DEVELOPMENT OF A NETWORK ALGORITHM AND ITS APPLICATION TO COMBINATORIAL PROBLEMS

by

ROBERT GARY PARKER

B. S., Kansas State University, 1968

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 would like to express a sincere debt of gratitude to my Major Professor, Dr. Said Ashour, for his guidance in the preparation of this work. It was he who not only suggested this piece of research but, more importantly, instigated the writer's great interest in scheduling theory.

Also, I would like to thank Dr. Frank A. Tillman, Head of the Department of Industrial Engineering; Dr. L. E. Grosh, Department of Industrial Engineering; and Dr. N. D. Eckhoff, Department of Nuclear Engineering, for their assistance.

Finally, I want to express thanks to Mrs. F. Rod Harris for her invaluable assistance in typing this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER I

Introduction

As problems in business and industry become more and more complex, fields such as operations research and management science have been called upon for their ability to solve or at least lessen these problems. The diversity accompanying such problems might be expected; however, their magnitude is sometimes overwhelming. For example, consider the seemingly simple task of sequencing five jobs on say three machines, so as to optimize some measure of performance such as schedule time. If one would attempt to evaluate every possible sequence in this problem, a total of $(5!)^3$ sequences would have to be evaluated. Obviously, when the number of jobs and machines increases, the magnitude of this type of problem increases tremendously, for the number of possible sequences can be expressed in general as $(J!)^M$, where J is the number of jobs and M represents the number of machines.

While the above illustration represents only a specific type of problem, namely that of scheduling theory, it more importantly, gives rise to a much wider range of emphasis and that is the combinatorial problem in general. Many techniques have been suggested for use in combinatorial problems and naturally, some are more powerful than others. Of course, any technique that reduces the computation involved in obtaining solutions, is much more desirable than simple enumeration.

Nevertheless, this work presents a discussion of a particular technique which can be used to solve various combinatorial problems. The first chapter is organized into three sections. The first involves a general discussion of the combinatorial problem and some of its characteristics. The second section considers a brief historical background pertaining specifically to the work done in this thesis, and the last section deals with proposed research.

1.1 The Combinatorial Problem

Rigorous definitions of the combinatorial problem, as has been suggested in literature of the field, are very difficult to formulate. In general, however, such problems concern themselves with the study of arrangements or groupings of finite numbers of elements into sets. Such arrangements are generally constrained by boundary restrictions imposed upon the problem. To illustrate this situation, let us return to the J x M scheduling problem again. If we consider the total number of sequences possible, to be one large set, we can further decompose such a set into smaller subsets with respect to two considerations.

The first consideration involves constraints on the problem, in the form of machine orderings. Such orderings are the result of existing technological requirements. Consequently, any sequence from the total of all sequences that violates such requirements would be non-feasible and can be removed from consideration as a possible solution.

The second major consideration is that of optimality. Obviously, there are solutions to the problem that are superior to other solutions, considering, of course, a particular measure of performance. Therefore, the set of solutions remaining from the entire set after removal of those that are non-feasible can be further decomposed into two subsets; those that are optimal and those that are non-optimal. This entire decomposition of solutions can be illustrated in general by Fig. 1.1.

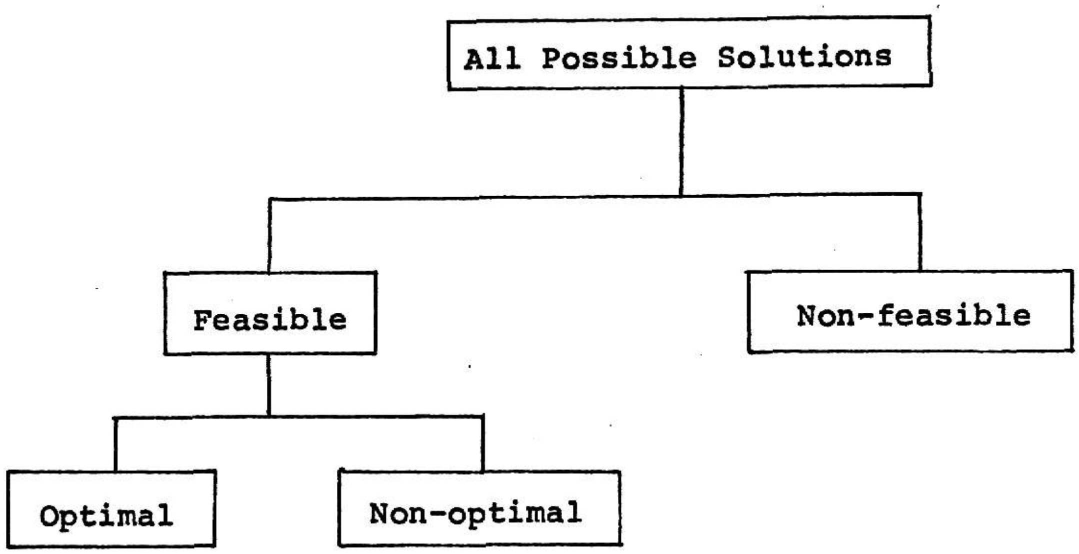


Fig. 1.1. Decomposition of possible solutions to the combinatorial problem.

Naturally, there are many problems that can be classified as combinatorial. Use has been made thus far of only one type of combinatorial problem, that of scheduling; however, other types of problems such as the traveling salesman, delivery, line balancing, and critical path, to mention a few, can be so classified.

1.2. Historical Background

In 1959, Giffler [8], introduced the concept of schedule algebra and with it, described techniques for solving production scheduling problems. His work was updated in 1962 when he presented a computational technique known informally as the schedule algebra algorithm, [5], which, of course, was based upon the use of the schedule algebra operators.

Since 1963, however, there seems to have been very little amplification of Giffler's work. Appearances in the literature occurred [6, 7, 9, 10], but such publications have basically been representations by Giffler of the original work. Consequently, since the schedule algebra algorithm was first introduced, there has been little continuation of its concept. The entire literature survey can be presented as shown below:

Year	Reference	Description
1959	[8]	A demonstration of the use of conventional matrix algebra in the solution of explosion problems, as well as the development and demonstration of a schedule algebra used to solve scheduling problems.
1959	[9]	A presentation of algorithms which can be used in solution of the general scheduling problem.
1959	[10]	An introduction to the concept of active schedules; however, basically, this presentation is equivalent to that in [9].

1961	[7]	A summary of various scheduling theories and discussion of the schedule algebra and its application to explosion and scheduling problems. This work in general is very similar to that of [5].
1962	[5]	A formulation of the schedule algebra operators and a formal presentation of the schedule algebra algorithm.
1968	[6]	A summary of the current status of schedule algebra, its origin, application, and its motivation. In general, this work embodies most of the concepts presented in Giffler's earlier work.

In trying to trace the work which may have led to the schedule algebra algorithm, it was found, in another work by Giffler in 1959, that a linear algorithm was presented [9]. It was believed that this algorithm was linked to the schedule algebra algorithm. While investigating such a relationship, another technique, which was unnamed at the time, was mentioned in the same literature. This technique did, indeed, bear a great deal of resemblance to the current schedule algebra algorithm. Furthermore, it was found that through implementation of some of the characteristics of the schedule algebra algorithm and simple experimentation, the network algorithm could be constructed. Referral to the technique as a network algorithm arose from its direct applicability to problems which can be formulated into networks.

At first, the use of this network algorithm was confined to the scheduling problem. However, its application to other network problems became evident and as such, the entire concern of this work became shifted from what was originally an analysis of Giffler's schedule algebra algorithm, to a formalization and application of the network algorithm.

1.3. Proposed Research

The motivation for beginning this research was the result of two primary factors. The first involved the fact that the original schedule algebra was developed in 1959 and, until the present time, has not been investigated further. This situation can be verified by examining the literature summary presented earlier. All work since 1959 seems to have been confined only to the originator of the technique.

The second consideration, which is an outgrowth of the first, pertains to the fact that there has been no new presentation made in the literature, with respect to the schedule algebra. While there have been several publications by a single individual, there was little diversity among such presentations; hence, it became evident that there was little or no new development of the schedule algebra concept.

The apparent scarcity of work dealing with the schedule algebra concept, instigated research into three basic areas. They are, (1) the development of a network algorithm which is based upon the schedule algebra operators and which embodies a criteria referred to as a lower bound to improve the solution,

(2) the extension of the application of this network algorithm to other combinatorial problems, and (3) the investigation into the computational experience which results when the network algorithm is applied to problems of varying dimensions.

CHAPTER II

Development of a Network Approach

At the outset, the main emphasis for this research was placed upon the use of the schedule algebra algorithm as presented in Appendix B. However, as the study progressed, this consideration was slowly modified until the resultant network approach emerged. Consequently, the scope of this work became centered around this modification and its applicability to the combinatorial problem in general. Nevertheless, this chapter will deal with the development and basic concepts of the network approach. Its application will be demonstrated with a sample problem, and, finally, a computational algorithm will be presented in a formal fashion. A general nature, with respect to the concepts of the approach, is maintained in order to allow applicability to other problem formulations. Such further application will be discussed in Chapter III.

2.1 Basic Concepts

The network technique is a systematic approach which searches for a solution among a subset of feasible sequences. The basic concepts of this network approach involve: (1) the representation of the problem in a network which, in turn, is depicted in a precedence matrix, (2) the manipulation of the precedence matrix based on the star algebra operators, and (3) the evaluation of the resulting sequence to obtain the corresponding schedule time.

A network can be described as consisting of a set of nodes and branches which connect various pairs of nodes. If these branches are specifically oriented, they are then said to be directed. For example, the network depicted in Fig. 2.1 is directed because every connecting branch is oriented in a specific direction.

The scheduling problem can be formulated into a network or directed linear graph by making use of the precedence relationships that are inherent in the machine orderings. As defined earlier, the scheduling problem can be represented by machine ordering and processing time matrices. This representation can be shown with the aid of the following sample problem:

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

By considering the second row of the machine ordering matrix, we can interpret the following: job 2 is processed on machine 1 first, machine 3 second, and machine 2 last.

Using the directed linear graph, the machine orderings of jobs 1, 2, 3, and 4, can be represented as shown in Fig. 2.2. These relationships are nothing more than those of direct precedence. Further, they can be considered as partial orderings.

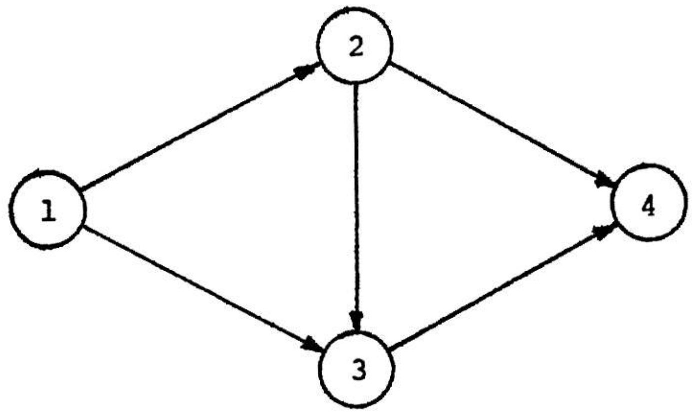


Fig. 2.1. A directed network

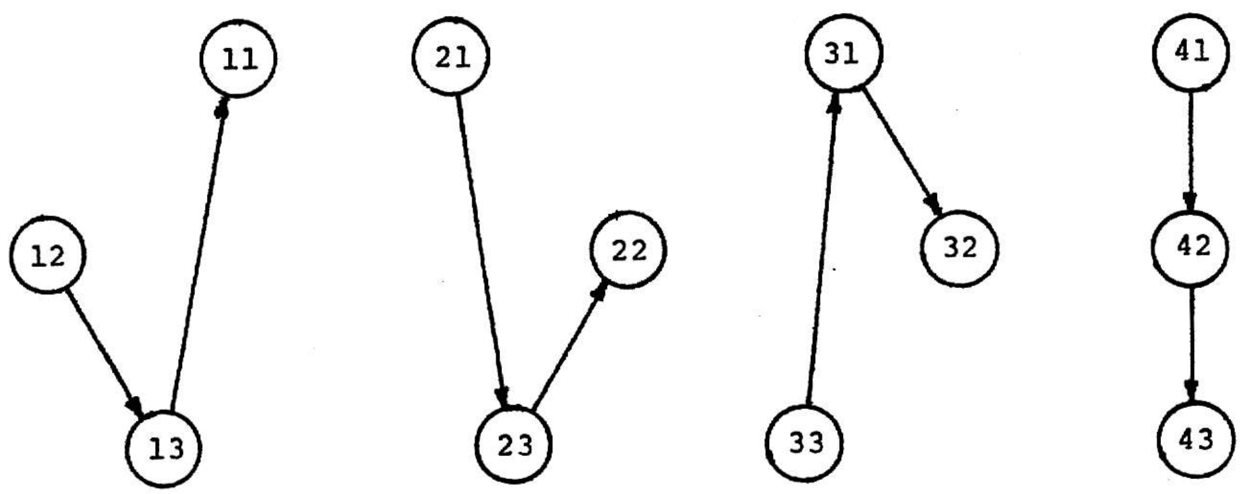


Fig. 2.2. Directed linear graph depicting machine orderings.

In fact, this re-definition will be used synonymously with machine orderings throughout this discussion. Nevertheless, any scheduling problem, as previously stated in mathematical form, can be depicted with the linear graph representation as illustrated in Fig. 2.2.

Once the linear graphs have been constructed, the search for a solution to the problem can commence. Such a solution involves determining the sequence of the jobs on each machine. That is, the task becomes one of finding a job sequencing matrix, S such that

$$S = \begin{bmatrix} 41 & 31 & 21 & 11 \\ 12 & 42 & 32 & 22 \\ 33 & 13 & 43 & 23 \end{bmatrix},$$

where machines 1, 2, and 3 perform the four jobs in the sequences { 4 3 2 1 }, { 1 4 3 2 }, and { 3 1 4 2 }, respectively.

Figure 2.3 shows a directed network which represents the above sequence. Note that this sequence is consistent with the machine ordering graphs.

In the scheduling problem, the network analysis is made with respect to two factors. The first is the partial ordering that is inherent in the statement of the problem. For example, the operation of reaming a hole could not precede the operation of drilling the hole. Technologically, it is not possible.

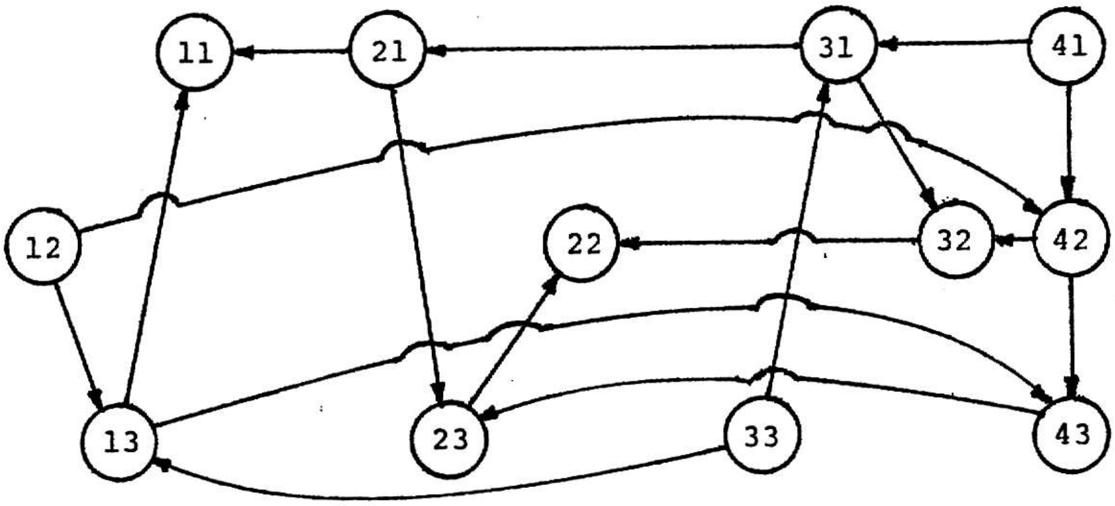


Fig. 2.3. Network depicting feasible sequence, S.

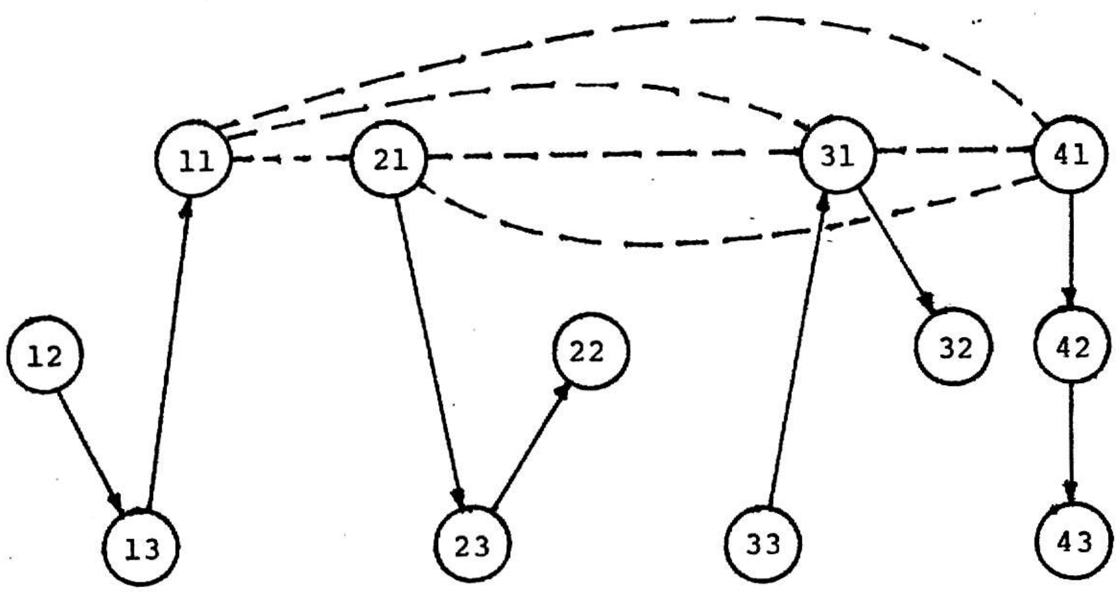


Fig. 2.4. Possible direct precedence relationships.

