ,10X,2I5)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BOUND4
C
C.....................................................................
C     THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
C     JOB-BASED BOUND IV , SUGGESTED BY'BROOKS AND WHITE'
C.....................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      DIMENSION GREAT(10),GRAT(10),I(10),GRT(10),SMALL(10),GRETE(10)
      IF(IPRINT.EQ.0) GO TO 999
      WRITE (3,86)
   86 FORMAT(1H0,10X,'  ILB    N')
  999 NLV=N(LV)
      DO 20 N1=1,NLV
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      DO 10 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
      IF(N2.EQ.N1) GREAT(N2)=JCT(J2,MACH-1)
      IF(N2.EQ.N1) GO TO 10
      DIF=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
      GREAT(N2)=JCT(J2,MACH-1)+DIF
   10 CONTINUE
      N2=NLV
      DO 30 J=1,JOBS
      NN=1
    9 IF(JJ(LV,NN).EQ.J) GO TO 30
      NN=NN+1
      IF(NN.GT.NLV) GO TO 40
      GO TO 9
   40 N2=N2+1
      GREAT(N2)=JCT(J,MACH-1)
      JJ(LV,N2)=J
   30 CONTINUE
      DO 600 KX=1,MACH
      L=0
      DO 502 NN=1,N2
      J3=JJ(LV,NN)
      M3=MM(J3,MACH)
      IF(M3.NE.KX) GO TO 502
      L=L+1
      GRAT(L)=GREAT(NN)
      I(L)=NN
      MR=NN
  502 CONTINUE
      IF(L.EQ.0) GO TO 600
      IF(L.GT.1) GO TO 510
      LL=0
      DO 210 IK=1,N2
      J4=JJ(LV,IK)
      IF(MM(J4,MACH-1).NE.KX) GO TO 210
      LL=LL+1
      IF(LL.EQ.1) JGRT=GREAT(IK)
  210 CONTINUE
      IF(LL.EQ.0) GO TO 580
```

```
      J5=JJ(LV,MR)
      IF(JGRT.GE.GREAT(MR)) GO TO 220
      GRT(MR)=GREAT(MR)+IT(J5,MACH)
      GO TO 600
  220 GRT(MR)=JGRT+IT(J5,MACH)
      GO TO 600
  510 DO 700 M=1,L
      SMALL(M)=GRAT(1)
      MN=I(1)
      KP=1
      DO 690 KS=2,L
      IF(GRAT(KS).GE.SMALL(M)) GO TO 690
      SMALL(M)=GRAT(KS)
      MN=I(KS)
      KP=KS
  690 CONTINUE
      GRAT(KP)=9999
      J6=JJ(LV,MN)
      IF(M.GT.1) GO TO 691
      IF(M.EQ.1) GRT(MN)=SMALL(M)+IT(J6,MACH)
      IF(M.EQ.1) GRETE(M)=SMALL(M)+IT(J6,MACH)
      GO TO 700
  691 IF(SMALL(M).LE.GRETE(M-1)) GRT(MN)=GRETE(M-1)+IT(J6,MACH)
      IF(SMALL(M).LE.GRETE(M-1)) GRETE(M)=GRETE(M-1)+IT(J6,MACH)
      IF(SMALL(M).GT.GRETE(M-1)) GRT(MN)=SMALL(M)+IT(J6,MACH)
      IF(SMALL(M).GT.GRETE(M-1)) GRETE(M)=SMALL(M)+IT(J6,MACH)
  700 CONTINUE
      GO TO 600
  580 J7=JJ(LV,MR)
      GRT(MR)=GREAT(MR)+IT(J7,MACH)
  600 CONTINUE
      ILB(LV,N1)=GRT(1)
      DO 15 IX=2,N2
      IF(ILB(LV,N1).GE.GRT(IX)) GO TO 15
      ILB(LV,N1)=GRT(IX)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,50) ILB(LV,N1),N1
   50 FORMAT(1H0,10X,2I5)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BOUND5 (T,K,LV)
C
C.......................................................................
C     THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
C     MACHINE-BASED BOUND LB V,  SUGGESTED BY CONWAY ET AL.
C.......................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      DIMENSION GREAT(10),GRAT(10),GRT(10),DIF(10)
      IF(IPRINT.EQ.0) GO TO 60
      WRITE (3,86)
   86 FORMAT(1H0,10X,'  ILB    N')
   60 NLV=N(LV)
      DO 20 N1=1,NLV
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      M1=MM(J1,I1)
      DO 11 J=1,JOBS
   11 DIF(J)=0.
      DO 10 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
      IF(N2.EQ.N1) DIF(J2)=0
      IF(N2.EQ.N1) GO TO 10
      DIF(J2)=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
   10 CONTINUE
      JGRT=JCT(J1,I1)
      DO 9 J=1,JOBS
      IF(J1.EQ.J) GO TO 9
      DO 8 I=1,MACH
      KC=MM(J,I)
      IF(KC.NE.K) GO TO 8
      IF(LA(LV,K,J).NE.0.AND.LA(LV,K,J).LT.T) GO TO 8
    6 JGRT=JGRT+IT(J,I)
    8 CONTINUE
    9 CONTINUE
      GREAT(K)=JGRT
      DO 40 L=1,MACH
      LL=0
      IF(K.EQ.L) GO TO 40
      DO 39 J=1,JOBS
      DO 38 I=1,MACH
      KE=MM(J,I)
      IF(KE.NE.L) GO TO 38
      IP=I-1
      IF(IP.EQ.0) GO TO 35
      IF(JCT(J,I).LT.T) GO TO 38
      LL=LL+1
      GRAT(LL)=JCT(J,IP)+DIF(J)
      GO TO 38
   35 JGPP=JCT(J,I)
      IF(JGPP.LT.T) GO TO 38
      LL=LL+1
      GRAT(LL)=0.
   38 CONTINUE
   39 CONTINUE
```