Such an ordering of precedence relationships are shown in the machine ordering matrix which was illustrated earlier. These partial orderings must be maintained in order to obtain a feasible solution. That is, any feasible sequence must be consistent with the partial orderings. Naturally, any sequence that does not maintain these partial orderings will be non-feasible.

The second factor pertains to the sequences in which jobs are processed on each of the machines. It is this determination of job sequencings that is the primary consideration in the scheduling problem. Any evaluation of various solutions is only an evaluation of the sequence in which jobs can be processed on each machine.

By considering Fig. 2.2 again, let us reconstruct the four linear graphs, or partial orderings, with the following addition. The nodes representing operations on machine 1 are connected by broken lines which, as can be seen, are not oriented in any direction. These branches are normally repeated for the nodes representing operations on the other two machines, but in order to avoid congestion, they have been omitted in Fig. 2.4. Nonetheless, the logic of the broken line branches is very critical to the discussion of the network formulation as well as to the construction of the precedence matrix which will be discussed shortly.

Each branch implies the existence of a possible direct precedence relationship. For example, operation or node (11), in Fig. 2.4, may directly precede operation or node (21). Conversely, node (21) may directly precede node (11). The exact precedence at this point is, of course, unknown. The significance of the broken line is simply to illustrate the unknown direction of precedence. However, it should be understood that certain broken lines will eventually become solid directed line segments which will depict a final direct precedence relationship. That is, when all broken lines are either made solid and directed or are deleted, a feasible sequence has been attained. Consequently, the construction in Fig. 2.4 cannot be classified as a network in its present state. This is only logical, because by virtue of the broken lines, not one, but many possible networks are represented. It is the task of the technique employed to determine one network from this population of many.

Once the problem has been formulated as shown in Fig. 2.4, it can be reformulated into a precedence matrix. It is this precedence matrix which is the basis for the network algorithm. The matrix is always square and its size is determined from the number of jobs and machines, such that the number of rows and columns is JM. The matrix is partitioned into M machine blocks, each of which has J rows and J columns. For example, a (4 x 3) scheduling problem can be represented by a precedence matrix of size 12 x 12. This matrix consists of 3 machine blocks. Machine block 1 includes all 4 jobs on machine 1, block 2; the 4 jobs on machine 2, and block 3 contains 4 jobs on machine 3.

Entries are made in the precedence matrix, Q in a conventional manner. That is, an entry $q_{(j\ m_\ell,\ j\ m_s)}$ is made with respect to its row first and then its column. The value of each entry is determined such that

$$q_{(j\ m_\ell,\ j\ m_s)} = \begin{cases} t_{(j\ m_\ell)}, & \text{if } (j\ m_\ell) << (j\ m_s) \\ xt_{(j\ m_\ell)}, & \text{if } (j\ m_\ell) << (j\ m_s) \text{ is possible,} \\ 0, & \text{otherwise} \end{cases}$$

where (jm_ℓ) indicates operation of job j on machine m_ℓ.

Once the precedence matrix has been constructed, the task of manipulation can begin. The very basic concept behind the manipulation of the precedence matrix is one of a step by step entry of nodes into solution. That is, by entering operations in a systematic manner, to be explained shortly, the investigator is, in essence, moving through the network. When this step by step process is completed, every node in the network will have been entered and, of course, some sequence will result.

Entries are made on a machine block basis. At each iteration, each machine block is checked for operations to enter next. This checking procedure involves scanning each column of the matrix in search of those nodes that are, in fact, ready for next entry into the solution. A node or operation can enter the solution or more generally, is a candidate for entry, if all nodes that directly precede it in the partial ordering, have already been entered. When such a condition for entry exists, the column represented by the node in question is said to be null or potentially null. Nonetheless, this concept of entry is valid, of course, because any entry of some node $(j\ m_\ell)$ before a preceding node $(j\ m_s)$ has been entered, would be inconsistent with the partial ordering and would result in a non-feasible sequence.

If there is more than one candidate for entry in a machine block, a lower bound is applied and the conflict is resolved in favor of the operation which has the least lower bound. Nevertheless, when a node for next entry has been determined for each block (if no node can next enter from a particular block, the block is unchanged at that iteration) the matrix is updated. This involves updating the entries in the column of the corresponding node such that

$$xq_{(j\ m_\ell,\ j\ m_s)} = \begin{cases} 0, & \text{if } (j\ m_\ell) \text{ has not been previously entered,} \\ q_{(j\ m_\ell,\ j\ m_s)}, & \text{if } (j\ m_\ell) \text{ has been entered.} \end{cases}$$

If the entry $q_{(j\ m_\ell,\ j\ m_s)}$ is made, then all the remaining entries in the same row as the updated element, become zero. This procedure for entry and the corresponding update of each column follows from the nature of the problem, in general.

By noting that the only elements in a diagonal machine block that are not zero are those of the form $xq_{(j\ m_\ell,\ j\ m_s)}$, and further, recalling that the real problem at hand is to determine a sequence of jobs on each machine, it is completely logical that the procedure described above be carried out.

Naturally, if a node has been selected to next enter or, with reference to the sequencing problem, if a job has been selected to be processed next, any node entered previously would precede it. Furthermore, all nodes not yet entered would not precede it, but would, in fact, be preceded by the node now being entered. Consequently, in the former case, the update to relevant entries in the column represented by the entered node would involve changing an entry of the form $xq_{(j\ m_\ell,\ j\ m_s)}$ to $q_{(j\ m_\ell,\ j\ m_s)}$, while in the latter case, the change would be to zero. This concept will be illustrated by a sample problem.

When all nodes have been entered, the precedence matrix will contain only positive scalar entries or zeros. All conflicts on all machines will have been resolved, and some feasible sequence been obtained. However, to determine the earliest starting times, and, hence, the schedule time for the sequence, the concept of a starting time vector is employed.

This vector consists of JM elements. As an illustration, a starting vector for the (4 x 3) problem would be a vector of 12 elements. Initially, entries in the vector are either iota, ι, or zero. If the earliest possible starting time of a node is known, the entry is ι; otherwise, the entry is zero. Specifically, all initial nodes in the partial orderings will command entries, in the initial starting vector, of ι, while all other entries will be zero.

Once the initial starting vector has been constructed, a series of multiplications follow. The starting vector is multiplied by the final precedence matrix over and over until there is no change in succeeding vectors. Prior to each new multiplication, the vector is updated by being added to the vector of the preceding multiplication. Of course, the operations of multiplication and addition are those of star algebra discussed in Appendix A. Finally, the resultant starting vector represents the earliest starting times of all nodes, consistent, of course, with the sequence determined by the final precedence matrix. The schedule time can be easily computed from this final-starting time vector.

2.2 Sample Problem

Let us consider the following machine ordering and processing time matrices:

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

Step 1. Construct the initial starting vector, T^{o}, such that

$$T^{o} = [\; 0 \;\; \iota \;\; 0 \;\; \iota \;\; \iota \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; \iota \;\; 0 \;],$$

Table 2.1. Initial Precedence Matrix, Q°.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x3	x3	x3	0	0	0	0	0	0	0	0
(21)	x8	0	x8	x8	0	0	0	0	0	8	0	0
(31)	x3	x3	0	x3	0	0	3	0	0	0	0	0
(41)	x7	x7	x7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	x4	x4	x4	4	0	0	0
(22)	0	0	0	0	x5	0	x5	x5	0	0	0	0
(32)	0	0	0	0	x9	x9	0	x9	0	0	0	0
(42)	0	0	0	0	x6	x6	x6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	x2	x2	x2
(23)	0	0	0	0	0	4	0	0	x4	0	x4	x4
(33)	0	0	6	0	0	0	0	0	x6	x6	0	x6
(43)	0	0	0	0	0	0	0	0	x2	x2	x2	0

where only the earliest starting times of nodes (21), (41), (12), and (33) are known and so signified by the entries of ι.

Step 2. Construct the precedence matrix, Q° from the partial orderings and possible direct precedence relationships as illustrated in Fig. 2.4.

Table 2.2. Intermediate Precedence Matrix, Q^1.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x3	x3	0	0	0	0	0	0	0	0	0
(21)	x8	0	x8	0	0	0	0	0	0	8	0	0
(31)	x3	x3	0	0	0	0	3	0	0	0	0	0
(41)	x7	x7	x7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	x4	x4	x4	4	0	0	0
(22)	0	0	0	0	0	0	x5	x5	0	0	0	0
(32)	0	0	0	0	0	x9	0	x9	0	0	0	0
(42)	0	0	0	0	0	x6	x6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	x2	0	x2
(23)	0	0	0	0	0	4	0	0	x4	0	0	x4
(33)	0	0	6	0	0	0	0	0	x6	x6	0	x6
(43)	0	0	0	0	0	0	0	0	x2	x2	0	0

Step 3. Check for nodes to enter. Columns (33), (12), (21), and (41) are potentially null, and thus, they can be marked. The conflict between jobs 2 and 4 on machine 1, is resolved in favor of job 4, based on a composite lower bound which is described in Appendix C. Upon application of the lower bound, it is found

Table 2.3. Intermediate Precedence Matrix, Q^2

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x3	0	0	0	0	0	0	0	0	0	0
(21)	x8	0	0	0	0	0	0	0	0	8	0	0
(31)	x3	x3	0	0	0	0	3	0	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	x5	0	0	0	0	0
(32)	0	0	0	0	0	x9	0	0	0	0	0	0
(42)	0	0	0	0	0	x6	x6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	x2	0	x2
(23)	0	0	0	0	0	4	0	0	0	0	0	x4
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	x2	0	0

that the bound for job 4 is lower than that of job 3; consequently, job 4 is selected to next start.

Step 4. Update the precedence matrix by entering the latest marked nodes. The three columns (33), (12), and (41) are made null, and the matrix Q^0 is updated to matrix Q^1.

Table 2.4. Intermediate Precedence Matrix, Q^3.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	0	0	0	0	0	0	0	0	0	0	0
(21)	x8	0	0	0	0	0	0	0	0	8	0	0
(31)	0	3	0	0	0	0	3	0	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	0	0	0	0	0	0
(32)	0	0	0	0	0	x9	0	0	0	0	0	0
(42)	0	0	0	0	0	0	6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	0	0	2
(23)	0	0	0	0	0	4	0	0	0	0	0	0
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	x2	0	0

Step 5. Repeat step 3 by checking for three new nodes to enter. From the updated matrix Q^1, we can mark columns (13), (42), (31), and (21). Once again, we see the existence of a tie in machine block 1. Both nodes (31) and (21) are potentially null; and consequently, we shall apply the composite bound and

Table 2.5. Final Precedence Matrix, Q^4.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	0	0	0	0	0	0	0	0	0	0	0
(21)	8	0	0	0	0	0	0	0	0	8	0	0
(31)	0	3	0	0	0	0	3	0	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	0	0	0	0	0	0
(32)	0	0	0	0	0	9	0	0	0	0	0	0
(42)	0	0	0	0	0	0	6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	0	0	2
(23)	0	0	0	0	0	4	0	0	0	0	0	0
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	2	0	0

resolve the tie in favor of job 3. This resolution is, made in favor of job 3 since its bound is lower than that of job 2. Nonetheless, columns (13), (42), and (31) are marked. The matrix Q^1 is now updated and becomes Q^2.

ILLEGIBLE DOCUMENT

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

THIS IS THE BEST COPY AVAILABLE

The next three columns selected to enter are (43), (32), and (21). It should be pointed out that there is again a tie in machine block 1. The tie, involving nodes (21) and (11), is broken in favor of (21) after application of the lower bound. After updating the above matrix with respect to these three nodes, the resulting matrix, Q^3, can be constructed.

The remaining three columns, (23), (22), and (11), are marked and matrix Q^3 can be updated to become Q^4. This resultant matrix Q^4 is the final matrix, for all nodes have been entered.

Step 6. Compute the earliest starting times of all nodes. Upon multiplying the initial starting vector T° by the final precedence matrix Q^4, the resultant vector is as follows:

$$[\ 8\ 0\ 7\ 0\ 0\ 0\ 0\ 7\ 6\ 8\ 0\ 0\ ].$$

When this vector is added to T°, the resultant is T^1, where

$$T^1 = [\ 8\ \iota\ 7\ \iota\ \iota\ 0\ 0\ 7\ 6\ 8\ \iota\ 0\ ].$$

By repeating this procedure until there is no change in succeeding T vectors, we find that we must compute a total of four T vectors. These vectors are:

$$T^2 = [\ 8\ 10\ 7\ \iota\ \iota\ 12\ 13\ 7\ 6\ 8\ \iota\ 13\ ],$$

$$T^3 = [\ 18\ 10\ 7\ \iota\ \iota\ 22\ 13\ 7\ 6\ 18\ \iota\ 13\ ],$$

and

$$T^4 = [\ 18\ 10\ 7\ \iota\ \iota\ 22\ 13\ 7\ 6\ 18\ \iota\ 13\ ].$$

Note that vector T^4 is identical to T^3.

Step 7. Calculate the final sequence and the schedule time. From the final T vector, we can compute the sequence and its corresponding schedule time. By ordering the jobs with respect to their starting times in each machine block of the final T vector, we have the following sequences on each machine:

machine 1: { 4 3 2 1 } ,

machine 2: { 1 4 3 2 } ,

machine 3: { 3 1 4 2 } .

The job sequencing matrix can be taken from the above ordering such that

$$S = \begin{bmatrix} 41 & 31 & 21 & 11 \\ 12 & 42 & 32 & 22 \\ 33 & 13 & 43 & 23 \end{bmatrix} .$$

If we locate the entries in each machine block with the highest starting times, we get (11), (22), and (23). Upon adding the processing times of each of these nodes to their respective start times, we get 21, 27, and 22 time units respectively. Consequently, the schedule time for the sequence just computed is 27. It should be noted that this is the optimal solution, since this problem was also solved by a branch-and-bound algorithm with backtracking [11]. Note that in breaking the ties differently, we could expect to obtain other solutions.

2.3 A Network Algorithm

Now that the basic concepts and a sample problem have been discussed with reference to the network approach, a formal step by step computational algorithm is presented below.

Step 1: Construct the initial starting vector, T°.

Form a vector with JM entries such that

$$\tau\ (j\ m_{\ell}) = \begin{cases} 1, & \text{for all } (j\ m_1), \\ 0, & \text{otherwise.} \end{cases}$$

Step 2: Construct the initial precedence matrix, Q°.

2.1 Partition a (JM x JM) matrix into M machine blocks.

2.2 Label the rows and columns of the matrix by the appropriate nodes.

2.3 Place the entries in the matrix such that

$$q_{(j\ m_{\ell}, j\ m_{s})} = \begin{cases} t_{(j\ m_{\ell})}, & \text{if } (j\ m_{\ell}) << (j\ m_{s}), \\ xt_{(j\ m_{\ell})}, & \text{if } (j\ m_{\ell}) << (j\ m_{s}) \text{ is possible}, \\ 0, & \text{otherwise.} \end{cases}$$

Step 3: Check for null or potentially null columns.

3.1 Within each machine block, mark the columns that are null or can be made null.

3.2 If there is more than one marked column in a machine block, break the tie by a particular bounding procedure.

Step 4: Update the precedence matrix.

At the intersection of a marked column and a marked row make the following change if possible:

$$xq_{(j\ m_{\ell},\ j\ m_{s})} = q_{(j\ m_{\ell},\ j\ m_{s})} ,$$

Update all other entries in that column and row such that

$$xq_{(j\ m_{\ell},\ j\ m_{s})} = 0 ,$$

then mark the corresponding row of the just updated column.

Step 5: Repeat steps 3 and 4 until there are no more entries in the precedence matrix which have x terms.

Step 6: Update the starting vector.

6.1 Multiply the final precedence matrix by the starting vector such that

$$T^{k'} = T^{k-1} \# Q,\ k = 1, 2, \ldots,$$

and add the starting vector to the resultant vector such that

$$T^{k} = T^{k-1} * T^{k'} .$$

6.2 Repeat step 6.1 until there is no change in succeeding starting vectors, or simply, until

$$T^k = T^{k-1} \quad .$$

Step 7: Find the sequence and the corresponding schedule time.

7.1 In each machine block of the final starting vector, order the jobs with respect to their start times.

7.2 Locate the element in each machine block of the final starting vector that has the latest start time.

7.3 Add the processing time to the starting time of each chosen element.

7.4 Select the operation which results in the greatest amount of time such that

$$T(S) = \max \left[\tau_{(j\ m_M)} + t_{(j\ m_M)} \right], \ j=1,\ 2,\ldots,J,$$

where T(S) is the schedule time for the sequence.

CHAPTER III

Applications to Combinatorial Problems.

In the preceding chapter the network algorithm was developed and demonstrated with a sample problem. The problem chosen in Chapter II was a typical job shop scheduling problem. In this chapter, however, three other types of problems have been used to illustrate that the network algorithm is not completely isolated in its use. These three problems are the traveling salesman, critical path, and explosion problems.

The format used in this chapter involves two specific divisions within each area of application. The first is the formulation of the problem for application of the network algorithm and the second involves a sample problem. It should be stressed that the main intent in this chapter is to point out the applicability of the network algorithm to at least some phase of other network problem solutions. In certain cases, complete solutions may be attainable; however, in other cases it may be necessary to use the network algorithm in combination with other techniques in order to obtain a specific solution.

3.1. The Traveling Salesman Problem

The traveling salesman problem can be stated as follows: A salesman has a given number of cities he must visit. Knowing the distances, costs, or say times between each pair of cities, the salesman's task is to select a route whereby he does, in fact, visit each city only once and in so doing, optimizes some measure of performance. It is, of course, understood that the salesman begins at some known point and ends his route at this same point. Nevertheless, this section deals with the application of the network algorithm to the traveling salesman problem and is organized as stated above.

Problem formulation. The traveling salesman problem can be considered as nothing more than a scheduling problem involving one machine. If such a scheduling problem makes use of say, setup time as a measure of performance, it becomes synonomous in nature to the traveling salesman problem. That is, a number of jobs are to be sequenced on a single machine so as to minimize total setup time between jobs, including the setup time between the final job in the sequence and the first job in the sequence. Nevertheless, in the construction of the precedence matrix for the traveling salesman problem, only one machine block is considered. In fact, the entire matrix is one machine block. Furthermore, remembering that in the scheduling problem, all conflicts within blocks were resolved by a bounding procedure, one can anticipate using some such criteria for resolving conflicts in this application. That is, since the precedence matrix for the traveling salesman problem is one large machine block, there will be a conflict at every iteration.

Necessary to the construction of the precedence matrix is the cost chart. Costs are used here, but it is understood that one could consider other criteria such as distance or time. Nevertheless, consider the asymmetrical cost chart proposed by Little, et al [12], and shown in Table 3.1. This cost chart as well as the network of Figure 3.1, plays a dual role. They are presented at this point for illustrative purposes only, with respect to the problem formulation; however, they also provide the sample problem that is solved in the second part of this section. Nevertheless, asymmetry implies the possibility of traveling from one node to another node, or conversely, from the latter node to the former. In conventional notation this can be represented such that

$$(i) << (j),$$

and

$$(j) << (i),$$

where (i) and (j) represent nodes in the network. This asymmetrical concept was evident in the relationships referred to previously as possible direct precedence. Just as before, these possible direct precedence relationships can be represented in the network by broken non-oriented branches. Such relationships can be seen from the network drawn in Figure 3.1.

Once the cost chart is known, the precedence matrix can be constructed. The manipulation of the precedence matrix is carried out in a manner consistent with the algorithm. The starting vector is, of course, employed in the same manner as before. The only entry in the initial starting vector not zero, will be that corresponding to the starting node in the network.

Table 3.1. A Cost Chart

	(1)	(2)	(3)	(4)	(5)	(6)
(1)	0	27	43	16	30	26
(2)	7	0	16	1	30	25
(3)	20	13	0	35	5	0
(4)	21	16	25	0	18	18
(5)	12	46	27	48	0	5
(6)	23	5	5	9	5	0

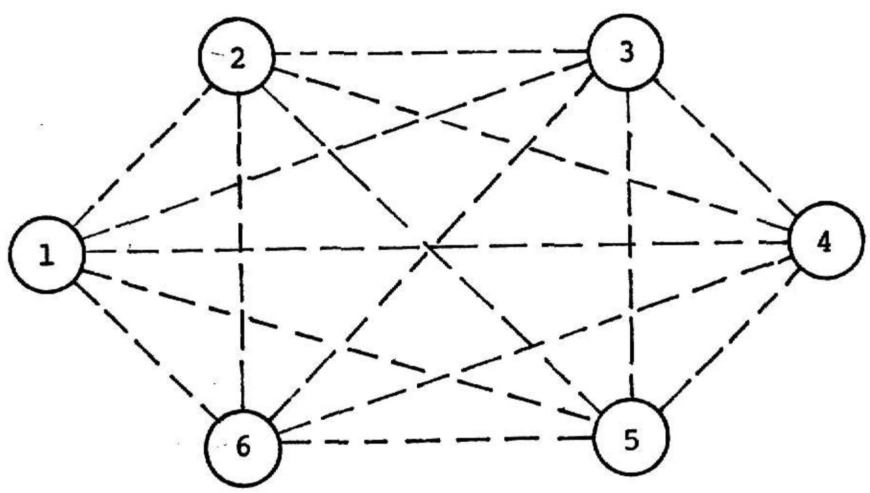


Figure 3.1. Network depicting possible direct precedence relationships.

Such a starting node can be considered as the salesman's home or, at least, someplace where he starts and to where he must return. Nevertheless, when the final starting vector is completed, the route and the route cost can be computed.

When such a route is evaluated, the resultant network would appear as shown in Figure 3.2.

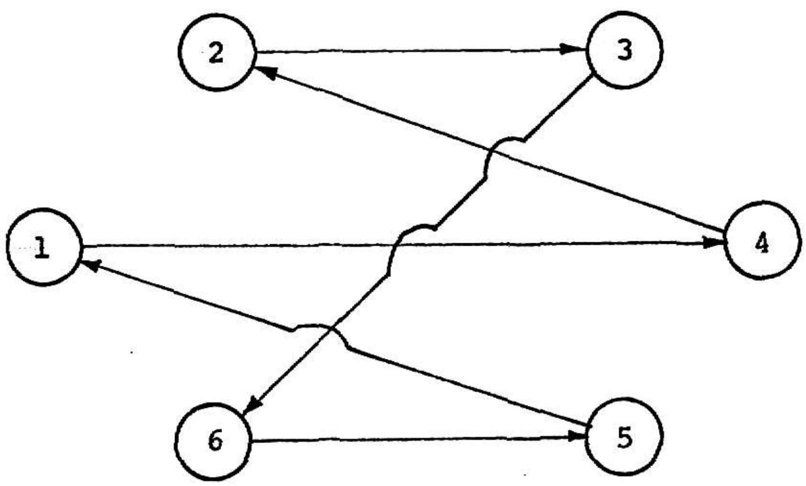


Figure 3.2. A typical solution to the traveling salesman problem.

Sample problem. Consider again the cost chart of Table 3.1 and the network of Figure 3.1. Further, let us consider the starting point of the salesman's journey to be node (1). That is, his route must begin at node (1) and terminate at node (1) after he has visited every other node in the network.

Step 1. Construct the initial starting vector, T^0. Knowing the starting point in the network, the initial starting vector, T^0 can be constructed such that

$$T^0 = [\ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ ].$$

Step 2. Construct the initial precedence matrix, Q^0. From the cost chart and the network diagram, the precedence matrix Q^0 can be constructed as shown in Table 3.2.

Table 3.2. Initial Precedence Matrix, Q^0.

	(1)	(2)	(3)	(4)	(5)	(6)
(1)	0	x27	x43	x16	x30	x26
(2)	x7	0	x16	x1	x30	x25
(3)	x20	x13	0	x35	x5	xι
(4)	x21	x16	x25	0	x18	x18
(5)	x12	x46	x27	x48	0	x5
(6)	x23	x5	x5	x9	x5	0

Note that the entry (3, 6) in the cost chart which is 0, appears as ι in Q^0.

Step 3. Check for nodes to enter. In general, any node can be entered at this point, because all of the columns in the matrix Q^0 are potentially null. However, since node (1) was specified as the starting point, it will be entered first. It should be noted that a simple procedure for resolving conflicts among entering nodes is used for all iterations after the initial one. This procedure simply involves scanning the row of the node currently being visited by the salesman, for the minimum element of the form $xq_{(i, j)}$. The column in which this minimum element occurs is entered next.

Step 4. Update the precedence matrix by entering the latest marked node. By making column (1) in matrix Q^0 null, the resultant matrix, Q^1 is formed as shown in Table 3.3.

Table 3.3. Intermediate Precedence Matrix, Q^1.

	✓(1)	(2)	(3)	✓(4)	(5)	(6)
✓(1)	0	x27	x43	x16	x30	x26
(2)	0	0	x16	x1	x30	x25
(3)	0	x13	0	x35	x5	x1
(4)	0	x16	x25	0	x18	x18
(5)	0	x46	x27	x48	0	x5
(6)	0	x5	x5	x9	x5	0

Step 5. Repeat step 3 by checking for a new node to enter. Obviously, all five remaining nodes in Q^1 can be entered; however, using the procedure discussed above, node (4) is marked to enter. Upon entering node (4), the resultant matrix, Q^2 is formed and is given in Table 3.4.

We have now moved to node (4) and in so doing, scan row (4) for the next node to enter. When this is done, we see that we can enter node (2). After making column (2) null, the updated matrix becomes Q^3.

The next node to enter is found to be node (3). When entry is made, the precedence matrix, Q^4 is formed. The next two nodes to enter are (6) and (5), where the corresponding updates to the precedence matrix yields Q^5 and Q^6, respectively.

Table 3.4. Intermediate Precedence Matrix, Q^2.

	✓ (1)	✓ (2)	(3)	✓ (4)	(5)	(6)
✓(1)	0	0	0	16	0	0
(2)	0	0	x16	0	x30	x25
(3)	0	x13	0	0	x5	x1
✓(4)	0	x16	x25	0	x18	x18
(5)	0	x46	x27	0	0	x5
(6)	0	x5	x5	0	x5	0

Table 3.5. Intermediate Precedence Matrix, Q^3.

	✓ (1)	✓ (2)	✓ (3)	✓ (4)	(5)	(6)
✓(1)	0	0	0	16	0	0
✓(2)	0	0	x16	0	x30	x25
(3)	0	0	0	0	x18	x1
✓(4)	0	16	0	0	0	0
(5)	0	0	x27	0	0	x5
(6)	0	0	x5	0	x5	0

Table 3.6. Intermediate Precedence Matrix, Q^4.

	✓(1)	✓(2)	✓(3)	✓(4)	(5)	✓(6)
✓(1)	0	0	0	16	0	0
✓(2)	0	0	16	0	0	0
✓(3)	0	0	0	0	x18	x1
✓(4)	0	16	0	0	0	0
(5)	0	0	0	0	0	x5
(6)	0	0	0	0	x5	0

Table 3.7. Intermediate Precedence Matrix, Q^5.

	✓(1)	✓(2)	✓(3)	✓(4)	✓(5)	✓(6)
✓(1)	0	0	0	16	0	0
✓(2)	0	0	16	0	0	0
✓(3)	0	0	0	0	0	1
✓(4)	0	16	0	0	0	0
(5)	0	0	0	0	0	0
✓(6)	0	0	0	0	x5	0

Table 3.8. Final Precedence Matrix, Q^6.

	✓ (1)	✓ (2)	✓ (3)	✓ (4)	✓ (5)	✓ (6)
✓(1)	0	0	0	16	0	0
✓(2)	0	0	16	0	0	0
✓(3)	0	0	0	0	0	ι
✓(4)	0	16	0	0	0	0
✓(5)	0	0	0	0	0	0
✓(6)	0	0	0	0	5	0

Step 6. Update the starting time vector. Now that all six nodes have been entered, a resultant route has been determined. When the final precedence matrix, Q^6 is multiplied by the initial starting vector, the resultant vector becomes:

$$T^{0'} = [\ 0 \quad 0 \quad 0 \quad 16 \quad 0 \quad 0\ ].$$

When this vector is added to T^0, the resultant is T^1, such that

$$T^1 = [\ \iota \quad 0 \quad 0 \quad 16 \quad 0 \quad 0\ ].$$

Continuing in this manner, a total of six T vectors are computed. The other five can be presented as follows:

$$T^2 = [\ \iota \quad 32 \quad 0 \quad 16 \quad 0 \quad 0\ ],$$

$$T^3 = [\ \iota \quad 32 \quad 48 \quad 16 \quad 0 \quad 0\ ],$$

$$T^4 = [\ \iota \quad 32 \quad 48 \quad 16 \quad 0 \quad 48\ ],$$

$$T^5 = [\ \iota\ \ 32\ \ 48\ \ 16\ \ 53\ \ 48\ ],$$

and

$$T^6 = [\ \iota\ \ 32\ \ 48\ \ 16\ \ 53\ \ 48\ ].$$

Note that T^6 is identical to T^5.

Step 7. Calculate the final route and route cost. By ordering the entries in the final starting vector with respect to start costs, the following sequence can be given:

(1) (4) (2) (3) (6) (5).

Therefore, the route to be taken becomes

$$1 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 6 \rightarrow 5 \rightarrow 1.$$

The total cost of this route can be computed such that

$$T(S) = \tau_{(5)} + c_{(5,\ 1)},$$

where $\tau_{(5)}$ is the final starting cost of node (5) and $c_{(5,\ 1)}$ is the cost from node (5) to node (1). Therefore, the final value of the route cost can be given as follows:

$$T(S) = 53 + 12 = 65.$$

The final network can be shown in Figure 3.3.

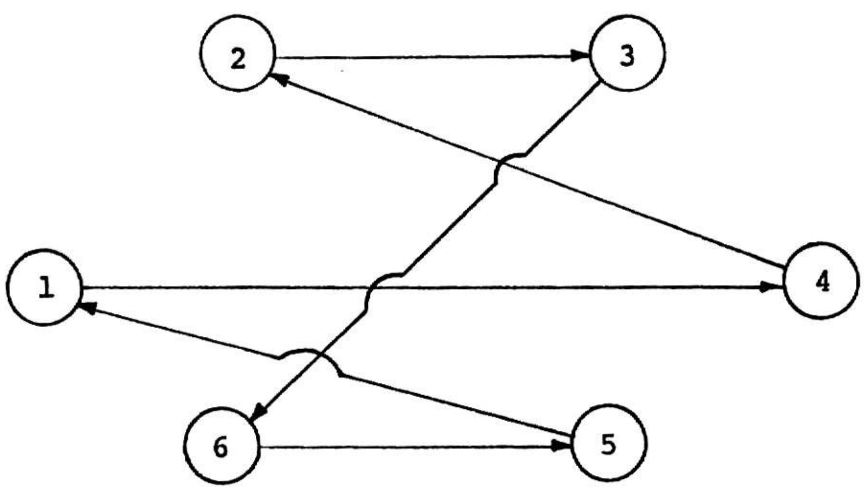


Figure 3.3. A solution to the sample problem.

It should be pointed out that the criterion used in selecting nodes to enter was chosen for simplicity only. No doubt, there are other criteria that may be more powerful; however, this discussion is concerned primarily with application of the network algorithm to the general traveling salesman problem. The resolution of conflicts is, of course, essential to the solution of the problem, but any discussion in depth of specific criteria such as bounding procedures, is not warranted at this time.

It should also be pointed out that the optimal solution to the above problem as presented in Little, et al [12], can be given as follows:

$$1 \rightarrow 4 \rightarrow 3 \rightarrow 5 \rightarrow 6 \rightarrow 2 \rightarrow 1 ,$$

where the cost of the route is 63. Using the simple criterion described earlier for entering nodes, the solution of Little, et al., cannot be obtained; however, by simply entering the nodes consistent with the optimal route above, the route cost of 63 is easily computed.

3.2. The Project Scheduling Problem

Sometimes it is desirable when considering a project network like that shown in Figure 3.4, to determine the critical path through the network. It is critical for, indeed, any shortening of its length, whether it be in terms of distance, time, cost, or any other measure of performance, would result in a savings with respect to the same measure of performance for the entire network. Nevertheless, such problems are referred to as critical path scheduling problems and their solution can be computed with the aid of the network algorithm. This section deals with the formulation of the critical path problem and applicability of the algorithm is demonstrated with a simple example.

Problem formulation. The critical path problem is constructed as a typical network as shown in Figure 3.4. The nodes represent events and the directed branches represent activities. The broken branch represents a dummy activity. It signifies that the event to which it is directed cannot begin until the event from which it is directed is finished. It is, of course, broken to imply that no real or physical precedence occurs. Nevertheless, once such a network is constructed, the process of obtaining a critical path can commence. This procedure entails determining the slack times for each event. Slack time is that amount of time that a node or event can be delayed without increasing the total time to complete the network. An event without slack time is a critical event and any continuous path or transitive chain of precedence relationships between events without slack is a critical path.

To determine the slack times of each event, we must determine the earliest and latest start times of the events. Such a determination of start times can be made using the network algorithm. However, once these start times have been computed,

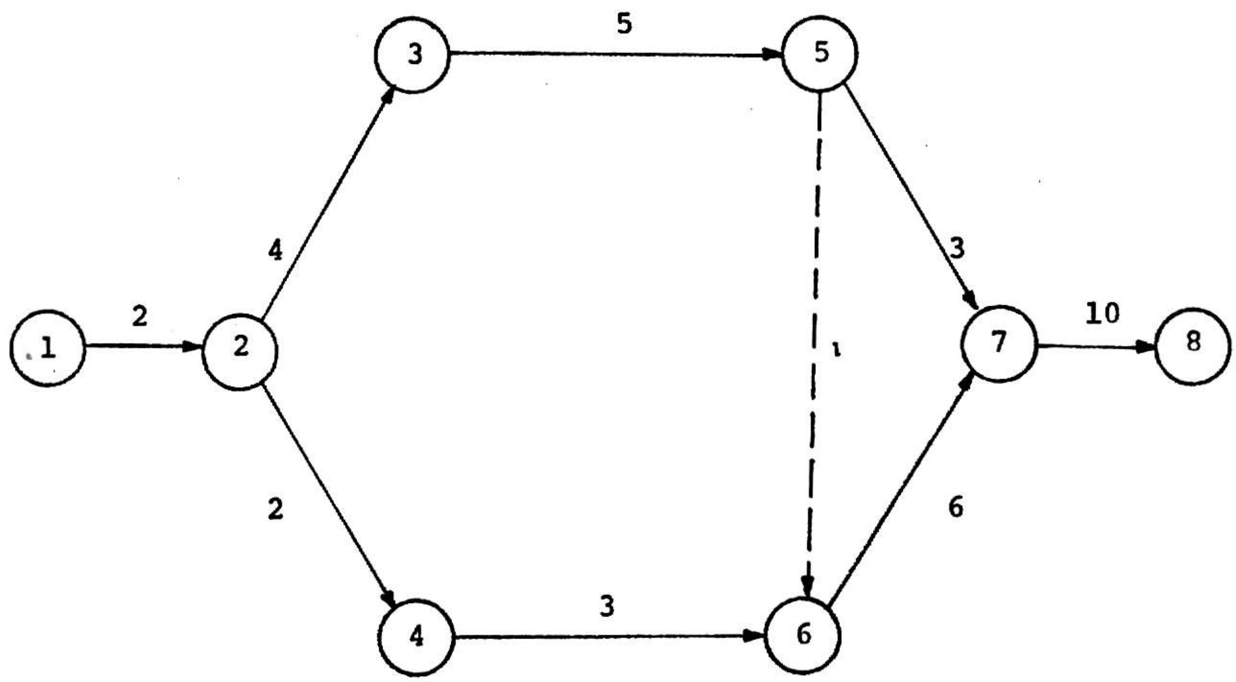


Figure 3.4. A typical network for critical path analysis

the remainder of the solution is computed consistently with such techniques as that described in [1].

Obviously, determination of earliest start times poses no problem. Such start times are simply the result of the final starting time vector, consistent with the method described in the algorithm. That is, once the precedence matrix is constructed, the initial starting vector is multiplied over and over until there is no change in succeeding vectors. The resultant final starting vector represents the earliest start times of all events in the network. It should be pointed out that the section of the algorithm pertaining to computation of the final precedence matrix can be omitted. The nature of the problem itself allows for one and only one precedence matrix because, naturally, all precedence relationships are finalized by virtue of the initial network itself.

While computation of the earliest start times of all events follows directly from the algorithm, the computation of the latest start times requires a slight addition to the operations of star algebra multiplication and addition. Such a change can be formulated such that

$$a \odot -b = |a| - |b| = c$$

and

$$c \oplus d = \min(c, d).$$

It should be pointed out that the above addition to the star algebra operators is made only to facilitate application of the network algorithm to the critical path problem. Any further application, although perhaps desirable, is not reported at this time.