```
      IF(LL-1)26,28,29
   29 JGP=GRAT(1)
      DO 27 LR=2,LL
      IF(GRAT(LR).LT.JGP) JGP=GRAT(LR)
   27 CONTINUE
      JGPR=JGP
      GO TO 7
   28 JGPR=GRAT(1)
    7 DO 50 J=1,JOBS
      DO 48 I=1,MACH
      KG=MM(J,I)
      IF(KG.NE.L) GO TO 48
      IF(LA(LV,L,J).NE.0.AND.LA(LV,L,J).LT.T) GO TO 48
    5 JGPR=JGPR+IT(J,I)
   48 CONTINUE
   50 CONTINUE
      GREAT(L)=JGPR
      GO TO 40
   26 NT=LA(LV,L,1)
      DO 30 J=2,JOBS
      IF(LA(LV,L,J).GT.NT) NT=LA(LV,L,J)
   30 CONTINUE
      GREAT(L)=NT
   40 CONTINUE
      IF(IPRINT.EQ.0) GO TO 12
      WRITE(3,900) (GREAT(MQ),MQ=1,MACH)
  900 FORMAT(1H ,4F8.1)
   12 ILB(LV,N1)=GREAT(1)
      DO 15 I=2,MACH
      IF(ILB(LV,N1).GE.GREAT(I)) GO TO 15
      ILB(LV,N1)=GREAT(I)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
   59 FORMAT(1H0,10X,2I5)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SMLILB(LV)
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      NLV=N(LV)
  120 DO 300 NL=1,NLV
      IF(JJ(LV,NL).NE.JACTIV(LV)) GO TO 300
      NK=NL
  300 CONTINUE
      ISWTCH=0
      NLV=N(LV)
      DO 10 NL=1,NLV
      IF(JJ(LV,NL).EQ.JACTIV(LV)) GO TO 10
      IF(ISTMIN.LE.ILB(LV,NL)) GO TO 10
      IF(ILB(LV,NL)-NILB(LV))10,8,9
    8 IF(NL.LT.NK) GO TO 10
    9 ISWTCH=ISWTCH+1
      IF(ISWTCH-1)11,11,12
   11 NT=ILB(LV,NL)
      NTL=NL
      GO TO 10
   12 IF(ILB(LV,NL).GE.NT) GO TO 10
      NT=ILB(LV,NL)
      NTL=NL
   10 CONTINUE
      NILB(LV)=NT
      JACTIV(LV)=JJ(LV,NTL)
   21 LV=LV+1
      DO 20 M=1,MACH
      DO 20 J=1,JOBS
   20 LA(LV,M,J)=LA(LV-1,M,J)
      DO 30 J=1,JOBS
   30 IOP(LV,J)=IOP(LV-1,J)
      IA=IOP(LV,JACTIV(LV-1))
      L=N(LV-1)
      K=MM(JACTIV(LV-1),IA)
      JCT(JACTIV(LV-1),IA)=LA(LV-1,K,JACTIV(LV-1))
      DO 15 NL=1,L
      IF(JJ(LV-1,NL).EQ.JACTIV(LV-1)) GO TO 15
      J1=JJ(LV-1,NL)
      I1=IOP(LV,J1)
      JCT(J1,I1)=JCT(JACTIV(LV-1),IA)+IT(J1,I1)
      M=MM(J1,I1)
      LA(LV,M,J1)=JCT(J1,I1)
      IK=I1+1
      IF(IK.GT.MACH) GO TO 15
      DO 14 I=IK,MACH
   14 JCT(J1,I)=JCT(J1,I-1)+IT(J1,I)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 35
      WRITE (3,88)
   88 FORMAT(1H0,10X,       'COMPLETION TIME MATRIX*')
      DO 13 J=1,JOBS
   13 WRITE (3,4) (JCT(J,I),I=1,MACH)
    4 FORMAT(1H ,15X,12I4)
   35 RETURN
      END
```

ILLEGIBLE

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

ILLEGIBLE

PROBLEM NUMBER = 1

PROCESSING TIME MATRIX

21	20	25	19	27
1	20	26	24	5
22	24	12	4	16
25	30	22	16	15
23	10	2	12	30
5	11	18	21	29
5	28	16	1	3
13	22	16	15	19

MACHINE ORDERING MATRIX

2	3	5	1	4
3	5	4	2	1
3	5	4	2	1
3	5	2	4	1
5	3	1	2	4
5	2	1	3	4
3	5	4	1	2
2	1	4	3	5

BOUNDING PROCEDURE 1

```
SOLUTION      218
IO CF CONFLICT LEVELS FOR SOLN     30
SOLUTION      217
IO OF CONFLICT LEVELS FOR SOLN     26
SOLUTION      214
IO OF CONFLICT LEVELS FOR SOLN     29
```

```
      C
      C     ***   BRANCH-AND-BOUND ALGORITHM   ***
      C
      C     ***        FOR JOB-SHOP PROBLEMS     ***
      C
      C     PROGRAMMED BY
      C                  S. R. HIREMATH
      C
      C.......................................................................
      C
      C.......................................................................
      C     THE BRANCH AND BOUND ALGORITHM DESCRIBED IN SECTION 2.4
      C     IS PROGRAMMED IN FORTRAN IV
      C     THIS PROGRAM CONSISTS OF MAIN PROGAM AND FIVE BOUNDING
      C     PROCEDURES AS SUBROUTINES. IN ADDISON IT ALSO CONSISTS
      C     OF THREE MORE SUBROUTINES.
      C
      C.......................................................................
      C
      C.......................................................................
      C
      C     *****    VARIABLES    *****
      C
      C     IT              PROCESSING TIME
      C     MM              MACHINE ORDERING
      C     JCT             COMPLETION TIME
      C     MACH            TOTAL NUMBER OF MACHINES OR OPERATIONS FOR
      C     JOBS            TOTAL NO. OF JOBS
      C                     A JOB
      C     LA              ENTRY IN THE SCHEDULING TABLE
      C     IOP             OPERATION
      C     JJ              JOB IN THE CONFLICT SET
      C     N               NO. OF JOBS IN CONFLICT SET AT A LEVEL
      C     ILB             LOWER BOUND FOR A NODE
      C     NILB            MIN. LOWER BOUND AT A LEVEL
      C     ISTMIN          SCHEDULE TIME
      C     JACTIV          ACTIVE NODE AT A LEVEL
      C
      C.......................................................................
      C
      C
      C     IREAD.EQ.0       GENERATING DATA (MACHINE-ORDERING AND
      C                      PROCESSING TIME MATRICES)
      C     IREAD.NE.0       READ DATA CARDS FOR  BOTH MATRICES
      C
      C     IF IPRINT.EQ.0   PRINT DETAILS
      C     IF IPRINT.NE.0   DO NOT PRINT DETAILS
      C
      C     IF ICARD.EQ.0    NO CARD OUTPUT DESIRED
      C     IF ICARD.NE.0    CARD OUTPUT DESIRED
      C
      C     LIMIT1&LIMIT2    THE LIMITS OF INTERVAL FOR PROCESS-
      C                      ING TIMES
      C
      C.......................................................................
      C
      C      MAIN PROGRAM
      C
```