Besides imposing the above convention for star algebra addition and multiplication, we must change the precedence matrix such that the new matrix becomes

$$Q' = [- \iota Q]^{T}.$$

That is, the original precedence matrix used to determine earliest start times is multiplied by $-\iota$, and made negative, after which time it is transposed.

When the new precedence matrix has been formed, the starting vector can be constructed such that the only non-zero entry in the vector is that corresponding to the earliest starting time of the final event in the network. That is, determination of the latest start times of all events can be accomplished by beginning at the final event in the network and proceeding backwards through the network until the first node or event is reached. Obviously, the only event whose start time is known is the final event.

In summary, the critical path analysis of a project network can be facilitated using the network algorithm in two capacities. The first is determination of the earliest start times of all events in the network and follows directly from the algorithm. The second involves determination of the latest starting times of all events and requires the following changes:

(1) the transpose of the original precedence matrix after it has been made negative, (2) alteration of the starting time vector by beginning at the last event in the network and moving from back to front, and (3) the additional convention for star algebra multiplication and addition. These changes as well as the entire application of the network algorithm to the critical path problem are illustrated in a sample problem.

Sample problem. Consider the project network of Figure 3.4, which was taken from Ashour [1]. Each node represents an event, while activities are represented by the directed branches. The duration of activities is represented by the numbers attached to each branch. The first analysis will be made to determine the earliest start time of each event.

Step 1. Construct the initial starting vector, T^0. Only the starting time of event (1) is known and is so signified by the entry of ι in the initial starting vector such that

$$T^0 = [\ \iota \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ].$$

Step 2. Construct the precedence matrix, Q. As mentioned in the problem formulation, there is only one precedence matrix; consequently, the exponents can be omitted from the notation. Nevertheless, the precedence matrix can be constructed as shown in Table 3.9. Since no updates are required to the precedence matrix, we can proceed directly to step 6.

Table 3.9. Precedence Matrix, Q.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1)	0	2	0	0	0	0	0	0
(2)	0	0	4	2	0	0	0	0
(3)	0	0	0	0	5	0	0	0
(4)	0	0	0	0	0	3	0	0
(5)	0	0	0	0	0	1	3	0
(6)	0	0	0	0	0	0	6	0
(7)	0	0	0	0	0	0	0	10
(8)	0	0	0	0	0	0	0	0

Step 6. Update the starting time vector. When the precedence matrix, Q is multiplied by T^0, the resultant becomes

$$[\ 0 \quad 2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ].$$

When the above vector is added to T^0, the resultant is T^1 such that

$$T^1 = [\ 1 \quad 2 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\ ].$$

Following in the above manner, a total of seven starting vectors are computed. They are as follows:

T^2 = [1 2 6 4 0 0 0 0],

T^3 = [1 2 6 4 11 7 0 0],

T^4 = [1 2 6 4 11 11 14 0],

T^5 = [1 2 6 4 11 11 17 24],

T^6 = [1 2 6 4 11 11 17 27],

and

T^7 = [1 2 6 4 11 11 17 27].

Note that T^6 and T^7 are identical.

Now that the earliest start times of all events have been computed, the latest start times can be determined.

Step 1. Construct the initial starting vector. Recalling that the latest start times are determined by moving from the back to the front of the network, the initial starting vector now can be constructed such that

T^0 = [0 0 0 0 0 0 0 27],

where the only event whose start time is known is the final event in the network or event (8).

Step 2. Construct the new precedence matrix, Q'. By making every non-zero entry in Q negative and transposing the resultant matrix, the matrix Q' can be given as shown in Table 3.10.

Step 6. Update the starting time vector. Utilizing the new convention for star algebra multiplication and addition, the product of the initial starting vector, T^0 and the precedence matrix, Q' can be given as

[0 0 0 0 0 0 17 0].

When the above vector is added to T^0, the resultant, T^1 can be given such that

$$T^1 = [\ 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 17 \quad 27\ ].$$

Note that in the above addition, normal star algebra holds.

Proceeding in the above manner, we find that there are seven starting vectors. They can be given as follows:

$$T^2 = [\ 0 \quad 0 \quad 0 \quad 0 \quad 14 \quad 11 \quad 17 \quad 27\ ],$$

$$T^3 = [\ 0 \quad 0 \quad 9 \quad 8 \quad 11 \quad 11 \quad 17 \quad 27\ ],$$

$$T^4 = [\ 0 \quad 5 \quad 6 \quad 8 \quad 11 \quad 11 \quad 17 \quad 27\ ],$$

$$T^5 = [\ 3 \quad 2 \quad 6 \quad 8 \quad 11 \quad 11 \quad 17 \quad 27\ ],$$

$$T^6 = [\ 0 \quad 2 \quad 6 \quad 8 \quad 11 \quad 11 \quad 17 \quad 27\ ],$$

and

$$T^7 = [\ 0 \quad 2 \quad 6 \quad 8 \quad 11 \quad 11 \quad 17 \quad 27\ ],$$

where, of course, T^6 and T^7 are identical.

Now that the latest start times of all events have been computed, the original network is reconstructed and each event in the network is accompanied by its earliest and latest start times as shown in Figure 3.5.

The application of the network algorithm is concluded at this point; however, further solution of the critical path problem can be found in [1]. Such analysis, in this work, is not pertinent at this time and is omitted.

Table 3.10. Precedence Matrix Q' used to Determine Latest Start Times of All Events.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
(1)	0	0	0	0	0	0	0	0
(2)	-2	0	0	0	0	0	0	0
(3)	0	-4	0	0	0	0	0	0
(4)	0	-2	0	0	0	0	0	0
(5)	0	0	-5	0	0	0	0	0
(6)	0	0	0	-3	-1	0	0	0
(7)	0	0	0	0	-3	-6	0	0
(8)	0	0	0	0	0	0	-10	0

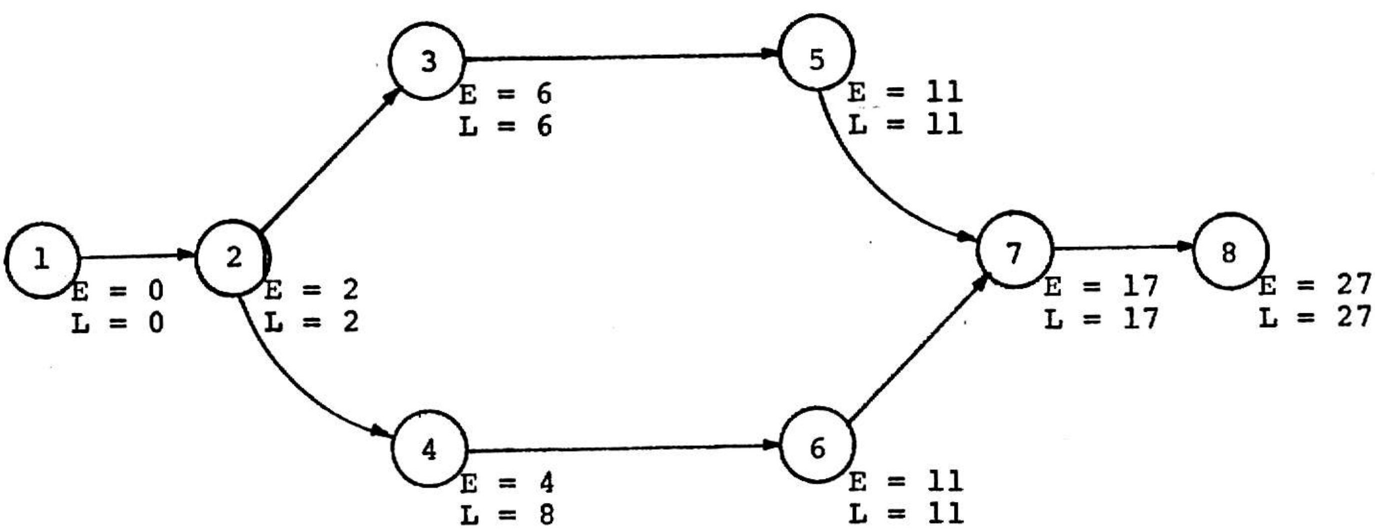


Figure 3.5. Sample problem, depicting earliest and latest start times of all events.

3.3 The Explosion Problem

The explosion problem, sometimes referred to as the parts requirement problem, presents a third area of application for the network algorithm. Moreover, just as in the project scheduling problem, all precedence relationships are initially specified by the nature of the explosion problem itself. Consequently, the explosion problem involves determination of operation (node) starting times only. Therefore, the use of the algorithm can be abbreviated to include only the construction of the precedence matrix and the manipulation of the starting time vector.

Problem Formulation. The general form of the explosion problem usually appears in a bill of material. Such a bill of material might appear as in Table 3.11, where the product or node (1) is constructed from nine components, as can be seen from Figure 3.6. Six of the components are made (2, 3, 4, 5, 6, and 7) and three are purchased (8, 9, and 10). Furthermore, there may be more than one component essential to the construction of another. That is, component 4 requires two parts of component 6. In general, the numbers on the directed branches between nodes in Figure 3.6 represent the requirement from one node to another.

The construction of the precedence matrix can be made directly from the bill of material and the network. Each branch in the network represents one or more components where the time per component is given in the bill of material. When this time per component is multiplied by the number of components necessary, the time to move from node (i) to node (j) is obtained. These times between nodes become the entries in the precedence matrix. Obviously, all entries in the precedence matrix will be either zero or some positive scaler. As mentioned earlier, this concept is logical since all precedence relationships are already determined from the nature of the problem. Consequently, the initial precedence matrix is also the final precedence matrix. Once this matrix is constructed, the network can be evaluated with respect to starting times of all nodes.

The starting vector is constructed in the usual manner, where the only entries not zero are those representing the initial components or nodes in the network. When the initial vector has been constructed, it can be multiplied over and over as described in the algorithm until the final starting vector has been obtained. The entries of the final vector represent the earliest start times of all nodes. Of course, the total time to make the product considered in the bill of material can be computed just as schedule time was computed earlier.

Sample problem. Consider the bill of material given in Table 3.11. Further, consider the corresponding network of Figure 3.6. The precedence matrix can be constructed from the bill of material and the network as shown in Table 3.12.

Table 3.11 Bill of Material for Sample Problem.

Node	Quantity	Time/Component	Total Time
1	1	5	5
2	3	4	12
3	2	3	6
4	1	6	6
5	5	2	10
6	4	3	12
7	3,2	4	12, 8
8	2	5	10
9	4,3	3	12, 9
10	5	1	5

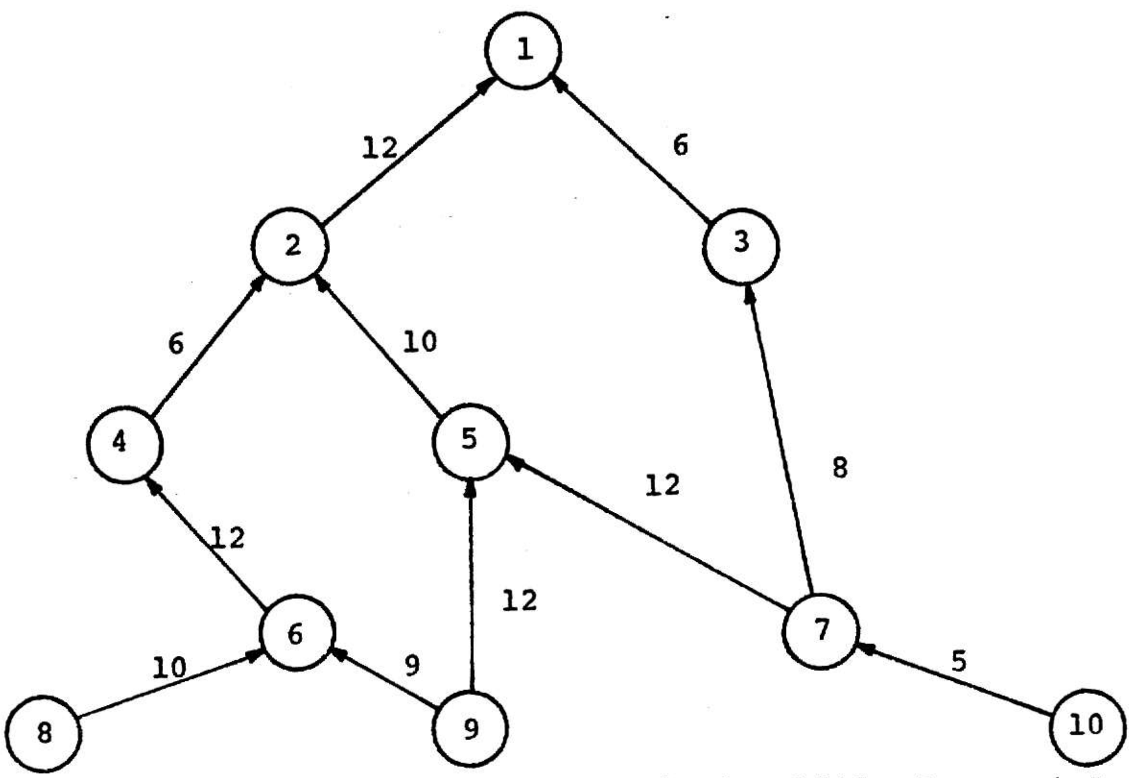


Figure 3.6. Network depicting bill of material for sample problem.

Table 3.12. Precedence Matrix, Q for Sample Problem.

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)
(1)	0	0	0	0	0	0	0	0	0	0
(2)	12	0	0	0	0	0	0	0	0	0
(3)	6	0	0	0	0	0	0	0	0	0
(4)	0	6	0	0	0	0	0	0	0	0
(5)	0	10	0	0	0	0	0	0	0	0
(6)	0	0	0	12	0	0	0	0	0	0
(7)	0	0	8	0	12	0	0	0	0	0
(8)	0	0	0	0	0	10	0	0	0	0
(9)	0	0	0	0	12	9	0	0	0	0
(10)	0	0	0	0	0	0	5	0	0	0

The initial starting vector, T^0 becomes

$$[\ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ \iota\ \ \iota\ \ \iota\ ],$$

where, obviously, only the earliest starting times of nodes (8), (9), and (10) are known and so signified by ι in T^0. When T^0 is multiplied by the precedence matrix, the resultant vector becomes

$$[\ 0\ \ 0\ \ 0\ \ 0\ \ 12\ \ 10\ \ 5\ \ 0\ \ 0\ \ 0\ ].$$

When added to T^0, the new vector, T^1, is constructed such that

$$T^1 = [\ 0 \quad 0 \quad 0 \quad 0 \quad 12 \quad 10 \quad 5 \quad \iota \quad \iota \quad \iota\ ].$$

Proceeding as usual, we find that a total of five starting vectors must be computed. They can be presented as follows:

$$T^2 = [\ 0 \quad 22 \quad 13 \quad 22 \quad 17 \quad 10 \quad 5 \quad \iota \quad \iota \quad \iota\ ],$$

$$T^3 = [34 \quad 28 \quad 13 \quad 22 \quad 17 \quad 10 \quad 5 \quad \iota \quad \iota \quad \iota\ ],$$

$$T^4 = [40 \quad 28 \quad 13 \quad 22 \quad 17 \quad 10 \quad 5 \quad \iota \quad \iota \quad \iota\ ],$$

and

$$T^5 = [40 \quad 28 \quad 13 \quad 22 \quad 17 \quad 10 \quad 5 \quad \iota \quad \iota \quad \iota\ ].$$

Note that T^5 is identical to T^4.

The total time T(S) can be computed as follows:

$$T(S) = \max\ (\tau_{(i)}) + t_{(i)},$$

$$= 40 + 5 = 45 \text{ time units.}$$

That is, the total time for one operator to assemble the product specified by the bill of material is 45 time units. Of course, this interpretation can be extended to include the case of multiple operators, in which case the time of 45 is simply the total of all individual times involved in the product's assembly.

CHAPTER IV

Computational Experiments

The network algorithm discussed in section 2.3 was programmed in FORTRAN IV for use on the IBM 360/50 computer. The program consists of a main program and two subroutines. The first subroutine represents the composite lower bound which is used to improve the solution while the second subroutine is used in conjunction with the bounding subroutine to compute completion times. Evaluation of the network algorithm was made possible by solving a wide range of problems. The bulk of the computational experiments were made with respect to the typical job shop scheduling problem. The number and type of such problems is shown in Table 4.1. The remaining experiments consist of four traveling salesman problems, which unlike the job shop problems, were simply solved by hand to illustrate the applicability of the network algorithm.

4.1. Job Shop Problems

The size of the job shop problems vary with respect to both the number of jobs and machines. The smallest number of jobs considered was 3, while the largest was 12. The number of machines ranges between 3 and 5. A total of 17 experiments were conducted. A total of 25 problems were solved for each experiment with the exception of experiments VIII and X, in which case only 10 problems were solved. The entries in the machine ordering and processing time matrices were generated in a random fashion. More specifically, the values of the processing times were generated from a uniform distribution between one and 30, inclusively.

The performance of the network algorithm was made with respect to three factors pertaining to the job shop problem: (1) the computational time involved to obtain a solution, (2) the quality of the solution, and (3) the number of iterations and conflicts for each problem size. The statistics maximum, minimum, mean, and standard deviation for the factors efficiency, number of conflicts, and number of iterations have also been computed.

Computational time. The computational time was, of course, one of the main considerations in the evaluation of the performance of the network algorithm. In Table 4.1, the computational time per problem size is shown on a job group basis. The relationship between computational time and an increase in the number of machines per job group can be seen in Figure 4.4. It is interesting to note that the experiments with the least number of jobs exhibited nearly linear relationships when the number of machines was increased from 3 through 5. Of course, such analysis cannot be made for problems with 8 or more jobs because only 2 machine sizes, namely 3 and 4, were considered.

When the number of machines is held constant and the job size is increased, the effect upon computational time can be shown graphically in Figures 4.2 through 4.4. Again, linearity is, at least, graphically evident when the level of jobs is in the range of 3 to 8. However, when the job level reaches 10 and 12, the computational time increases rapidly.

The next analysis that was made with respect to computational time was that in which the time per node was investigated. These realtionships are tabulated in Table 4.4 and shown graphically in Figure 4.5. The instigation for making this type of analysis arose when it was noticed that for problems with the same number of nodes, the computational time per problem was very close. For example, from Table 4.4, it can be seen that the 4x5 and 5x4 problems, both consisting of 20 nodes, exhibited the same computational time. Such was the case for the 15 node or 3x5 and 5x3 problems. Consequently, all of the problems were reorganized with respect to their corresponding number of nodes. When a plot was made of the average time per node for each total number of nodes, it was observed from Figure 4.5, that, just as before, the lower portion of the curve approximated a linear relationship, while the upper portion did not. More specifically, the portion of the curve that is non-linear seems to occur when the node level reaches approximately 30.

At this stage in the research, only a speculative explanation can be made concerning the rapid increase in computational time when the number of nodes increases. Nevertheless, consider the case of all problems consisting of 30 or more nodes. These problems were 6x5, 8x4, 10x3, 10x4, 12x3, and 12x4. By recalling that entries are made in the precedence matrix on a machine block basis, where each block consists of J jobs, it is natural that as J increases, the number of conflicts that occur per machine block increases. The immediate result of this phenomena is that the lower bounds of all nodes in the resulting conflict sets must be computed in order that a resolution can be made. Naturally, as the number of such lower bound computations increases, the execution or computational time per problem would increase. Of course, inherent in this situation is the expectation that as conflicts themselves increase, the number of nodes in the conflict set would also increase, thereby increasing computation involved in resolving the conflict. It is believed that if such lower bound computation were excluded from the total computational time, the relationship between time per node and the number of nodes would be, at least, more nearly linear. However, such an exclusion would mean that conflicts would be broken at random or, at best, by some procedure requiring much less computation than that of a lower bound on schedule time. If this were the case, the expected efficiency of solution using such a selection criteria, would be much less than that now experienced using the composite lower bound.

Quality of Solution. The efficiency of the solution found by the network algorithm was computed from the ratio of that solution to the known optimal solution.obtained by B-and-B technique [11]. These results have been tabulated in Table 4.1

and 4.2. The number of optimal solutions per problem size as well as the percent of optimal solutions and the corresponding range of efficiency are shown in Table 4.5. The scarcity in the number of available optimal solutions does not allow a good analysis with respect to possible trends in efficiency as problem size increases; however, one could expect efficiency to decrease as problems become larger. The reason for this can be attributed, most likely, to the increase in the number of conflicts that occur when the problem size, especially job size, increases. That is, when a conflict is encountered and is ultimately resolved, the result is that idle time is generated by virtue of the resolution.

The logic used above is supported, in part, at least by the fact that in every instance where there were no conflicts in a problem solution, the optimal solution was obtained. Of course, the frequency of such occurrances is low due to the fact that whenever the number of jobs is greater than the number of machines, there will be at least one conflict and most probably, more than one. As can be seen from the table of experiments, 12 of 17 problem sets involved situations in which J was greater than M.

Number of Iterations. As mentioned earlier in this thesis, only one node can be entered per machine block per iteration. Obviously then, if M nodes are entered at every

Table 4.1. Computational time***, iterations, conflicts, and efficiency with job grouping.

Exp. No.	Prob. Size	No. of Prob.	No. of Opt. Solu. Found	Efficiency μ_{eff}	Efficiency σ_{eff}	Number of Iterations max	Number of Iterations min	Number of Iterations μ	Number of Iterations σ	Number of Conflicts max	Number of Conflicts min	Number of Conflicts μ	Number of Conflicts σ	Ave. Comp. Time
I	3x4	25	25	96.23	4.56	7	4	5.76	.86	5	0	2.36	1.41	1.44
II	3x5	25	25	97.55	4.33	8	5	6.40	.98	6	0	2.32	1.62	2.16
III	4x3	25	25	96.67	5.23	6	5	5.56	.50	6	3	4.76	0.81	1.58
IV	4x4	25	25	95.21	6.37	8	5	6.40	.69	7	2	4.56	1.39	2.45
V	4x5	25	25	95.78	6.03	9	6	7.16	.73	8	2	4.28	1.73	3.74
VI	5x3	25	25	96.99	3.87	7	5	6.24	.59	9	5	7.08	1.35	2.16
VII	5x4	25	25	93.17	5.97	9	7	7.48	.64	14	4	8.28	2.11	3.74
VIII	5x5	10	10	93.82	6.30	10	7	8.00	.77	8	4	7.00	1.18	5.76
IX	6x3	25	25	93.77	6.87	8	6	7.20	.57	13	6	9.76	1.50	3.60
X	6x4**	25	24	89.44	9.32	10	7	8.13	.67	13	7	10.71	1.46	5.90
XI	6x5	10	2	86.71	3.62	11	8	9.60	.92	16	10	12.90	1.81	10.08
XII	8x3	25	11	93.65	5.18	10	8	9.16	.61	19	9	15.40	2.38	7.49
XIII	8x4	25	4	89.80	6.74	11	9	9.88	.59	23	15	18.36	2.10	12.53
XIV	10x3	25	4	93.73	6.37	12	10	10.96	.82	25	15	21.04	2.81	9.65
XV	10x4	25	1	92.48		14	10	11.76	.91	31	19	25.88	2.92	23.42
XVI	12x3	25	7	96.71	1.87	15	12	13.20	.69	32	22	27.28	2.79	23.76
XVII	12x4	25	*	*	*	16	12	13.84	.97	40	29	34.76	3.06	40.03

*No optimal solution is available.
**Computational time based on 25 problems; all other statistics based on 24 problems.
***Computational time in seconds

Table 4.2. Computational time**, iterations, conflicts, and efficiency with machine grouping.

Exp. No.	Prob. Size	No. of Prob.	No. of Opt. Solu. Found	Efficiency μ_{eff}	Efficiency σ_{eff}	Number of Iterations max	min	μ	σ	Number of Conflicts max	min	μ	σ	Ave. Comp. Time
III	4x3	25	25	96.97	5.23	6	5	5.56	0.50	6	3	4.76	0.81	1.58
VI	5x3	25	25	96.99	3.87	7	5	6.24	0.59	9	5	7.08	1.35	2.16
IX	6x3	25	25	93.77	6.87	8	6	7.20	0.57	13	6	9.76	1.50	3.60
XII	8x3	25	11	93.65	5.18	10	8	9.16	0.61	19	9	15.40	2.38	7.49
XIV	10x3	25	4	93.73	6.37	12	10	10.96	0.82	25	15	21.04	2.81	9.65
XVI	12x3	25	7	96.71	1.87	15	12	13.20	0.69	32	22	27.28	2.79	23.76
I	3x4	25	25	96.23	4.56	7	4	5.76	0.86	5	0	2.36	1.41	1.44
IV	4x4	25	25	95.21	6.37	8	5	6.40	0.69	7	2	4.56	1.39	2.45
VII	5x4	25	25	93.17	5.97	9	7	7.48	0.64	14	4	8.28	2.11	3.74
X	6x4	25	24	89.44	9.32	10	7	8.13	0.67	13	7	10.71	1.46	5.90
XIII	8x4	25	4	89.80	6.74	11	9	9.88	0.59	23	15	18.36	2.10	12.53
XV	10x4	25	1	92.48		14	10	11.76	0.91	31	19	25.88	2.92	23.42
XVIII	12x4	25	*	*	*	16	12	13.84	0.97	40	29	34.76	3.06	40.03
II	3x5	25	25	97.55	4.33	8	5	6.40	0.98	6	0	2.32	1.62	2.16
V	4x5	25	25	95.78	6.03	9	6	7.16	0.73	8	2	4.28	1.73	3.74
VIII	5x5	10	10	93.82	6.30	10	7	8.00	0.77	8	4	7.00	1.18	5.76
XI	6x5	10	2	86.71	3.62	11	8	9.60	0.92	16	10	12.90	1.81	10.08

*No optimal solution is available
**Computational time in seconds

Table 4.3. Number of Iterations Per Problem Size.

Number of jobs	Problem Size	Number of Iterations	Ave. Iterations
3	3 x 4	5.76	6.08
	3 x 5	6.40	
4	4 x 3	5.56	
	4 x 4	6.40	6.37
	4 x 5	7.16	
5	5 x 3	6.24	
	5 x 4	7.48	7.22
	5 x 5	8.00	
6	6 x 3	7.20	
	6 x 4	8.13	8.31
	6 x 5	9.60	
8	8 x 3	9.16	9.52
	8 x 4	9.88	
10	10 x 3	10.96	11.36
	10 x 4	11.76	
12	12 x 3	13.20	13.52
	12 x 4	13.84	

Table 4.4. Computational Time per Node*.

Number of nodes Per Problem	Problem Size	Ave. Comp. Time	Ave. Comp. Time/ Number Nodes	Time/Node
12	3x4 4x3	1.44 1.58	1.51	0.126
15	3x5 5x3	2.16 2.16	2.16	0.144
16	4x4	2.45	2.45	0.153
18	6x3	3.60	3.60	0.200
20	4x5 5x4	3.74 3.74	3.74	0.187
24	6x4 8x3	5.90 7.49	6.70	0.279
25	5x5	5.76	5.76	0.230
30	6x5 10x3	10.08 9.65	9.87	0.329
32	8x4	12.53	12.53	0.391
36	12x3	23.76	23.76	0.660
40	10x4	23.42	23.42	0.586
48	12x4	40.03	40.03	0.834

*Computational time in seconds

Table 4.5. Number of Optimal Solutions Found per Problem Size.

Prob. Size	No. of Opt. Sol. Found Using B&B	No. of Opt. Sol. Found Using Network Algorithm	%Optimal	Range
3x4	25	12	48.0	.84 - 1.00
3x5	25	16	64.0	.82 - 1.00
4x3	25	11	44.0	.80 - 1.00
4x4	25	11	44.0	.76 - 1.00
4x5	25	13	52.0	.78 - 1.00
5x3	25	12	48.0	.90 - 1.00
5x4	25	5	20.0	.79 - 1.00
5x5	10	2	20.0	.81 - 1.00
6x3	25	9	36.0	.80 - 1.00
6x4	24	4	17.0	.71 - 1.00
6x5	2	0	0.0	.83 - 1.00
8x3	11	3	27.0	.85 - 1.00
8x4	4	0	0.0	.80 - .98
10x3	4	2	50.0	.86 - 1.00
10x4	1	0	0.0	
12x3	7	1	14.0	.94 - 1.00
12x4	*	*		

*No optimal solution available

iteration, a minimum of J iterations results. The number of iterations per size of problem has been tabulated in Table 4.1 and 4.2. From the tables as well as Figure 4.6, it can be seen that as the problem size, and more specifically the job size, increases, the number of iterations is more nearly J.

4.2. The Traveling Salesman Problem.

Four traveling salesman problems have been solved by hand and are presented in this section only as a demonstration of the applicability of the network algorithm. The problems are presented in the form of a distance chart. Included, of course, is the solution as well as the corresponding efficiency. As in Chapter III, the criteria for entry in the precedence matrix, is a simple look ahead technique which is used for simplicity. However, some criteria could be developed to improve the solution.

Problem 1. The problem formulated in Table 4.6 is taken from Cochran [14]. The solution obtained using the network algorithm is 37. A measure of efficiency is not possible because an optimal solution to the problem is not available. Nevertheless, the route to be taken can be given as follows:

$$1 \rightarrow 3 \rightarrow 6 \rightarrow 4 \rightarrow 5 \rightarrow 2 \rightarrow 1 \quad ,$$

where node 1 is considered to be home.

Table 4.6 Distance Chart for Problem 1

0	11	9	12	13	10
11	0	10	11	4	8
9	10	0	8	9	4
12	11	8	0	7	2
13	4	9	7	0	5
10	8	4	2	5	0

Problem 2. The second problem shown in Table 4.7* involves five cities. Again, no optimal solution is available; however, the solution obtained with the network algorithm is 38. The corresponding route can be given as follows:

$$1 \rightarrow 5 \rightarrow 4 \rightarrow 2 \rightarrow 3 \rightarrow 1 \quad ,$$

where node 1 is home.

Table 4.7 Distance Chart for Problem 2

0	10	8	4	2
10	0	11	9	12
8	11	0	10	11
4	9	10	0	8
2	12	11	8	0

Problem 3. The problem shown in Table 4.8 is a five-city problem taken from [15] . The solution obtained from

*We are somewhat at a loss to identify the original author of this problem.

the network algorithm is 148 which is the same as the optimal solution, yielding an efficiency of 100%. The route to be taken can be given such that

$$2 \rightarrow 3 \rightarrow 4 \rightarrow 5 \rightarrow 1 \rightarrow 2 \quad ,$$

where home is node 2.

Table 4.8 Distance Chart for Problem 3

0	30	26	50	40
30	0	24	40	50
26	24	0	24	26
50	40	24	0	30
40	50	26	30	0

Problem 4. The final problem, shown in Table 4.9, is a ten-city problem from [15]. The solution by the network algorithm is 387 while the optimal solution is 378; hence, an efficiency of 98%. If home is considered to be node 9, the route can be given as follows:

$$9 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1 \rightarrow 7 \rightarrow 6 \rightarrow 8 \rightarrow 10 \rightarrow 9 \quad .$$

Table 4.9 Distance Chart for Problem 4.

0	28	57	72	81	85	80	113	89	80
28	0	28	45	54	57	63	85	63	63
57	28	0	20	30	28	57	57	40	57
72	45	20	0	10	20	72	45	20	45
81	54	30	10	0	22	81	41	10	41
85	57	28	20	22	0	63	28	28	63
80	63	57	72	81	63	0	80	89	113
113	85	57	45	41	28	80	0	40	80
89	63	40	20	10	28	89	40	0	40
80	63	57	45	41	63	113	80	40	0

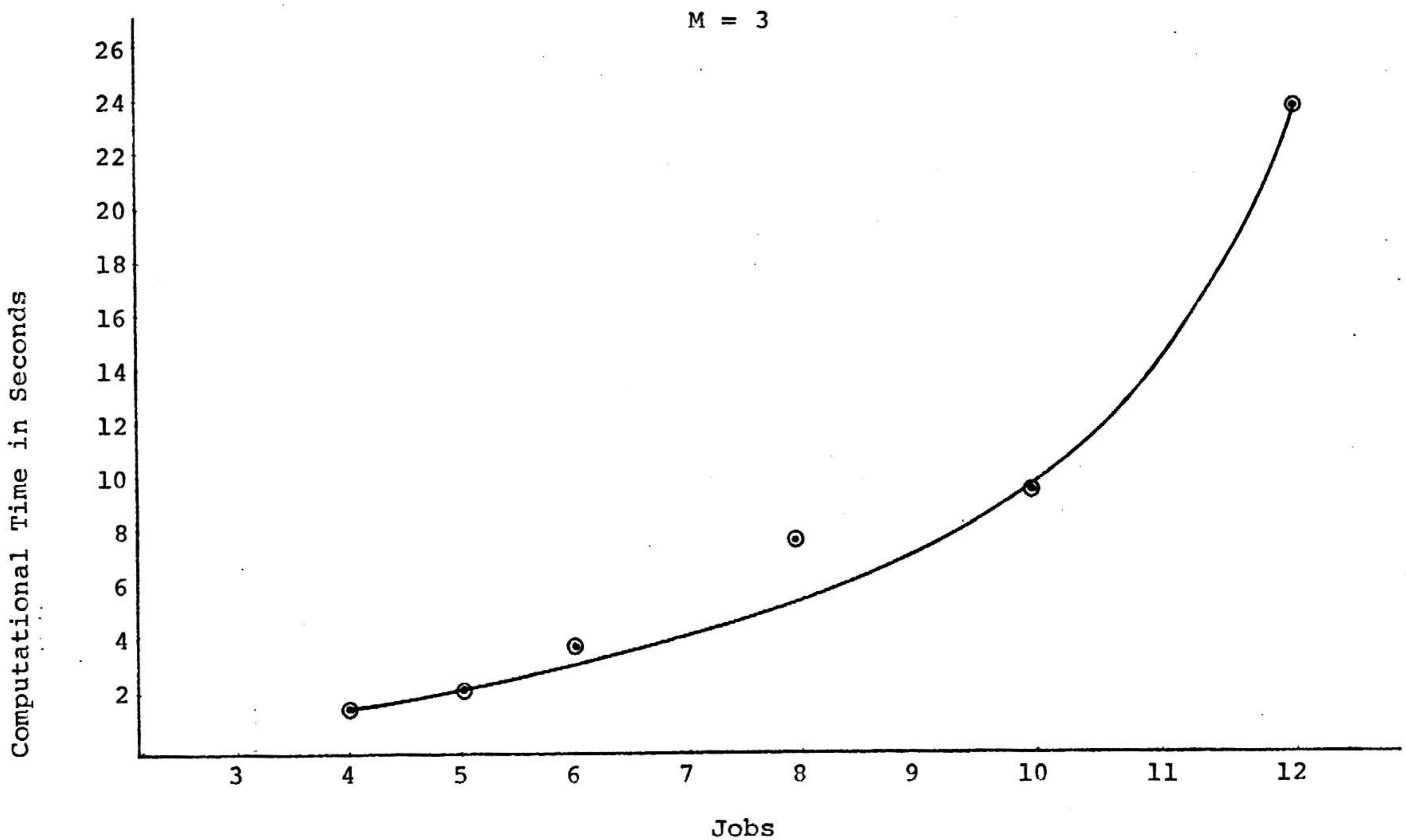


Figure 4.1. Relationship between computational time and the number of jobs, with constant number of machines.