```
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      DIMENSION IRAND2(50)
      READ(1,1) MACH,JOBS,LIMIT1,LIMIT2,NPROB,NFLB,NLLB,IREAD,ISKIP,
     1IPRINT,ICARD,IX,IY,IB
    1 FORMAT(11I4,2I8,I4)
      DO 32 NP=1,NPROB
      WRITE (3,33) NP
   33 FORMAT(1H0,10X,' PROBLEM NUMBER = ',I3)
      IF(IREAD.EQ.0) GO TO 600
C
C     READ PROCESSING TIME MATRIX
C
      DO 9 J=1,JOBS
    9 READ(1,2) (IT(J,I),I=1,MACH)
    2 FORMAT(10I5)
      GO TO 77
C
C     GENERATE PROCESSING TIME MATRIX
C
  600 DO 211 M=1,MACH
      DO 211 J=1,JOBS
  211 IT(J,M)=RANDNO(IY)*(LIMIT2-LIMIT1+1)+LIMIT1
   77 WRITE (3,82)
   82 FORMAT(1H ,10X,       'PROCESSING TIME MATRIX')
      DO 11 J=1,JOBS
   11 WRITE (3,4) (IT(J,I),I=1,MACH)
    4 FORMAT(1H ,10X,12I4)
      IF(IREAD.EQ.0) GO TO 698
C
C     READ MACHINE-ORDERING MATRIX
C
      DO 3 J=1,JOBS
    3 READ(1,2) (MM(J,I),I=1,MACH)
      GO TO 76
C
C     GENERATE MACHINE ORDERING MATRIX
C
  698 DO 235 J=1,JOBS
      DO 231 M=1,MACH
  231 IRAND2(M)=M
      M1=MACH
  232 IRAN=RANDNO(IY)*M1+1
      MM(J,M1)=IRAND2(IRAN)+100*J
      IF (IRAN .EQ. M1) GO TO 234
      M1=M1-1
      IF (M1 .EQ. 0) GO TO 235
      DO 233 M2=IRAN,M1
  233 IRAND2(M2)=IRAND2(M2+1)
      GO TO 232
  234 IF (M1 .EQ. 1) GO TO 235
      M1=M1-1
      GO TO 232
  235 CONTINUE
   76 DO 96 J=1,JOBS
      DO 96 I=1,MACH
   96 MM(J,I)=MM(J,I)-J*100
```

```
            WRITE (3,81)
         81 FORMAT(1H ,10X,      'MACHINE ORDERING MATRIX')
            DO 8 J=1,JOBS
          8 WRITE (3,4) (MM(J,I),I=1,MACH)
            WRITE (3,35) IB
         35 FORMAT(1H0,10X,' BOUNDING PROCEDURE',I3)
      C
      C     FORM COMPLETION TIME MATRIX
      C
            DO 10 J=1,JOBS
            JCT(J,1)=IT(J,1)
            DO 10 I=2,MACH
            JCT(J,I)=JCT(J,I-1)+IT(J,I)
         10 CONTINUE
            IF(IPRINT.EQ.0) GO TO 21
            WRITE (3,83)
         83 FORMAT(1H ,10X,      'COMPLETION TIME MATRIX')
            DO 13 J=1,JOBS
         13 WRITE (3,2) (JCT(J,I),I=1,MACH)
      C
      C     INITIALIZE
      C     SET UP SCHEDULING TABLE
      C
         21 DO 40 LV=1,90
            DO 40 I=1,MACH
            DO 40 J=1,JOBS
         40 LA(LV,I,J)=0
            DO 41 LV=1,90
            DO 41 J=1,JOBS
         41 IOP(LV,J)=1
      C
      C     ENTER FIRST OPERATIONS OF EACH JOB
      C
            T=0.
            ISWTCH=0
            ISTMIN=99999
            LV=1
            NBKTRK=0
            NNODES=0
            NCNFLT=0
            CALL TIME(NT1)
            DO 50 J=1,JOBS
            M=MM(J,1)
         50 LA(1,M,J)=JCT(J,1)
      C
      C     FIND SMALLEST T AND NEXT HIGHER T
      C
         51 CALL SMALLT(T,LV)
            IF(IPRINT.EQ.0) GO TO 22
            WRITE (3,84)
         84 FORMAT(1H ,10X,      'SCHEDULING TABLE')
            WRITE (3,7) ((LA(LV,K,J),J=1,JOBS),K=1,MACH)
          7 FORMAT(1H ,10X,30I4)
      C
      C     CHECK FOR CONFLICT
      C
        551 WRITE (3,85)
         85 FORMAT(1H ,10X,' JOB MC   LV  JCT')
```

```
   22 DO 70 K=1,MACH
      DO 72 J=1,JOBS
      IF(LA(LV,K,J).NE.T) GO TO 72
      N(LV)=0
      DO 69 JM=1,JOBS
   99 IF(LA(LV,K,JM).GE.T) GO TO 68
      GO TO 69
   68 N(LV)=N(LV)+1
      JJ(LV,N(LV))=JM
      IF(IPRINT.EQ.0) GO TO 69
      WRITE (3,65) JJ(LV,N(LV)),K,LV,LA(LV,K,JM)
   65 FORMAT(1H ,10X,4I4)
   69 CONTINUE
C
C     DETERMINE LOWER BOUNDS AND RESOLVE CONFLICT
C
      IF(N(LV).GT.1) CALL CONFLT(T,K,LV,NNODES,NCNFLT,&110)
      GO TO 70
   72 CONTINUE
   70 CONTINUE
C
C     UPDATE THE ARRAY AND ENTER NEXT OPERATION
C
   89 DO 80 K=1,MACH
      DO 80 J=1,JOBS
      IF(LA(LV,K,J).NE.T) GO TO 80
      IOP(LV,J)=IOP(LV,J)+1
      IF(IOP(LV,J).GT.MACH) GO TO 75
      KK=MM(J,IOP(LV,J))
      LA(LV,KK,J)=JCT(J,IOP(LV,J))
      GO TO 80
   75 IOP(LV,J)=IOP(LV,J)-1
   80 CONTINUE
C
C     CHECK FOR T
C     IF T IS THE HIGHEST ENTRY A SOLUTION HAS BEEN FOUND
C     OTHERWISE FIND NEXT HIGHER T
C
   79 DO 16 M=1,MACH
      DO 16 J=1,JOBS
      IF(T.LT.LA(LV,M,J)) GO TO 51
   16 CONTINUE
      IF(T.GE.ISTMIN) GO TO 110
      ISTMIN=T
      WRITE (3,6) ISTMIN
    6 FORMAT(1H ,' A SOLUTION   ',I6)
  980 FORMAT(1H0,10X,' COMPUTATION TIME =',F12.4)
      LEVEL=LV-1
      WRITE(3,1001) LEVEL
 1001 FORMAT(1H ,' NO OF CONFLICT LEVELS FOR SOLN',I6)
      ISWTCH=ISWTCH+1
      NBKTRK=NBKTRK+1
      IF(ISWTCH.EQ.1) NBKTRK=0
      IF(ISWTCH.NE.1) GO TO 110
      IF(ICARD.EQ.0) GO TO 110
      WRITE (2,1003) ISTMIN
 1003 FORMAT(I8)
C
```