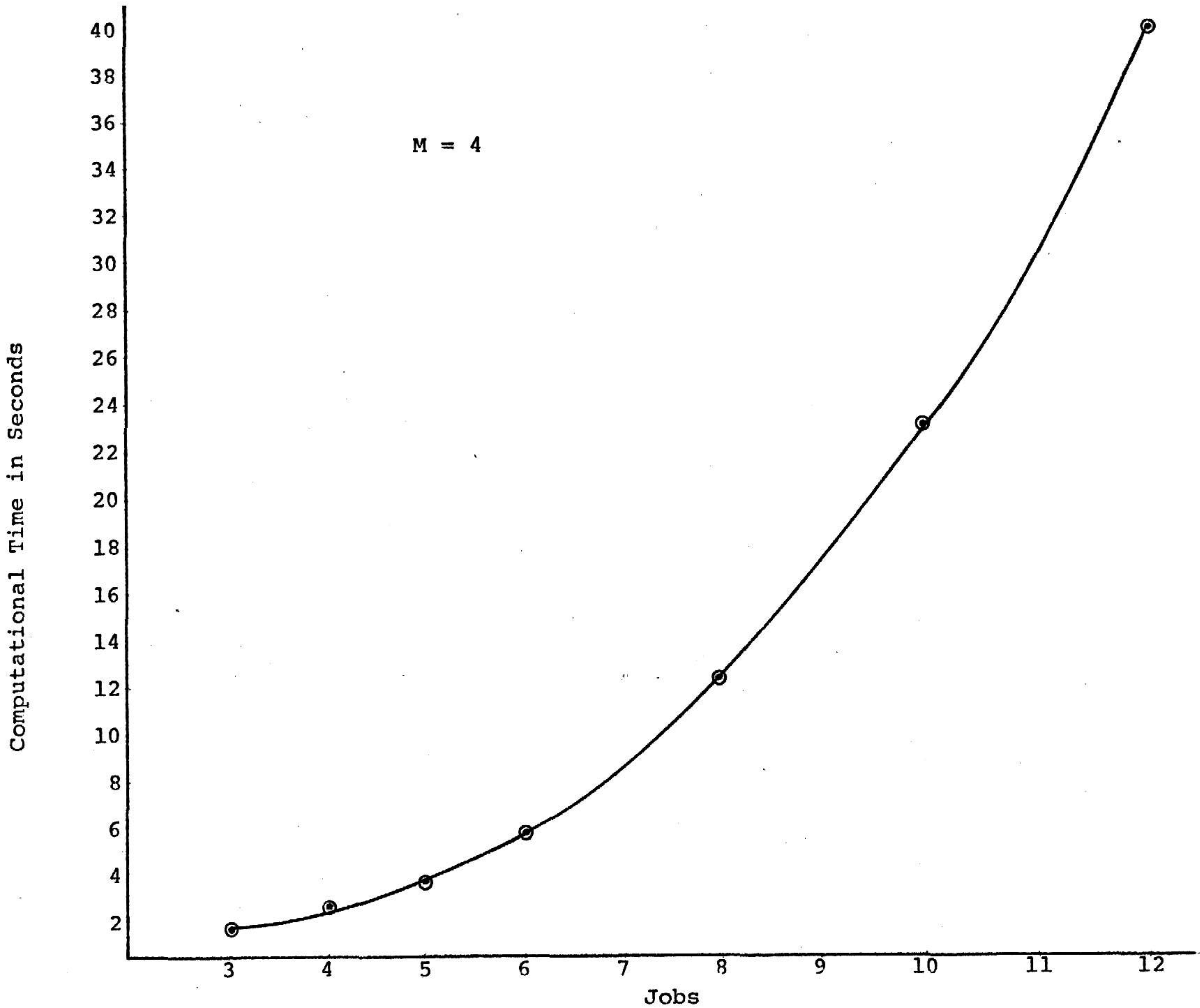


Figure 4.2. Relationship between computational time and the number of jobs with constant number of machines.

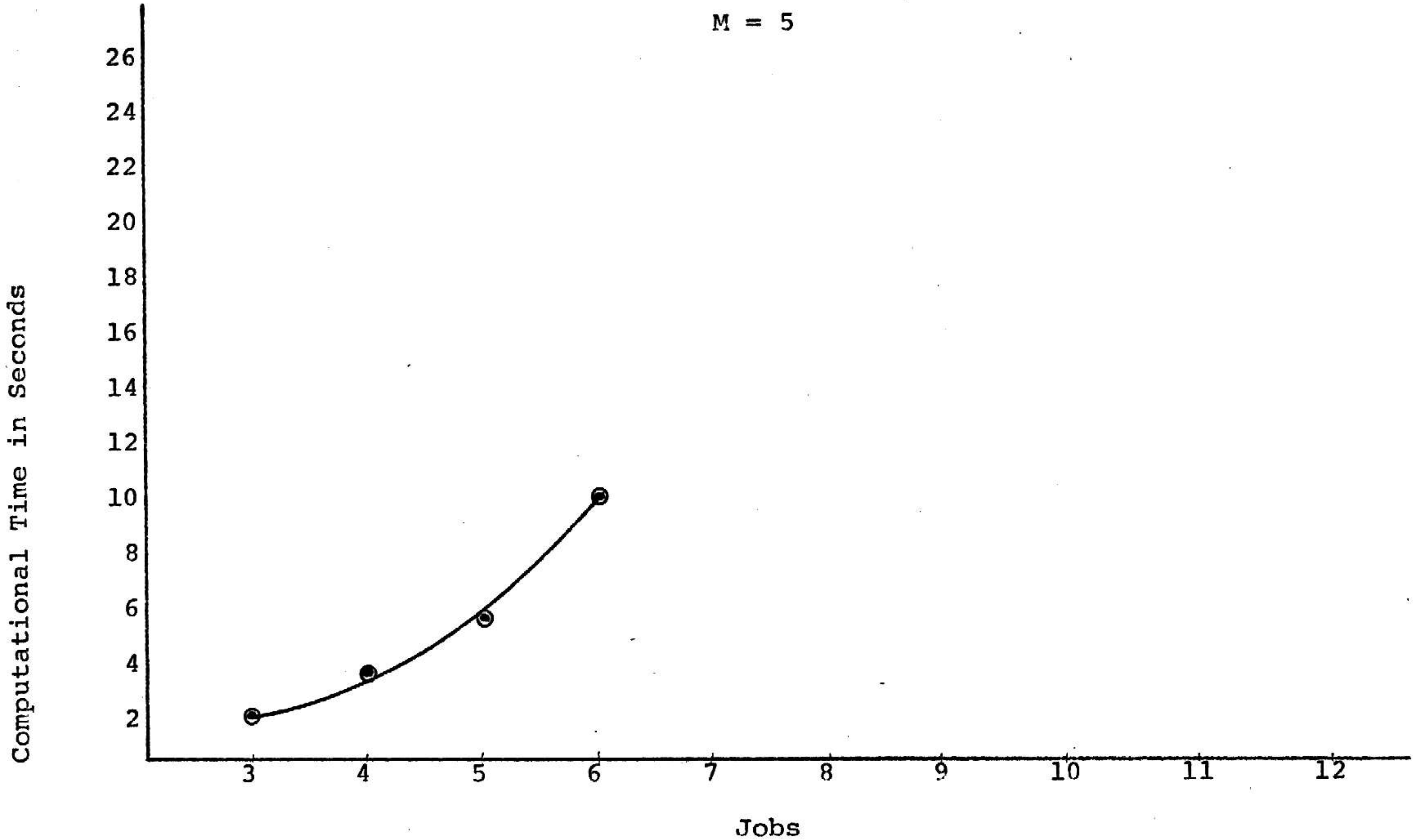


Figure 4.3. Relationship between computational times and the number of jobs with constant number of machines.

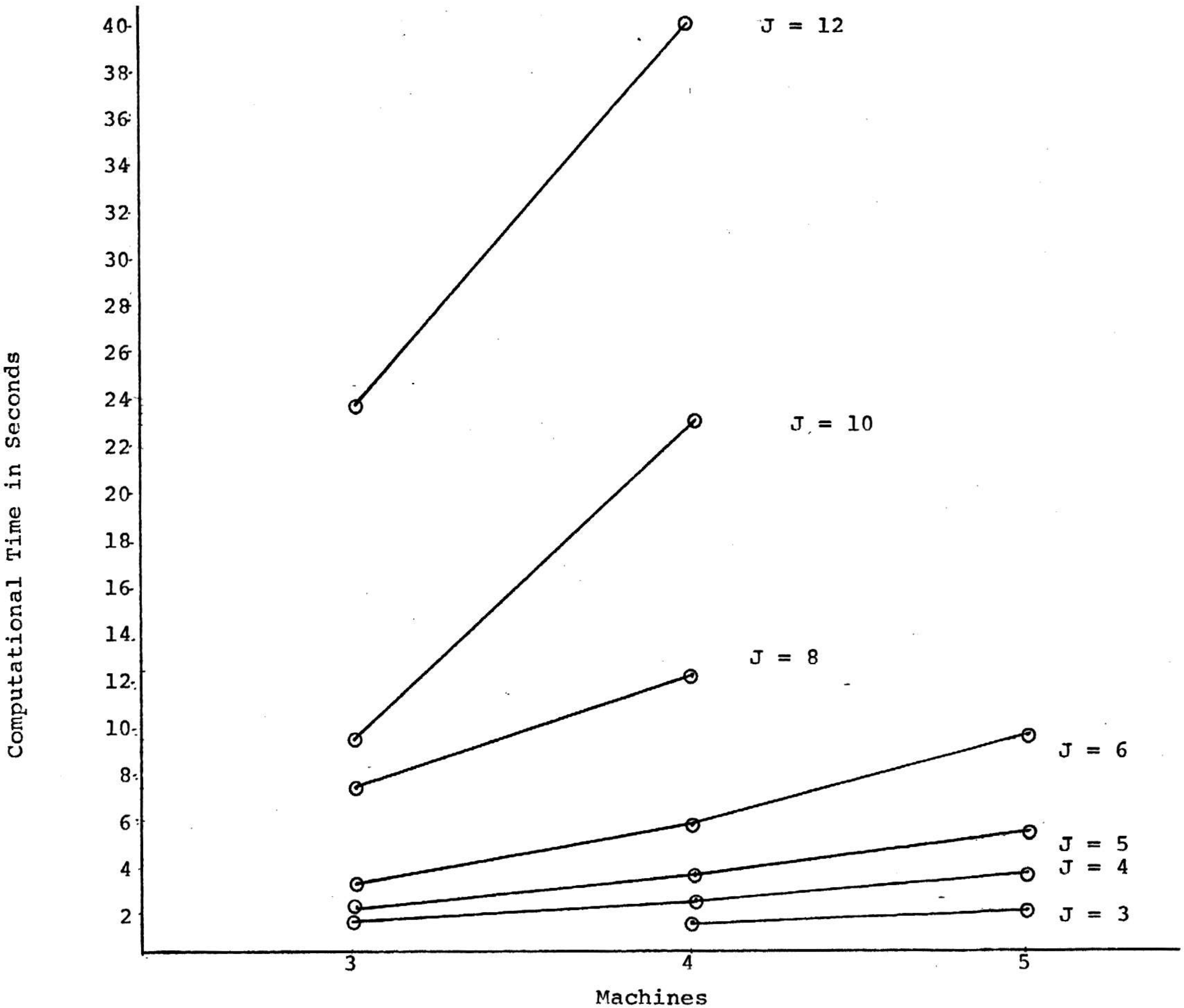


Figure 4.4. Relationship between computational times and number of machines.

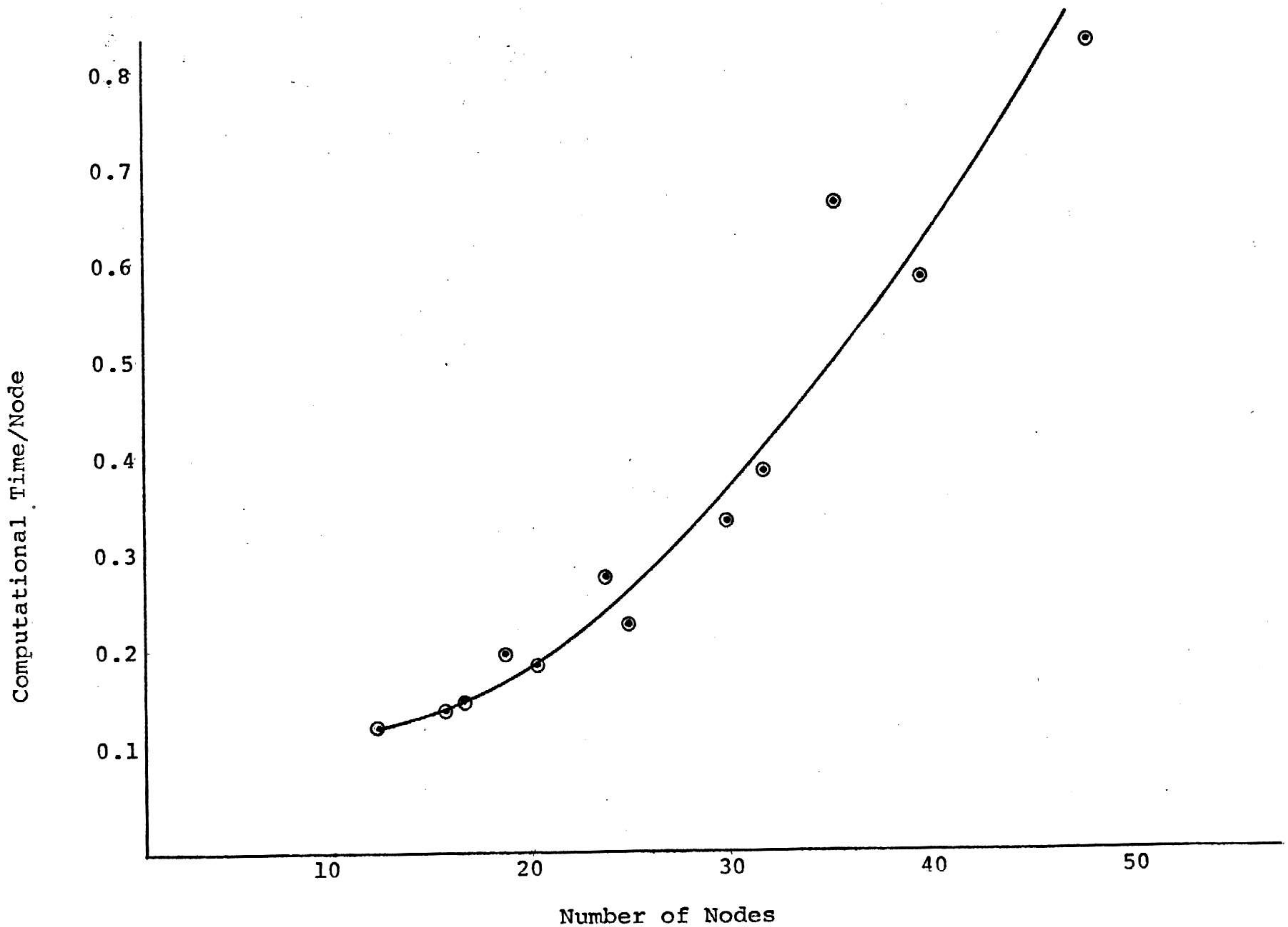


Figure 4.5. Relationship between the number of nodes and computational time per node.

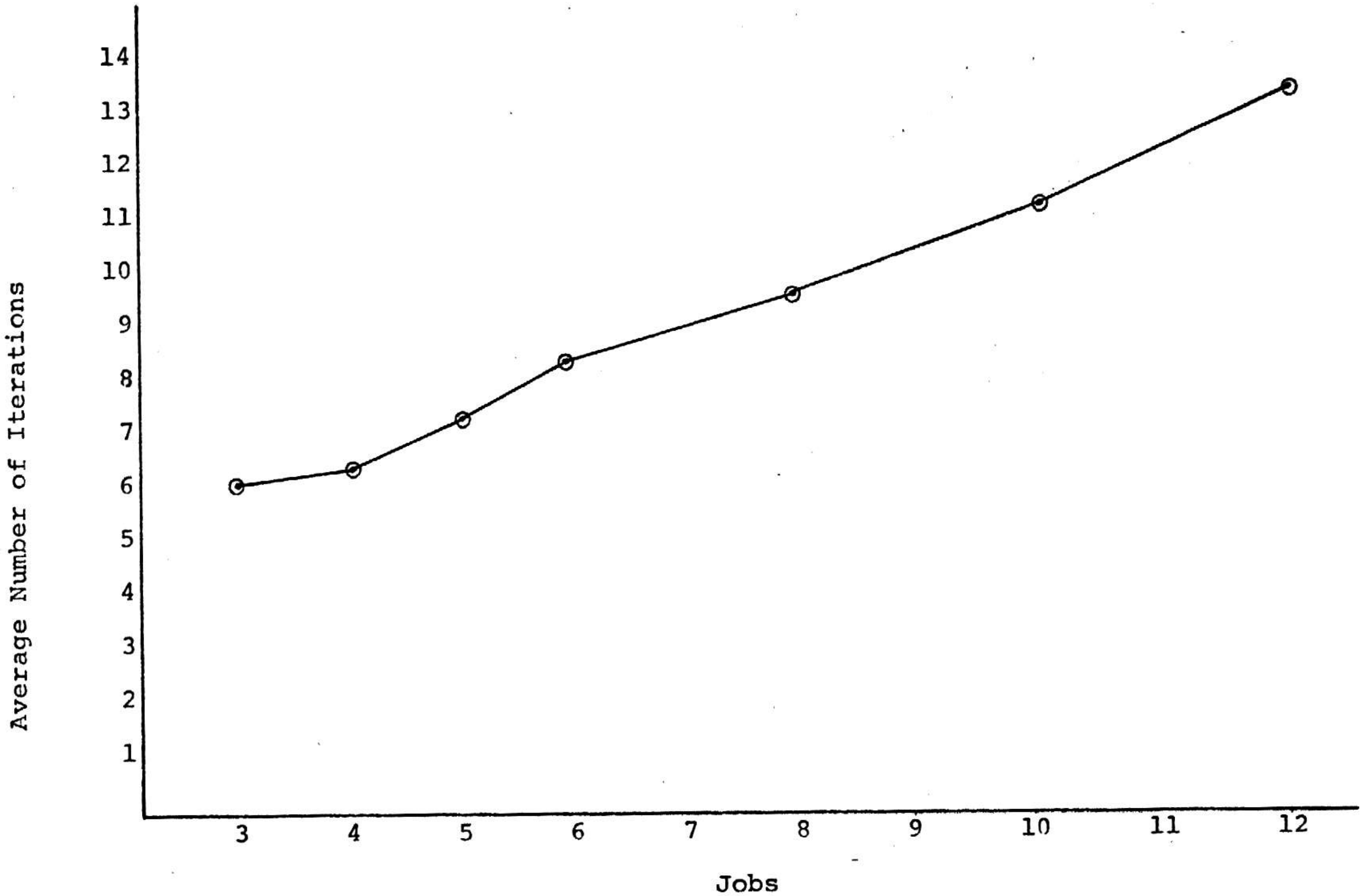


Figure 4.6. Relationship between job size and the number of iterations.

CHAPTER V

Summary and Conclusions

The objective of this thesis is to present an algorithm which is based upon the schedule algebra operators and which is used to solve combinatorial problems. The immediate application of the algorithm is shown in the job shop scheduling problem; however, its use is extended to three other types of problems, namely, the traveling salesman, project scheduling, and the explosion problems. In the case of the latter three applications, the use of the algorithm is for demonstration purposes only; however, for the job shop scheduling problems, a fairly rigorous computational experience was obtained. Furthermore, in the case of the job shop problem, a composite-based bound was embodied in the algorithm which is used to improve the solution.