```
C       BACKTRACKING
C
    110 DO 95 I=1,MACH
        DO 95 J=1,JOBS
     95 LA(LV,MM(J,I),J)=0
        LV=LV-1
        IF(IPRINT.EQ.0) GO TO 23
        WRITE (3,501) LV
    501 FORMAT(1H ,' LEVEL',I5)
     23 NLV=N(LV)
    120 DO 300 NL=1,NLV
        IF(JJ(LV,NL).NE.JACTIV(LV)) GO TO 300
        NK=NL
    300 CONTINUE
        DO 360 NL=1,NLV
        IF(JJ(LV,NL).EQ.JACTIV(LV)) GO TO 360
        IF(ISTMIN.LE.ILB(LV,NL)) GO TO 360
        IF(ILB(LV,NL).LT.NILB(LV)) GO TO 360
        IF(ILB(LV,NL).GT.NILB(LV)) GO TO 310
        IF(NL.GT.NK) GO TO 310
    360 CONTINUE
        IF(LV-1)400,400,110
    310 JL=JJ(LV,1)
        IL=IOP(LV,JL)
        ML=MM(JL,IL)
        T=LA(LV,ML,JL)
        DO 350 NL=2,NLV
        J2=JJ(LV,NL)
        I2=IOP(LV,J2)
        M2=MM(J2,I2)
        IF(LA(LV,M2,J2).LT.T) T=LA(LV,M2,J2)
    350 CONTINUE
        IF(IPRINT.EQ.0) GO TO 24
        WRITE (3,48) T
     48 FORMAT(1H ,' T*',F8.1)
     24 DO 583 J=1,JOBS
        DO 580 I=1,MACH
        M=MM(J,I)
        IF(LA(LV,M,J).EQ.0) GO TO 585
        JCT(J,I)=LA(LV,M,J)
        GO TO 580
    585 JCT(J,I)=JCT(J,I-1)+IT(J,I)
    580 CONTINUE
    583 CONTINUE
        CALL SMLILB(LV)
        GO TO 22
    400 WRITE (3,509) ISTMIN
    509 FORMAT(1HO,10X,' OPTIMAL SCHEDULE TIME = ',I6)
        WRITE (3,73) NNODES
        WRITE (3,74) NCNFLT
        WRITE (3,78) NBKTRK
        CALL TIME(NT2)
        COTIME=(NT2-NT1)/100.
        WRITE(3,980) COTIME
        IF(ICARD.EQ.0) GO TO 32
        WRITE(2,902) NNODES,NCNFLT,NBKTRK,COTIME,ISTMIN
     78 FORMAT(1HO,10X,' NUMBER OF BACKTRACKS =',I12)
     73 FORMAT(1HO,10X,' NUMBER OF NODES EXPLORED =',I12)
```

```
0190        74 FORMAT(1H0,10X,' NUMBER OF CONFLTS =',I12)
0191       902 FORMAT(3I12,F11.3,I10)
0192        32 CONTINUE
0193       100 STOP
0194           END
```

```
      FUNCTION RANDNO(IY)
      IY=IY*65627
      IF(IY)5,6,6
    5 IY=IY+2147483647+1
    6 RANDNO=IY*.4656613E-9
      RETURN
      END
```

```
0001            SUBROUTINE SMALLT(T,LV)
          C
          C.................................................................
          C
          C     THIS SUBROUTINE FINDS THE SMALLEST NUMBER IN THE SCHE-
          C     DULING TABLE AT LEVEL 1 IN THE BEGINNING. EVERYTIME,
          C     IT UPDATES THE VALUE OF T TO THE NEXT HIGHER VALUE IN
          C     THE SCHEDULING TABLE AT A LEVEL LV.
          C
          C.................................................................
          C
0002            COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
0003            COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
0004            COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
0005            ISWTCH=0
0006            DO 20 M=1,MACH
0007            DO 20 J=1,JOBS
0008            IF(LA(LV,M,J).LE.T) GO TO 20
0009            ISWTCH=ISWTCH+1
0010            IF(ISWTCH.EQ.1) TS=LA(LV,M,J)
0011         10 IF(LA(LV,M,J).LT.TS) TS=LA(LV,M,J)
0012         20 CONTINUE
0013            T=TS
0014            IF(IPRINT.EQ.0) GO TO 25
0015            WRITE (3,502) T
0016        502 FORMAT(1H ,' T  ',F8.1)
0017         25 RETURN
0018            END
```

```
0001             SUBROUTINE CONFLT (T,K,LV,NNODES,NCNFLT,*)
           C
           C.....................................................................
           C
           C     THIS SUBROUTINE CHECKS FOR CONFLICT. IF A CONFLICT
           C     EXISTS, THE LOWER BOUNDS FOR THE NODES IN THE CONFLICT
           C     SET ARE COMPUTED USING ONE OF THE LOWER BOUNDS.
           C
           C.....................................................................
           C
0002             COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
0003             COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
0004             COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
0005             NLV=N(LV)
0006             L=0
           C
           C     CHECKING FOR CONFLICT USING BEENHAKKERS FORMULA.
           C
0007             DO 40 KL=1,NLV
0008             J2=JJ(LV,KL)
0009             I2=IOP(LV,J2)
0010             M2=MM(J2,I2)
0011             IF(LA(LV,M2,J2).EQ.T) GO TO 39
0012             IF((T+IT(J2,I2)).LE.JCT(J2,I2)) GO TO 40
0013          39 L=L+1
0014          40 CONTINUE
0015             IF(L.LE.1) GO TO 35
           C
           C     IF CONFLICT EXISTS, ONE OF THE BOUND SUBROUTINE IS
           C     CALLED TO COMPUTE THE LOWER BOUNDS FOR THE NODES IN
           C     THE CONFLICT SET AT A LEVEL.
           C
0016             GO TO(91,92,93,94,95),IB
0017          91 CALL BOUND1(T,K,LV)
0018             GO TO 96
0019          92 CALL BOUND2(T,K,LV)
0020             GO TO 96
0021          93 CALL BOUND3(T,K,LV)
0022             GO TO 96
0023          94 CALL BOUND4(T,K,LV)
0024             GO TO 96
0025          95 CALL BOUND5(T,K,LV)
0026          96 NILB(LV)=ILB(LV,1)
           C
           C     DETERMINE THE NODE WITH MINIMUM LOWER BOUND. IF A TIE
           C     EXISTS, IT IS BROKEN USING THE LEFT HAND RULE.
           C
0027             NNODES=NNODES+N(LV)
0028             JACTIV(LV)=JJ(LV,1)
0029             DO 10 NL=2,NLV
0030             IF(NILB(LV).LE.ILB(LV,NL)) GO TO 10
0031             NILB(LV)=ILB(LV,NL)
0032             JACTIV(LV)=JJ(LV,NL)
0033          10 CONTINUE
0034             IF(NILB(LV)-ISTMIN)21,22,22
0035          21 LV=LV+1
0036             NCNFLT=NCNFLT+1
0037             DO 20 M=1,MACH
```

```
              DO 20 J=1,JOBS
           20 LA(LV,M,J)=LA(LV-1,M,J)
              DO 30 J=1,JOBS
           30 IOP(LV,J)=IOP(LV-1,J)
              IA=IOP(LV,JACTIV(LV-1))
              L=N(LV-1)
              DO 15 NL=1,L
              IF(JJ(LV-1,NL).EQ.JACTIV(LV-1)) GO TO 15
              J1=JJ(LV-1,NL)
      C
      C       UPDATE THE COMPLETION TIME MATRIX IN FAVOR OF THE NODE
      C
              I1=IOP(LV,J1)
              JCT(J1,I1)=JCT(JACTIV(LV-1),IA)+IT(J1,I1)
              M=MM(J1,I1)
              LA(LV,M,J1)=JCT(J1,I1)
              IK=I1+1
              IF(IK.GT.MACH) GO TO 15
              DO 14 IC=IK,MACH
           14 JCT(J1,IC)=JCT(J1,IC-1)+IT(J1,IC)
           15 CONTINUE
              IF(IPRINT.EQ.0) GO TO 35
              WRITE (3,83)
           83 FORMAT(1H0,10X,        'COMPLETION TIME MATRIX*')
              DO 16 J=1,JOBS
           16 WRITE (3,4) (JCT(J,I),I=1,MACH)
            4 FORMAT(1H ,10X,12I4)
              GO TO 35
           22 NCNFLT=NCNFLT+1
              RETURN 1
           35 RETURN
              END
```

```
0001                SUBROUTINE BOUND1 (T,K,LV)
              C
              C.................................................................
              C
              C     THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
              C     COMPOSITE-BASED BOUND LB I. THE LOWER BOUND FOR A NODE
              C     IS COMPUTED AS THE MAXIMUM OF THE JOB-BASED BOUND
              C     LB III (BOUND 3)&THE MACHINE-BASED BOUND LBV (BOUND 5)
              C
              C.................................................................
              C
0002                COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
0003                COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
0004                COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
0005                DIMENSION GREAT(10),GRAT(10),GRT(10),DIF(10)
0006                IF(IPRINT.EQ.0) GO TO 60
0007                WRITE (3,86)
0008             86 FORMAT(1H0,10X,'  ILB   N')
0009             60 NLV=N(LV)
0010                DO 20 N1=1,NLV
0011                J1=JJ(LV,N1)
0012                I1=IOP(LV,J1)
0013                M1=MM(J1,I1)
0014                DO 11 J=1,JOBS
0015             11 DIF(J)=0.
0016                DO 10 N2=1,NLV
0017                J2=JJ(LV,N2)
0018                I2=IOP(LV,J2)
0019                IF(N2.EQ.N1) DIF(J2)=0
0020                IF(N2.EQ.N1) GO TO 10
0021                DIF(J2)=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
0022             10 CONTINUE
0023                JGRT=JCT(J1,I1)
0024                DO 9 J=1,JOBS
0025                IF(J1.EQ.J) GO TO 9
0026                DO 8 I=1,MACH
0027                KC=MM(J,I)
0028                IF(KC.NE.K) GO TO 8
0029                IF(LA(LV,K,J).NE.0.AND.LA(LV,K,J).LT.T) GO TO 8
0030              6 JGRT=JGRT+IT(J,I)
0031              8 CONTINUE
0032              9 CONTINUE
0033                GREAT(K)=JGRT
0034                DO 40 L=1,MACH
0035                LL=0
0036                IF(K.EQ.L) GO TO 40
0037                DO 39 J=1,JOBS
0038                DO 38 I=1,MACH
0039                KE=MM(J,I)
0040                IF(KE.NE.L) GO TO 38
0041                IP=I-1
0042                IF(IP.EQ.0) GO TO 35
0043                IF(JCT(J,I).LT.T) GO TO 38
0044                LL=LL+1
0045                GRAT(LL)=JCT(J,IP)+DIF(J)
0046                GO TO 38
0047             35 JGPP=JCT(J,I)
0048                IF(JGPP.LT.T) GO TO 38
```

```
      LL=LL+1
      GRAT(LL)=0.
   38 CONTINUE
   39 CONTINUE
      IF(LL-1)26,28,29
   29 JGP=GRAT(1)
      DO 27 LR=2,LL
      IF(GRAT(LR).LT.JGP) JGP=GRAT(LR)
   27 CONTINUE
      JGPR=JGP
      GO TO 7
   28 JGPR=GRAT(1)
    7 DO 50 J=1,JOBS
      DO 48 I=1,MACH
      KG=MM(J,I)
      IF(KG.NE.L) GO TO 48
      IF(LA(LV,L,J).NE.0.AND.LA(LV,L,J).LT.T) GO TO 48
    5 JGPR=JGPR+IT(J,I)
   48 CONTINUE
   50 CONTINUE
      GREAT(L)=JGPR
      GO TO 40
   26 NT=LA(LV,L,1)
      DO 30 J=2,JOBS
      IF(LA(LV,L,J).GT.NT) NT=LA(LV,L,J)
   30 CONTINUE
      GREAT(L)=NT
   40 CONTINUE
      IF(IPRINT.EQ.0) GO TO 12
      WRITE(3,900) (GREAT(MQ),MQ=1,MACH)
  900 FORMAT(1H ,4F8.1)
   12 ILB(LV,N1)=GREAT(1)
      DO 15 I=2,MACH
      IF(ILB(LV,N1).GE.GREAT(I)) GO TO 15
      ILB(LV,N1)=GREAT(I)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 65
      WRITE (3,59) ILB(LV,N1),N1
   59 FORMAT(1H0,10X,2I5)
   65 JXP1=ILB(LV,N1)
      DO 80 MI=1,NLV
   80 GREAT(MI)=0.
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      DO 70 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
      IF(N2.EQ.N1) GREAT(N2)=JCT(J2,MACH)
      IF(N2.EQ.N1) GO TO 70
      KIF=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
      GREAT(N2)=JCT(J2,MACH)+KIF
   70 CONTINUE
      ILB(LV,N1)=GREAT(1)
      DO 75 I=2,NLV
      IF(ILB(LV,N1).GE.GREAT(I)) GO TO 75
      ILB(LV,N1)=GREAT(I)
   75 CONTINUE
      IF(IPRINT.EQ.0) GO TO 66
```