The algorithm employs a network approach, the basic concepts of which are presented in Chapter II. The demonstration of such an approach is made in the case of a simple job shop problem where J jobs are to be sequenced on M machines. Finally, the computational algorithm is presented in formal fashion, using fairly rigorous notation.

The extension of the applicability of the algorithm is presented in Chapter III. The three classes of problems mentioned above are used for demonstration. In the case of the traveling salesman problem, a complete application can be made because of the nature of the problem itself. By considering the traveling salesman problem as nothing more than a job shop problem consisting of only one machine, the solution by the network algorithm is fairly routine. However, in the case of the project scheduling and explosion problems, the network algorithm is employed in a partial capacity. None-theless, in such a partial application, the main point that should be evident, is the flexibility of the algorithm.

The performance of the algorithm is evaluated in Chapter IV. A total of 17 job shop experiments, exercising a wide range of sizes, were run on the IBM 360/50. In addition, four traveling salesman problems were solved. Three factors were considered in the evaluation of the algorithm with respect to the job shop problem: (1) computational time, (2) quality of the solution, and (3) the number of iterations and conflicts experienced in the solution. The results are formalized in numerous tables and figures in Chapter IV. The computational time was found to exhibit a fair linear relationship at the small problem level. However, as problem size, especially the number of jobs, increased, the computational time increased rather rapidly. The effect of computational time per node was investigated and, again, computational time seemed to increase in a non-linear fashion at the large problem level. It is believed that this is due, in part, to the increased computational time involved in resolving conflicts with the composite-based lower bound. Such conflicts increased, obviously, as the job size increased.

The quality of the solutions obtained was computed from the ratio of the network algorithm solution to the optimal solution. Although not graphically evident because of the inconsistency of available optimal solutions, the efficiency of solution should decrease as the number of conflicts increase.

In the case of the number of iterations experienced in obtaining solutions for the various problem sizes, it was found that as the number of jobs, J, increase, the number of iterations seems to be more nearly the minimum which, of course, is J.

While a large number of at least one class of combinatorial problems were solved, the main intent in this work was not one of computational experience. Rather, it was the objective of this thesis to develop a network approach to such problems as those included and to present a formal algorithm which can be used in their solution. Of course, as is exhibited by the algorithm's application to some problems, only partial solutions have been obtained. Consequently, it is in this area that further research has been proposed. Specifically, further work should be done in two immediate areas. The first involves the further applicability of the algorithm to such problems as the delivery and the line-balancing problems, as well as increased applicability in the project scheduling problem. Secondly, further improvement of the network algorithm solution should be considered. At this point in the research, the composite-based lower bound has proved to be the most desirable criteria; however, its applicability has been made in only one type of problem. Finally, it should be pointed out that the possibility of further improvement in the algorithm itself should be investigated.

REFERENCES

1. Ashour, S., Introduction to Scheduling: Concepts, Analyses, and Performances, John Wiley and Sons, New York, N.Y., in press.

2. Ashour, S. and M. N. Quraishi, "Analysis and Comparison of Various Lower-Bounds on Schedule Times for the Solution of Flow-Shop Problems," Proceedings of the American Astronautical Society, to be published.

3. 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, 1-4.

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

5. Giffler, B., "Scheduling General Production Systems Using Schedule Algebra," Naval Research Logistics Quarterly, Vol. 10, No. 3, Set., 1963.

6. Giffler, B., "Schedule Algebra: A Progress Report," Naval Research Logistics Quarterly, Vol. 15, No. 2, June, 1968.

7. Giffler, B., "Schedule Algebras and Their Use in Formulating General Systems Simulation," Chapter 4 in reference.

8. Giffler, B., "Mathematical Solution of Explosion and Scheduling Problems," IBM Research Report RC-128, Yorktown Heights, New York, May, 1959.

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

10. Giffler, B., and G. L. Thompson, "Algorithms for Solving Production Scheduling Problems," Journal of Operations Research, Vol, 8, July, 1960, pp. 487-503.

11. Hiremath, MS Thesis, Kansas State University, January, 1970.

12. Little, J.D., K. G. Murty, D. N. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," Operations Research, Vol. 11 (D 63), pp. 972-989.

13. Muth, J. F., and G. L. Thompson. Industrial Scheduling. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1963.

Related References

14. Cochran, H., MS Thesis, Kansas State University, 1968.

15. Karg, R. L., and G. L. Thompson, "A Heuristic Approach to Solving Traveling Salesman Problems", Management Science, vol. 10, no. 2, January, 1964, pp. 225-48.

APPENDIX A

Schedule Algebra Theory

The theory upon which the schedule algebra is built is based on certain fundamental concepts found specifically in the scheduling problem and more generally in the combinatorial problem. Beginning with the basic concept of the precedence relationship, the theory for the algebra is developed to a point where the operators can be presented in a formal fashion. After the schedule algebra is presented, an adaptation of the algebra known as star algebra is presented in a similar manner.

A.1. Precedence relationships:

One of the very basic concepts pertaining to the theory of the schedule algebra and the consequent formulation of the operators, is that of the precedence relationship. This concept can best be explained by considering the following network.

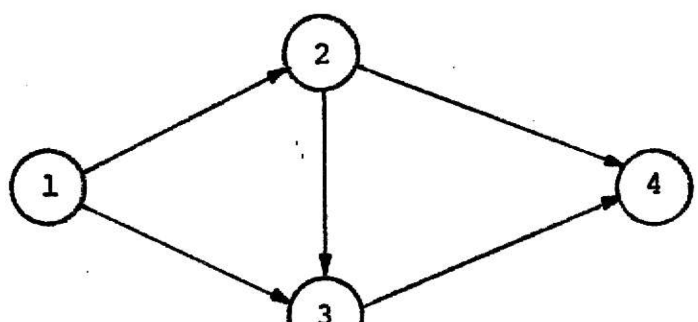


Figure A.1. A Typical Network

It can be seen from Figure A.1. that there are four nodes, each being connected to at least one other node in the network. It is precisely these relationships between connecting nodes that define the nature of the precedence relationships.

When a node must begin at the same time as or before another node, it is said to precede that node. From Figure A.1., it can be seen that node (1) precedes all other nodes in the network. Node (4), on the other hand, does not precede any nodes. If a node must begin before another node with no other nodes between them, the first node is said to directly precede the second. This relationship is seen to exist between nodes (1) and (2), (1) and (3), (2) and (3), (2) and (4), and between (3) and (4).

The relationships of precedence and direct precedence can be symbolized by adopting the following convention. If a node (i) precedes a node (j), it shall be denoted as follows:

$$(i) < (j).$$

While not pointed out earlier, a node can be taken to precede itself, or

$$(i) < (i)$$

If a node (i) is to begin before node (j), with no other nodes in between, then node (i) is said to directly precede node (j), or

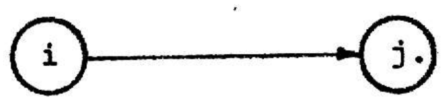
(i) << (j).

Unlike the precedence relationship, a node does not directly precede itself. That is,

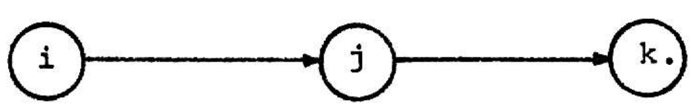
(i) </< (i).

Nevertheless, it can be seen that with the exception of a node directly preceding itself, the set of all direct precedence relationships is included in the set of precedence relationships.

<u>Precedence chains</u>. The concept of the chain relationship is basically a simple one. Such a relationship is evident when two nodes are connected by one or more branches in a network. The length of the chain is dependent upon the number of connecting branches. Furthermore, the lengths are referred to by levels, such that the chains can be 0, 1, 2, or, in general, of level L. For example, a one-level chain can be illustrated as follows:



This relationship illustrates the direct precedence of node (i) to node (j) and, in general, points out that all direct precedence relationships are constructed of one-level chains. Consider the case below:



Obviously, node (i)<<(j), (j)<<(k), and (i)<(k). This is a two-level chain, but more importantly, is the result of two one-level chains or two direct precedence relationships. Consequently, any L-level chain is the result of L one-level chains, which in turn, can be interpreted as L direct precedence relationships.

Finally, a chain of level zero occurs when a node precedes itself. That is, when the following relationship is evident:

$$(i) < (i)$$

Consultation of the network in Figure A.1., shows that there are four 0-level chains, five 1-level chains, four 2-level chains, and one 3-level chain. These chains and their precedence interpretations can be seen in the following summary.

0-level	1-level	2-level	3-level
1	1<<2	1<<2<<4	1<<2<<3<<4
2	1<<3	1<<2<<3	
3	2<<3	2<<3<<4	
4	2<<4	1<<3<<4	
	3<<4		

<u>The precedence matrix</u>. A precedence matrix is nothing more than an arrangement, in matrix form, of the direct precedence relationships that exist for a particular network. The matrix is of size (n x n) where n is the number of nodes

in the network. Entries n(i,j) are made in the matrix N, where node (i) directly precedes node (j). The value of the entry is made with respect to two cases, such that

$$n_{(i,j)} = \begin{cases} 0, & \text{if } (i) < / < (j) \\ 1, & \text{if } (i) << (j). \end{cases}$$

By considering the network of Figure A.1., the following precedence matrix can be constructed.

$$N = \begin{array}{c} \\ (1) \\ (2) \\ (3) \\ (4) \end{array} \begin{array}{c} \begin{matrix} (1) & (2) & (3) & (4) \end{matrix} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

Reading from the matrix, node (1) directly precedes node (2) and (3). In like manner, node (2) directly precedes nodes (3) and (4), while node (3) directly precedes node (4) and node (4) directly precedes no node. Nonetheless, the matrix N, represents all of the 1-level chain relationships associated with the network.

In multiplying the matrix N by itself, we get:

$$N^2 = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} .$$

The entries $n^2_{(i,j)}$ in N^2, represent all existing chains of level 2. That is, there exists one 2-level chain from node (1) to node (3). Two 2-level chains from (1) to (4) and one 2-level chain from (2) to (4). These relationships do, indeed, exist as can be seen from the network.

If N^2 shows all 2-level relationships, it follows that N^3 shows all 3-level chain relationships.

$$N^3 = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}.$$

As the resultant matrix, N^3, shows and as the network verifies, there is only one 3-level chain and that is from node (1) to node (4).

In general, the number of L-level chains between two nodes (i) and (j), can be found by taking the L-th power of the precedence matrix, N. Moreover, the total paths, P; that is, the total of all paths of any level, from (i) to (j) is simply the sum of all powers of N, such that

$$\begin{aligned} P &= N^0 + N^1 + N^2 + \ldots + N^L \\ &= I + N + N^2 + \ldots + N^L, \end{aligned}$$

where I is the identity matrix which is formed by the precedence of each node to itself. After the matrix has been raised to the (L + 1)-th power, it will become a null matrix. Such a matrix signifies the non-existence of any paths of level L + 1 or higher. If N^4 is computed in the above example, it will be found to be a null matrix. Obviously, there are no paths of length four in the network.

The concept regarding the number of paths between pairs of nodes is an important one and should be discussed in some depth. Let us once again consider the matrix N^2, such that

$$N^2 = \begin{bmatrix} 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} .$$

The elements in the matrix are simply the result of conventional matrix multiplication, while the values of the elements were obtained with conventional arithmetic. For example, the element $n^2{}_{(1,4)}$ whose value is seen to be 2, was obtained when the N matrix was squared. That is,

$$n^2{}_{(1,4)} = [(n_{(1,1)} \cdot n_{(1,4)})] + [(n_{(1,2)} \cdot n_{(2,4)})] + [(n_{(1,3)} \cdot n_{(3,4)})] + [(n_{(1,4)} \cdot n_{(4,4)})] .$$

This indicates that the first sum is really concerned with the transitive relationship between the precedence of node (1) to node (4). The second, the transitive path from nodes (1) to (2) and (2) to (4). The others involve this same type of consideration. Nevertheless, in all four cases, the existence of a path from node (1) to node (4) is checked. Further, such a path exists only when the two direct precedence relationships that might compose the path exist. In the Zero-One notation, such a path exists when both components of the multiplication are 1. If either are 0, the path does not exist. By checking the four multiplications involved in arriving at the entry $n^2_{(1,4)}$, one can compute

$$\begin{aligned} n^2_{(1,4)} &= (0)\cdot(0) + (1)\cdot(1) + (1)\cdot(1) + (0)\cdot(0) \\ &= 0 + 1 + 1 + 0 \\ &= 2. \end{aligned}$$

Obviously, the only paths that exist between (1) and (4) that are composed of two direct precedence relationships are (1)<<(2)<<(4), and (1)<<(3)<<(4). Consequently, the number of paths existing between various pairs of nodes can be computed easily and in a logical fashion using conventional matrix multiplication and, of course, conventional addition and multiplication.

Quantitative aspect. Thus far, the main consideration has been given to simply counting the number of paths between nodes in a network. However, this analysis can be extended to include the measurement of the lengths of various paths in the network.

Let us reconstruct the original network and include the times to transverse each direct precedence path.

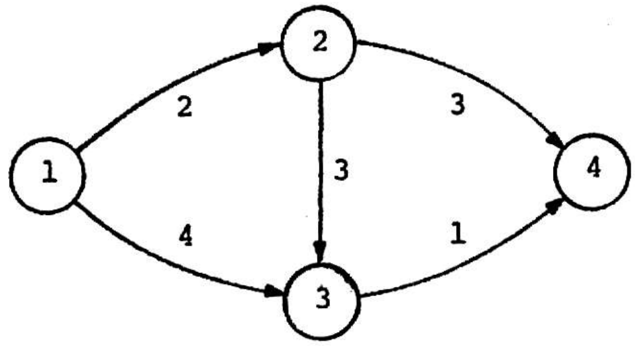


Figure A.2. Typical Network with Branch Lengths Affixed.

By constructing the precedence matrix for the above network using path lengths rather than the Zero-One notation, the following formulation can be made:

$$N = \begin{array}{c} \\ (1) \\ (2) \\ (3) \\ (4) \end{array} \begin{array}{c} \begin{array}{cccc} (1) & (2) & (3) & (4) \end{array} \\ \begin{bmatrix} 0 & 2 & 4 & 0 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} .$$

Interpretation of the matrix N , is similar to that made earlier, except each entry, $n^2_{(i,j)}$ is extended to represent an element set. This element set contains the lengths of all paths between the nodes (i) and (j). This can be illustrated by computing the matrix N^2, and examining the entries of the resultant:

$$N^2 = \begin{array}{c} \\ (1) \\ (2) \\ (3) \\ (4) \end{array} \begin{array}{c} \begin{array}{cccc} (1) & (2) & (3) & (4) \end{array} \\ \begin{bmatrix} 0 & 0 & 5 & 5,5 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{array} .$$

Consider the element $n^2_{(1,4)}$ which is the element set (5,5). This entry signifies that there are two 2-level chain relationships from node (1) to node (4) each having a length of 5. A quick check of the network shows that such relationships and corresponding lengths do exist.

Noting that the discussion earlier, concerning the number of paths between nodes called for conventional matrix operations and more specifically, conventional arithmetic operations, it should be readily noticeable that the element sets above were n ot the result of such computation. Had the normal operations of multiplication and addition been performed, the element $n^2_{(1,4)}$ would have yielded,

$$\begin{aligned} n_{(1,4)} &= (n_{(1,1)} \cdot n_{(1,4)}) + (n_{(1,2)} \cdot n_{(2,4)}) + \\ &\quad (n_{(1,3)} \cdot n_{(3,4)}) + (n_{(1,4)} \cdot n_{(4,4)}) \\ &= (0) \cdot (0) + (2) \cdot (3) + (4) \cdot (1) + (0) \cdot (0) \\ &= 6 + 4 \\ &= 10 \end{aligned}$$

The element 10 would have been meaningless, since the main concern is the length of the paths from node (1) to node (4). However, by looking at the network, it is evident that to obtain the length of the paths between node (1) and (4), one should add the length of the 1-level chains that compose the desired paths, which for N^2 are, of course, chains of level two. Consequently, by reformulating the computational procedure used in obtaining $n^2_{(1,4)}$, one can write

$$(n_{(1,1)} \odot n_{(1,4)}) \oplus (n_{(1,2)} \odot n_{(2,4)}) \oplus (n_{(1,3)} \odot n_{(3,4)}) \oplus (n_{(1,4)} \odot n_{(4,4)})$$

where the symbol $\odot$ is taken to signify conventional addition. Therefore, the resulting values from the operations defined in the parentheses are (0+0), (2+3), (4+1), and (0+0). Once the corresponding elements in the matrices have been multiplied (added conventionally), they are then combined into an element set as mentioned previously. Such a combination which is signified by the symbol $\oplus$, can be illustrated by completing the computation of $n^2_{(1,4)}$ such that

$$\begin{aligned} n^2_{(1,4)} &= (0+0) \oplus (2+3) \oplus (4+1) \oplus (0+0) \\ &= (0, 5, 5, 0) \\ &= (5, 5). \end{aligned}$$

The zero elements are omitted; however, the two elements (5,5) are the lengths of the two different 2-level chains from node (1) to node (4).

This example points out the computational derivation for the special forms of addition and multiplication. These special forms have been given the names of schedule algebra addition and schedule algebra multiplication. They can be presented formally by the following formulation:

$$n^2_{(i,j)} = \sum_k (n_{(i,k)} \odot n_{(k,j)})$$

where $\odot$ implies schedule algebra multiplication and the summation over k refers to schedule algebra addition, $\oplus$, or to the combination of the schedule algebra products into element sets. Finally, it should be pointed out that the above formulation is not general, for it refers only to those relationships of level 2. This is indicated by the power of 2 to which $n_{(i,j)}$ is raised; however, the formulation with respect to the operators of addition and multiplication is valid in all cases.

A.2. Schedule Algebra Operators:

As implied in the earlier discussion, schedule algebra is identical to conventional matrix algebra with respect to matrix operations such as addition and multiplication. However, it differs from matrix algebra in its characterization of a matrix and in the arithmetic operations involving elements of the matrices.

Characteristics of matrices. Matrices in schedule algebra can be considered to be arrays of element sets. The entries within these sets are called elements. Such elements take the form of the usual numbers or ratios of such numbers. The only exception is the addition of the element $\pm\iota$. This term represents numerical zero or zero magnitude. With reference to earlier discussion, this term would be the quantifying element for chains of level 0.