```
      WRITE (3,59) ILB(LV,N1),N1
   66 JXP2=ILB(LV,N1)
      IF(JXP2-JXP1)61,61,62
   61 ILB(LV,N1)=JXP1
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
      GO TO 20
   62 ILB(LV,N1)=JXP2
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BOUND2 (T,K,LV)
C
C.......................................................................
C
C     THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
C     COMPOSITE-BASED BOUND LB II. THE LOWER BOUND FOR A
C     NODE IS COMPUTED AS THE MAXIMUM OF THE JOB-BASED BOUND
C     LB IVI (BOUND 4)&THE MACHINE-BASED BOUND LBV (BOUND 5)
C
C.......................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      DIMENSION GREAT(10),GRAT(10),GRT(10),DIF(10),IZ(10),SMALL(10),
     1GRETE(10)
      IF(IPRINT.EQ.0) GO TO 999
      WRITE (3,86)
   86 FORMAT(1H0,10X,'  ILB   N')
  999 NLV=N(LV)
      DO 20 N1=1,NLV
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      DO 10 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
      IF(N2.EQ.N1) GREAT(N2)=JCT(J2,MACH-1)
      IF(N2.EQ.N1) GO TO 10
      KIF=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
      GREAT(N2)=JCT(J2,MACH-1)+KIF
   10 CONTINUE
      N2=NLV
      DO 30 J=1,JOBS
      NN=1
    9 IF(JJ(LV,NN).EQ.J) GO TO 30
      NN=NN+1
      IF(NN.GT.NLV) GO TO 40
      GO TO 9
   40 N2=N2+1
      GREAT(N2)=JCT(J,MACH-1)
      JJ(LV,N2)=J
   30 CONTINUE
      DO 600 KX=1,MACH
      L=0
      DO 502 NN=1,N2
      J3=JJ(LV,NN)
      M3=MM(J3,MACH)
      IF(M3.NE.KX) GO TO 502
      L=L+1
      GRAT(L)=GREAT(NN)
      IZ(L)=NN
      MR=NN
  502 CONTINUE
      IF(L.EQ.0) GO TO 600
      IF(L.GT.1) GO TO 510
      LL=0
      DO 210 IK=1,N2
      J4=JJ(LV,IK)
```

```
      IF(MM(J4,MACH-1).NE.KX) GO TO 210
      LL=LL+1
      IF(LL.EQ.1) JGRT=GREAT(IK)
  210 CONTINUE
      IF(LL.EQ.0) GO TO 580
      J5=JJ(LV,MR)
      IF(JGRT.GE.GREAT(MR)) GO TO 220
      GRT(MR)=GREAT(MR)+IT(J5,MACH)
      GO TO 600
  220 GRT(MR)=JGRT+IT(J5,MACH)
      GO TO 600
  510 DO 700 M=1,L
      SMALL(M)=GRAT(1)
      MN=IZ(1)
      KP=1
      DO 690 KS=2,L
      IF(GRAT(KS).GE.SMALL(M)) GO TO 690
      SMALL(M)=GRAT(KS)
      MN=IZ(KS)
      KP=KS
  690 CONTINUE
      GRAT(KP)=9999
      J6=JJ(LV,MN)
      IF(M.GT.1) GO TO 691
      IF(M.EQ.1) GRT(MN)=SMALL(M)+IT(J6,MACH)
      IF(M.EQ.1) GRETE(M)=SMALL(M)+IT(J6,MACH)
      GO TO 700
  691 IF(SMALL(M).LE.GRETE(M-1)) GRT(MN)=GRETE(M-1)+IT(J6,MACH)
      IF(SMALL(M).LE.GRETE(M-1)) GRETE(M)=GRETE(M-1)+IT(J6,MACH)
      IF(SMALL(M).GT.GRETE(M-1)) GRT(MN)=SMALL(M)+IT(J6,MACH)
      IF(SMALL(M).GT.GRETE(M-1)) GRETE(M)=SMALL(M)+IT(J6,MACH)
  700 CONTINUE
      GO TO 600
  580 J7=JJ(LV,MR)
      GRT(MR)=GREAT(MR)+IT(J7,MACH)
  600 CONTINUE
      ILB(LV,N1)=GRT(1)
      DO 15 IX=2,N2
      IF(ILB(LV,N1).GE.GRT(IX)) GO TO 15
      ILB(LV,N1)=GRT(IX)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 65
      WRITE (3,59) ILB(LV,N1),N1
   59 FORMAT(1H0,10X,2I5)
   65 JXP1=ILB(LV,N1)
   21 DO 47 L=1,10
      GREAT(L)=0.
      GRT(L)=0.
   47 GRAT(L)=0.
      NLV=N(LV)
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      M1=MM(J1,I1)
      DO 11 J=1,JOBS
   11 DIF(J)=0.
      DO 41 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
```

```
      IF(N2.EQ.N1) DIF(J2)=0
      IF(N2.EQ.N1) GO TO 41
      DIF(J2)=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
   41 CONTINUE
      JGRT=JCT(J1,I1)
      DO 42 J=1,JOBS
      IF(J1.FQ.J) GO TO 42
      DO 8 I=1,MACH
      KC=MM(J,I)
      IF(KC.NE.K) GO TO 8
      IF(LA(LV,K,J).NE.0.AND.LA(LV,K,J).LT.T) GO TO 8
    6 JGRT=JGRT+IT(J,I)
    8 CONTINUE
   42 CONTINUE
      GREAT(K)=JGRT
      DO 43 L=1,MACH
      LL=0
      IF(K.EQ.L) GO TO 43
      DO 39 J=1,JOBS
      DO 38 I=1,MACH
      KE=MM(J,I)
      IF(KE.NE.L) GO TO 38
      IP=I-1
      IF(IP.EQ.0) GO TO 35
      IF(JCT(J,I).LT.T) GO TO 38
      LL=LL+1
      GRAT(LL)=JCT(J,IP)+DIF(J)
      GO TO 38
   35 JGPP=JCT(J,I)
      IF(JGPP.LT.T) GO TO 38
      LL=LL+1
      GRAT(LL)=0.
   38 CONTINUE
   39 CONTINUE
      IF(LL-1)26,28,29
   29 JGP=GRAT(1)
      DO 27 LR=2,LL
      IF(GRAT(LR).LT.JGP) JGP=GRAT(LR)
   27 CONTINUE
      JGPR=JGP
      GO TO 7
   28 JGPR=GRAT(1)
    7 DO 44 J=1,JOBS
      DO 48 I=1,MACH
      KG=MM(J,I)
      IF(KG.NE.L) GO TO 48
      IF(LA(LV,L,J).NE.0.AND.LA(LV,L,J).LT.T) GO TO 48
    5 JGPR=JGPR+IT(J,I)
   48 CONTINUE
   44 CONTINUE
      GREAT(L)=JGPR
      GO TO 43
   26 NT=LA(LV,L,1)
      DO 45 J=2,JOBS
      IF(LA(LV,L,J).GT.NT) NT=LA(LV,L,J)
   45 CONTINUE
      GREAT(L)=NT
   43 CONTINUE
```

```
      IF(IPRINT.EQ.0) GO TO 52
      WRITE(3,900) (GREAT(MQ),MQ=1,MACH)
  900 FORMAT(1H ,4F8.1)
   52 ILB(LV,N1)=GREAT(1)
      DO 46 I=2,MACH
      IF(ILB(LV,N1).GE.GREAT(I)) GO TO 46
      ILB(LV,N1)=GREAT(I)
   46 CONTINUE
      IF(IPRINT.EQ.0) GO TO 66
      WRITE (3,59) ILB(LV,N1),N1
   66 JXP2=ILB(LV,N1)
      IF(JXP2-JXP1)61,61,62
   61 ILB(LV,N1)=JXP1
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
      GO TO 20
   62 ILB(LV,N1)=JXP2
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BOUND3 (T,K,LV)
C
C.....................................................................
C     THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
C     JOB-BASED BOUND III, SUGGESTED BY CONWAY ET AL.
C.....................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      DIMENSION GREAT(10)
      IF(IPRINT.EQ.0) GO TO 60
      WRITE (3,86)
   86 FORMAT(1H ,10X,'  ILB   N')
   60 NLV=N(LV)
      DO 20 N1=1,NLV
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      DO 10 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
      IF(N2.EQ.N1) GREAT(N2)=JCT(J2,MACH)
      IF(N2.EQ.N1) GO TO 10
      DIF=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
      GREAT(N2)=JCT(J2,MACH)+DIF
   10 CONTINUE
      ILB(LV,N1)=GREAT(1)
      DO 15 I=2,NLV
      IF(ILB(LV,N1).GE.GREAT(I)) GO TO 15
      ILB(LV,N1)=GREAT(I)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,50) ILB(LV,N1),N1
   50 FORMAT(1H ,10X,2I5)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BOUND4
C
C..................................................................
C     THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
C     JOB-BASED BOUND IV , SUGGESTED BY'BROOKS AND WHITE'
C..................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      DIMENSION GREAT(10),GRAT(10),I(10),GRT(10),SMALL(10),GRETE(10)
      IF(IPRINT.EQ.0) GO TO 999
      WRITE (3,86)
   86 FORMAT(1H0,10X,'  ILB   N')
  999 NLV=N(LV)
      DO 20 N1=1,NLV
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      DO 10 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
      IF(N2.EQ.N1) GREAT(N2)=JCT(J2,MACH-1)
      IF(N2.EQ.N1) GO TO 10
      DIF=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
      GREAT(N2)=JCT(J2,MACH-1)+DIF
   10 CONTINUE
      N2=NLV
      DO 30 J=1,JOBS
      NN=1
    9 IF(JJ(LV,NN).EQ.J) GO TO 30
      NN=NN+1
      IF(NN.GT.NLV) GO TO 40
      GO TO 9
   40 N2=N2+1
      GREAT(N2)=JCT(J,MACH-1)
      JJ(LV,N2)=J
   30 CONTINUE
      DO 600 KX=1,MACH
      L=0
      DO 502 NN=1,N2
      J3=JJ(LV,NN)
      M3=MM(J3,MACH)
      IF(M3.NE.KX) GO TO 502
      L=L+1
      GRAT(L)=GREAT(NN)
      I(L)=NN
      MR=NN
  502 CONTINUE
      IF(L.EQ.0) GO TO 600
    - IF(L.GT.1) GO TO 510
      LL=0
      DO 210 IK=1,N2
      J4=JJ(LV,IK)
      IF(MM(J4,MACH-1).NE.KX) GO TO 210
      LL=LL+1
      IF(LL.EQ.1) JGRT=GREAT(IK)
  210 CONTINUE
      IF(LL.EQ.0) GO TO 580
```

```
      J5=JJ(LV,MR)
      IF(JGRT.GE.GREAT(MR)) GO TO 220
      GRT(MR)=GREAT(MR)+IT(J5,MACH)
      GO TO 600
  220 GRT(MR)=JGRT+IT(J5,MACH)
      GO TO 600
  510 DO 700 M=1,L
      SMALL(M)=GRAT(1)
      MN=I(1)
      KP=1
      DO 690 KS=2,L
      IF(GRAT(KS).GE.SMALL(M)) GO TO 690
      SMALL(M)=GRAT(KS)
      MN=I(KS)
      KP=KS
  690 CONTINUE
      GRAT(KP)=9999
      J6=JJ(LV,MN)
      IF(M.GT.1) GO TO 691
      IF(M.EQ.1) GRT(MN)=SMALL(M)+IT(J6,MACH)
      IF(M.EQ.1) GRETE(M)=SMALL(M)+IT(J6,MACH)
      GO TO 700
  691 IF(SMALL(M).LE.GRETE(M-1)) GRT(MN)=GRETE(M-1)+IT(J6,MACH)
      IF(SMALL(M).LE.GRETE(M-1)) GRETE(M)=GRETE(M-1)+IT(J6,MACH)
      IF(SMALL(M).GT.GRETE(M-1)) GRT(MN)=SMALL(M)+IT(J6,MACH)
      IF(SMALL(M).GT.GRETE(M-1)) GRETE(M)=SMALL(M)+IT(J6,MACH)
  700 CONTINUE
      GO TO 600
  580 J7=JJ(LV,MR)
      GRT(MR)=GREAT(MR)+IT(J7,MACH)
  600 CONTINUE
      ILB(LV,N1)=GRT(1)
      DO 15 IX=2,N2
      IF(ILB(LV,N1).GE.GRT(IX)) GO TO 15
      ILB(LV,N1)=GRT(IX)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,50) ILB(LV,N1),N1
   50 FORMAT(1H0,10X,2I5)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE BOUND5 (T,K,LV)
C
C.......................................................................
C     THIS SUBROUTINE COMPUTES THE LOWER BOUND USING THE
C     MACHINE-BASED BOUND LB V,  SUGGESTED BY CONWAY ET AL.
C.......................................................................
C
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      DIMENSION GREAT(10),GRAT(10),GRT(10),DIF(10)
      IF(IPRINT.EQ.0) GO TO 60
      WRITE (3,86)
   86 FORMAT(1H0,10X,'  ILB   N')
   60 NLV=N(LV)
      DO 20 N1=1,NLV
      J1=JJ(LV,N1)
      I1=IOP(LV,J1)
      M1=MM(J1,I1)
      DO 11 J=1,JOBS
   11 DIF(J)=0.
      DO 10 N2=1,NLV
      J2=JJ(LV,N2)
      I2=IOP(LV,J2)
      IF(N2.EQ.N1) DIF(J2)=0
      IF(N2.EQ.N1) GO TO 10
      DIF(J2)=JCT(J1,I1)+IT(J2,I2)-JCT(J2,I2)
   10 CONTINUE
      JGRT=JCT(J1,I1)
      DO 9 J=1,JOBS
      IF(J1.EQ.J) GO TO 9
      DO 8 I=1,MACH
      KC=MM(J,I)
      IF(KC.NE.K) GO TO 8
      IF(LA(LV,K,J).NE.0.AND.LA(LV,K,J).LT.T) GO TO 8
    6 JGRT=JGRT+IT(J,I)
    8 CONTINUE
    9 CONTINUE
      GREAT(K)=JGRT
      DO 40 L=1,MACH
      LL=0
      IF(K.EQ.L) GO TO 40
      DO 39 J=1,JOBS
      DO 38 I=1,MACH
      KE=MM(J,I)
      IF(KE.NE.L) GO TO 38
      IP=I-1
      IF(IP.EQ.0) GO TO 35
      IF(JCT(J,I).LT.T) GO TO 38
      LL=LL+1
      GRAT(LL)=JCT(J,IP)+DIF(J)
      GO TO 38
   35 JGPP=JCT(J,I)
      IF(JGPP.LT.T) GO TO 38
      LL=LL+1
      GRAT(LL)=0.
   38 CONTINUE
   39 CONTINUE
```

```
      IF(LL-1)26,28,29
   29 JGP=GRAT(1)
      DO 27 LR=2,LL
      IF(GRAT(LR).LT.JGP) JGP=GRAT(LR)
   27 CONTINUE
      JGPR=JGP
      GO TO 7
   28 JGPR=GRAT(1)
    7 DO 50 J=1,JOBS
      DO 48 I=1,MACH
      KG=MM(J,I)
      IF(KG.NE.L) GO TO 48
      IF(LA(LV,L,J).NE.0.AND.LA(LV,L,J).LT.T) GO TO 48
    5 JGPR=JGPR+IT(J,I)
   48 CONTINUE
   50 CONTINUE
      GREAT(L)=JGPR
      GO TO 40
   26 NT=LA(LV,L,1)
      DO 30 J=2,JOBS
      IF(LA(LV,L,J).GT.NT) NT=LA(LV,L,J)
   30 CONTINUE
      GREAT(L)=NT
   40 CONTINUE
      IF(IPRINT.EQ.0) GO TO 12
      WRITE(3,900) (GREAT(MQ),MQ=1,MACH)
  900 FORMAT(1H ,4F8.1)
   12 ILB(LV,N1)=GREAT(1)
      DO 15 I=2,MACH
      IF(ILB(LV,N1).GE.GREAT(I)) GO TO 15
      ILB(LV,N1)=GREAT(I)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 20
      WRITE (3,59) ILB(LV,N1),N1
   59 FORMAT(1H0,10X,2I5)
   20 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SMLILB(LV)
      COMMON IT(15,15),MACH,MM(15,15),JCT(15,15),LA(90,15,15)
      COMMON IOP(90,15),JJ(90,15),ILB(90,15),JOBS,ISTMIN
      COMMON N(90),NILB(90),JACTIV(90),IPRINT,IB
      NLV=N(LV)
  120 DO 300 NL=1,NLV
      IF(JJ(LV,NL).NE.JACTIV(LV)) GO TO 300
      NK=NL
  300 CONTINUE
      ISWTCH=0
      NLV=N(LV)
      DO 10 NL=1,NLV
      IF(JJ(LV,NL).EQ.JACTIV(LV)) GO TO 10
      IF(ISTMIN.LE.ILB(LV,NL)) GO TO 10
      IF(ILB(LV,NL)-NILB(LV))10,8,9
    8 IF(NL.LT.NK) GO TO 10
    9 ISWTCH=ISWTCH+1
      IF(ISWTCH-1)11,11,12
   11 NT=ILB(LV,NL)
      NTL=NL
      GO TO 10
   12 IF(ILB(LV,NL).GE.NT) GO TO 10
      NT=ILB(LV,NL)
      NTL=NL
   10 CONTINUE
      NILB(LV)=NT
      JACTIV(LV)=JJ(LV,NTL)
   21 LV=LV+1
      DO 20 M=1,MACH
      DO 20 J=1,JOBS
   20 LA(LV,M,J)=LA(LV-1,M,J)
      DO 30 J=1,JOBS
   30 IOP(LV,J)=IOP(LV-1,J)
      IA=IOP(LV,JACTIV(LV-1))
      L=N(LV-1)
      K=MM(JACTIV(LV-1),IA)
      JCT(JACTIV(LV-1),IA)=LA(LV-1,K,JACTIV(LV-1))
      DO 15 NL=1,L
      IF(JJ(LV-1,NL).EQ.JACTIV(LV-1)) GO TO 15
      J1=JJ(LV-1,NL)
      I1=IOP(LV,J1)
      JCT(J1,I1)=JCT(JACTIV(LV-1),IA)+IT(J1,I1)
      M=MM(J1,I1)
      LA(LV,M,J1)=JCT(J1,I1)
      IK=I1+1
      IF(IK.GT.MACH) GO TO 15
      DO 14 I=IK,MACH
   14 JCT(J1,I)=JCT(J1,I-1)+IT(J1,I)
   15 CONTINUE
      IF(IPRINT.EQ.0) GO TO 35
      WRITE (3,88)
   88 FORMAT(1H0,10X,      'COMPLETION TIME MATRIX*')
      DO 13 J=1,JOBS
   13 WRITE (3,4) (JCT(J,I),I=1,MACH)
    4 FORMAT(1H ,15X,12I4)
   35 RETURN
      END
```

```
   PROBLEM NUMBER =    1
  PROCESSING TIME MATRIX
    21   20   25   19   27
     1   20   26   24    5
    22   24   12    4   16
    25   30   22   16   15
    23   10    2   12   30
     5   11   18   21   29
     5   28   16    1    3
    13   22   16   15   19
  MACHINE ORDERING MATRIX
     2    3    5    1    4
     3    5    4    2    1
     3    5    4    2    1
     3    5    2    4    1
     5    3    1    2    4
     5    2    1    3    4
     3    5    4    1    2
     2    1    4    3    5

   BOUNDING PROCEDURE  1
A SCLUTION      218
NO CF CONFLICT LEVELS FOR SOLN     30
A SOLUTION      217
NO OF CONFLICT LEVELS FOR SOLN     26
A SCLUTION      214
NO OF CONFLICT LEVELS FOR SOLN     29
```

A BRANCH-AND-BOUND ALGORITHM

FOR JOB-SHOP PROBLEMS

by

SANGAYYA RACHAYYA HIREMATH

B.E. (Mech.), Karnatak University

Dharwar, Mysore-State, India, 1967

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Industrial Engineering

KANSAS STATE UNIVERSITY

Manhattan, Kansas

1970

The scheduling problem with which this thesis is concerned consists of determining the sequence of J jobs to be processed on M machines so that the schedule time is minimized.

In this thesis, a branch-and-bound technique for job-shop problems is developed. This technique generates an optimal solution after the generation of only a small subset of possible sequences. The basic concepts of this approach consist of the branching, bounding and back-tracking processes. The branching process generates a set of new nodes from a node at the preceeding level. The bounding process helps select a particular node at a level for further branching and thus, makes it possible to achieve a reduction in the generation of nodes at each level. The backtracking process guarantees an optimal solution.

In this thesis, two new lower bounds, referred to as composite-based bounds LB I and LB II, are developed. Three other existing lower bounds, referred to as bounding procedures LB III, LB IV and LB V, are analyzed in a mathematical form and rigorous notation for comparison purposes. A considerable number of experiments has been conducted on IBM 360/50 computer. The results are obtained in terms of the number of nodes explored and the computational time required to obtain the optimal solution and the efficiency of solution obtained without back-tracking.

The various lower bounds are compared with the help of the above results. It is found that the number of nodes explored and the computational time to obtain the optimal solution increase rapidly with the increase in the size of the problem. In general, the performance of the composite-based bounds LB I and LB II is better than any of the bounding procedures LB III, LB IV and LB V.