Schedule algebra matrices are signified by capital letters and are, as usual, enclosed by brackets. The entries in the matrices; that is, the element sets are identified by lower case double subscripted letters. Consequently, the identity, with few exceptions, of the schedule algebra matrices is very similar to that of conventional matrices. Nevertheless, an example of a schedule algebra matrix can be constructed as follows:

$$A = \begin{bmatrix} (6,\ 4) & (8,\ 4) \\ (0) & (-2,2) \\ (\ ,\ 1) & (1,\ 1) \end{bmatrix} .$$

Before the operators are presented, it should be pointed out that there are identity matrices in schedule algebra just as in conventional matrix algebra. These identities are given below, for addition and multiplication respectively:

$$\begin{bmatrix} (0) & (0) \\ (0) & (0) \end{bmatrix} , \quad \begin{bmatrix} (\iota) & (0) \\ (0) & (\iota) \end{bmatrix}$$

<u>Schedule algebra addition</u>. The symbol for schedule algebra addition, as pointed out earlier, is $\oplus$. The procedure for addition can be presented in three steps:

1 - Combine all entries of the element sets to be added into one set.

2 - Delete all pairs of elements which are identical in magnitude but opposite in sign and replace with a zero and

3 - Delete all zeros if the set contains at least one element which is not zero; however, if nothing but zeros remain, reduce the set to only one element of zero.

Consider the following examples:

$$(\iota,\ 1,\ 4) \oplus (6,\ -1) = (\iota,\ 6,\ 0,\ 4)$$
$$\neq (\iota,\ 6,\ 4)$$

and

$$(1,\ -3) \oplus (-1,\ 3) = (0,\ 0)$$
$$= (0).$$

<u>Schedule algebra multiplication</u>. The symbol for schedule algebra multiplication is $\odot$. The rules for schedule algebra multiplication can best be summarized in the following manner:

$$x \odot y = \begin{cases} 0, \text{ if } x = 0 \text{ or } y = 0, \\ |x| + |y| \text{ , if } x,y \neq 0 \text{ and have the same sign,} \\ -[|x| + |y|] \text{ , if } x,y \neq 0 \text{ and have opposite signs,} \\ + y, \text{ if } x = \pm\iota \text{, and } y \neq 0. \end{cases}$$

The symbol, + that appears in the second and third cases above, implies conventional addition and not schedule algebra addition. Consider the following:

$$6 \odot 7 = 13$$
$$4 \odot -7 = -11$$
$$2 \odot 0 = 0$$
$$8 \odot \iota = 8$$
$$\iota \odot -\iota = -\iota$$

The above formulation and examples describe multiplication of elements. To multiply two element sets; however, is to form the cross products of the elements of the sets. Of course, the resultant set is that obtained from the schedule algebra addition of cross products. Consider the following examples:

$$(1, \iota) \odot (2, 3) = (3, 2, 4, 3)$$

$$(1, \iota) \odot (-2, -\iota) = (-3, -2, -1, -\iota).$$

<u>Schedule algebra subtraction</u>. Once the multiplication operation has been discussed, the operation of subtraction can follow such that,

$$(x) \ominus (y) = (x) \oplus (-\iota) \odot [(y), I,$$

where for example

$$(6, \iota) \ominus (4, -6, \iota) = (6, \iota) \oplus (-4, 6, -\iota)$$
$$= (-4, 6, 6).$$

<u>Schedule algebra division</u>. By considering the schedule algebra operation of division as an operation involving ratios of integers, the following rules can be presented. Consider two non-zero integers x and y such that

$$x / y = \begin{cases} |x| - |y| = z, & \text{if } x>y\text{, and both x and y have the same signs,} \\ -z, & \text{if } x>y\text{, and x and y have different signs,} \\ \iota / y - x, & \text{if } y>x\text{, and x and y have the same signs,} \\ -\iota / y - x, & \text{if } y>x\text{, and x and y have different signs,} \\ \iota, & \text{if } x = y \text{ and both x andy have the same sign} \\ -\iota, & \text{if } x = y\text{, and x and y have different signs.} \end{cases}$$

In general, schedule algebra division is defined for all ratios x/y in which $y \neq 0$. However, if $x = 0$, then $0/y$ is equal to 0 for all $y \neq 0$. It should be noted that the subtraction operation used above is one of a conventional nature. Consider the following examples for the ratio of two integers:

$$
\begin{aligned}
6/\ 4 &= 6 - 4 = 2, \\
6/\ -4 &= -(6 - 4) = -2, \\
4/\ 6 &= \iota/6 - 4 = \iota/\ 2, \\
4/\ -6 &= -\iota/\ 6 - 4 = -\iota/\ 2, \\
5/\ 5 &= \iota \\
-5/\ 5 &= -\iota
\end{aligned}
$$

By defining the operation of division, it follows that every set must have an inverse of multiplication (every non-empty set). Consider the following:

$$
\begin{aligned}
(4)^{-1} &= \iota/\ 4, \\
(\iota/\iota,\ -1)^{-1} &= (\iota,\ -1).
\end{aligned}
$$

<u>Matrix operations</u>. The operations and rules of matrix algebra hold true for schedule algebra matrices in the same manner as for conventional matrices. Of course, the arithmetic involved in manipulation is unique, based on the operators presented above. For example, consider the sum of the following two matrices:

$$\begin{bmatrix} (\iota) & (5,\ 1) \\ (6,\ \iota) & (-3,\ 1) \end{bmatrix} + \begin{bmatrix} (4) & (0) \\ (\iota,\ 2) & (\iota) \end{bmatrix} = \begin{bmatrix} (\iota,\ 4) & (5,\ 1) \\ (6,\ \iota,\ \iota,\ 2) & (-3,\ 1,\ \iota) \end{bmatrix} .$$

The product of the same two matrices is

$$\begin{bmatrix} (4) \oplus (5,\ 7,\ 1,\ 3) & (0) \oplus (5,\ 1) \\ (10,\ 4) \oplus (-3,\ -5,\ 1,\ 3) & (-3,\ 1) \end{bmatrix}$$

or

$$\begin{bmatrix} (4,\ 5,\ 7,\ 1,\ 3) & (5,\ 1) \\ (10,\ 4,\ -5,\ 1) & (-3,\ 1) \end{bmatrix}$$

A.3. Star Algebra Operators:

As Giffler has pointed out, it is not always computationally feasible to keep all elements in an element set. Rather, only those elements that are the maximum in each set are maintained. Such a maximizing rule for addition has led to the formulation of the star algebra. Star algebra is equivalent to conventional matrix algebra with respect to its matrix operations. The primary difference is that all matrices under star algebra are non-negative or zero.

Star algebra addition. The operation for addition of elements under star algebra can be formulated as follows:

$$x * y = \max (x, y),$$

where the symbol $*$ has replaced the schedule algebra symbol for addition of $\oplus$.

Star algebra multiplication. Star algebra multiplication can be given as follows:

$$(x, y) \# (v, z) = \max (x \odot v, x \odot z, y \odot v, y \odot z),$$

where the symbol, #, has replaced the schedule algebraic symbol of $\odot$, for multiplication.

The following sample problem illustrates the use of the star algebra.

$$\begin{bmatrix} 6 & 4 \\ \iota & 3 \end{bmatrix} \# \begin{bmatrix} 5 & 2 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \max(5 \odot 6,\ 4 \odot 0) & \max(2 \odot 6,\ 4 \odot 1) \\ \max(\iota \odot 5,\ 3 \odot 0) & \max(2 \odot \iota,\ 3 \odot 1) \end{bmatrix}$$

$$\begin{bmatrix} \max(11,\ 0) & \max(8,\ 5) \\ \max(5,\ 0) & \max(2,\ 4) \end{bmatrix}$$

$$\begin{bmatrix} 11 & 8 \\ 5 & 4 \end{bmatrix}$$

APPENDIX B

A Schedule Algebra Algorithm

As was mentioned in Chapter II, the research for this thesis was instigated by the schedule algebra algorithm as formulated by Giffler, Of course, the emphasis eventually became centered around the network approach and its development. However, the network approach was an outgrowth of the schedule algebra algorithm, and, as such, it is logical that the basic concept of the schedule algebra algorithm be presented in a formal fashion. This appendix is made up of three sections which are identical to those maintained for the discussion of the network approach. They are the basic concepts of the algorithm, a sample problem, and a formal presentation of the computational algorithm.

B.1. Basic Concepts:

The schedule algebra technique, much like the network approach, is a systematic approach which searches a subset of feasible sequences for a solution. The basic concept of this approach can be broken down into the same three areas as were used in the discussion of the network algorithm. They are (1) the representation of the problem in a precedence matrix, (2) the manipulation of the matrix based on the star algebra operators, and (3) the evaluation of the resulting sequence to obtain the corresponding schedule time.

Representation of the problem by a precedence matrix and the corresponding construction of such, is identical to that presented in Chapter II. The matrix is partitioned into M machine blocks, each having J rows and J columns. Entries are made in the matrix in the same manner as in the network approach. The concepts of partial ordering and the possible direct precedence relationship remain unchanged. Consequently, both techniques employ the same initial precedence matrix. With respect to the schedule algebra algorithm, this matrix is referred to as S.

Once the precedence matrix is constructed, the process of entry can begin. The technique for entry is the same in concept as that used in the network approach. However, the method of entry is somewhat different. Nodes are selected to enter one at a time. That is, only one node can enter per iteration with the schedule algebra. Nonetheless, a node is a candidate for entry if its column is null or potentially null. If there are more than one such candidates for entry, the conflicts are resolved with a particular bounding procedure. This procedure makes use of a bound which is really an evaluation of the earliest machine available time and is referred to as the FACAT (facility available time). The value of the FACAT represents the earliest time that a particular machine will be available after processing the job associated with the node in question. Obviously, the node with the earliest FACAT is chosen to next start.

When a node is chosen to be entered into solution, its column is updated as follows:

$$xs_{(j\,m_\ell,\; j\,m_s)} = 0,$$
$$ys_{(j\,m_\ell,\; j\,m_s)} = s_{(j\,m_\ell,\; j\,m_s)},$$

where all other terms with y's in the same row as the y-term above, are made 0, and all elements in the corresponding row of the updated column are updated such that

$$xs_{(j\,m_\ell,\; j\,m_s)} = y_{(j\,m_\ell,\; j\,m_s)}.$$

As can be seen, a new term has been introduced. The concept of the y-term is one of a transitory nature. That is, a y-term is taken to imply an x-term that will eventually become 0 or ι. When the y's appear in the matrix, they act as ι's and behave as such in all computations.

Once a node has entered the solution and the matrix has been updated accordingly, the starting time vector is updated. This procedure represents another phase of the schedule algebra algorithm that is in contrast to the network approach. Before, the starting vector was used only when the final precedence matrix was obtained; however, here the vector is used after each update to the precedence matrix. The concept and construction of the starting vector is identical to that discussed earlier.

In general, the procedure involved in the algorithm would entail choosing a node for entry, updating the precedence matrix, and finally, updating the starting vector. At this point, the process would begin again, being completed, of course, when all nodes had been entered. Finally, the schedule time for a sequence must be obtained. The computation of this schedule time follows identically the procedure outlined in Chapter II.

B.2. Sample Problem:

Consider again the sample problem solved in Chapter II. The corresponding machine ordering and processing time matrices are reproduced below for convenience:

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

Step 1. Construct the initial starting vector, T°, such that

$$T^\circ = [\ 0\ \iota\ 0\ \iota\ \iota\ 0\ 0\ 0\ 0\ 0\ \iota\ 0\ ]\ ,$$

where the ι's signify the earliest starting times of nodes (21), (41), (12), and (33).

Step 2. Construct the precedence matrix, S°, from the partial orderings and possible direct precedence relationships.

Step 3. Check for nodes to enter. It can be seen from S° that columns (21), (41), (12), and (33) are potentially null. Since there are four nodes competing for entry, the FACATS, $A_{(j\ m_\ell)}$, must be computed for each entry such that

$$A_{(j\ m_\ell)} = \tau_{(j\ m_\ell)} + t_{(j\ m_\ell)}\ .$$

The FACATS for the nodes can be computed as follows:

$$A_{(21)} = \iota + 8 = 8,$$

$$A_{(41)} = \iota + 7 = 7,$$

$$A_{(12)} = \iota + 4 = 4,$$

$$A_{(33)} = \iota + 6 = 6.$$

Since column (12) has the minimum FACAT, it is selected to enter.

Step 4. Update the precedence matrix by entering the node just selected in step 3. By making column (12) null and, further, updating the matrix as described in the basic concept, the matrix S^1, can be computed.

Step 5. Update the starting vector. When the T^0 vector is multiplied by the matrix S^1, the resultant can be given as

$$[\ 0\ 0\ 6\ 0\ 0\ 4\ 4\ 7\ 4\ 8\ 0\ 0\ ].$$

Upon adding this vector to T^0, the resultant vector, T^1, becomes

$$T^1 = [\ 0\ \iota\ 6\ \iota\ \iota\ 4\ 4\ 7\ 4\ 8\ \iota\ 0\ ].$$

Step 6. Repeat step 3 by checking for the next node to enter. After checking S^1, picking the candidates for entry, and evaluating the FACATS of each node, (33) was chosen to enter next. The updated matrix becomes S^2 and the updated starting vector becomes T^2. This procedure is repeated until all nodes have been entered. The entry of the nodes can be given in the following order, beginning with the third node to enter, (41).

The order of entry is (13), (31), (42), (11), (21), (43), (23), (32), and (22). It should be pointed out that there were some ties in FACATS; consequently, these ties have been broken by random. The precedence matrices and the corresponding starting vectors are presented after each iteration.

Step 7. Calculate the final sequence and the schedule time. By arranging the jobs in each machine block of the final starting vector, T^{12}, with respect to starting times, we can formulate the following sequences on each machine:

machine 1: { 4 3 1 2 }

machine 2: { 1 4 3 2 }

machine 3: { 3 1 4 2 }

The job sequencing matrix, S can be constructed such that,

$$S = \begin{bmatrix} 41 & 31 & 11 & 21 \\ 12 & 42 & 32 & 22 \\ 33 & 13 & 43 & 23 \end{bmatrix}$$

Upon locating the operations in each machine block which have the highest starting times, we can obtain nodes (21), (22), and (23). When the processing times of each node are added to the starting times, the results are 21, 30, and 25 time units, respectively. Consequently, the schedule time for the sequence is 30. Note that the optimal schedule time is 27.

Table B.1. Initial Precedence Matrix, S°.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x3	x3	x3	0	0	0	0	0	0	0	0
(21)	x8	0	x8	x8	0	0	0	0	0	8	0	0
(31)	x3	x3	0	x3	0	0	3	0	0	0	0	0
(41)	x7	x7	x7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	x4	x4	x4	4	0	0	0
(22)	0	0	0	0	x5	0	x5	x5	0	0	0	0
(32)	0	0	0	0	x9	x9	0	x9	0	0	0	0
(42)	0	0	0	0	x6	x6	x6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	x2	x2	x2
(23)	0	0	0	0	0	4	0	0	x4	0	x4	x4
(33)	0	0	6	0	0	0	0	0	x6	x6	0	x6
(43)	0	0	0	0	0	0	0	0	x2	x2	x2	0

Table B.2. Intermediate Precedence Matrix, S^1.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x3	x3	x3	0	0	0	0	0	0	0	0
(21)	x8	0	x8	x8	0	0	0	0	0	8	0	0
(31)	x3	x3	0	x3	0	0	3	0	0	0	0	0
(41)	x7	x7	x7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	y4	y4	y4	4	0	0	0
(22)	0	0	0	0	0	0	x5	x5	0	0	0	0
(32)	0	0	0	0	0	x9	0	x9	0	0	0	0
(42)	0	0	0	0	0	x6	6	x0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	x2	x2	x2
(23)	0	0	0	0	0	4	0	0	x4	0	x4	x4
(33)	0	0	6	0	0	0	0	0	x6	x6	0	x6
(43)	0	0	0	0	0	0	0	0	x2	x2	x2	0

Table B.3. Intermediate Precedence Matrix, S^2 and Starting Time Vector, T^2.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x^3	x^3	x^3	0	0	0	0	0	0	0	0
(21)	x8	0	x8	x8	0	0	0	0	0	8	0	0
(31)	x3	x3	0	x3	0	0	3	0	0	0	0	0
(41)	x7	x7	x7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	y4	y4	y4	4	0	0	0
(22)	0	0	0	0	0	0	x5	x5	0	0	0	0
(32)	0	0	0	0	0	x9	0	x9	0	0	0	0
(42)	0	0	0	0	0	x6	x6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	x2	0	x2
(23)	0	0	0	0	0	4	0	0	x4	0	0	x4
(33)	0	0	6	0	0	0	0	0	y6	y6	0	y6
(43)	0	0	0	0	0	0	0	0	x2	x2	0	0

and

$$\begin{bmatrix} 6 & \iota & 6 & \iota & \iota & 12 & 9 & 7 & 6 & 8 & \iota & 13 \end{bmatrix}$$

Table B.4. Intermediate Precedence Matrix, S^3
and Starting Time Vector, T^3.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x3	x3	0	0	0	0	0	0	0	0	0
(21)	x8	0	x8	0	0	0	0	0	0	8	0	0
(31)	x3	x3	0	0	0	0	3	0	0	0	0	0
(41)	y7	y7	y7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	y4	y4	y4	4	0	0	0
(22)	0	0	0	0	0	0	x5	x5	0	0	0	0
(32)	0	0	0	0	0	x9	0	x9	0	0	0	0
(42)	0	0	0	0	0	x6	x6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	x2	0	x2
(23)	0	0	0	0	0	4	0	0	x4	0	0	x4
(33)	0	0	6	0	0	0	0	0	y6	y6	0	y6
(43)	0	0	0	0	0	0	0	0	x2	x2	0	0

and

8	7	7	ι	ι	12	9	7	6	8	ι	13

Table B.5. Intermediate Precedence Matrix, S^4 and Starting Time Vector, T^4.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x3	x3	0	0	0	0	0	0	0	0	0
(21)	x8	0	x8	0	0	0	0	0	0	8	0	0
(31)	x3	x3	0	0	0	0	0	3	0	0	0	0
(41)	y7	y7	y7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	y4	y4	y4	4	0	0	0
(22)	0	0	0	0	0	0	x5	x5	0	0	0	0
(32)	0	0	0	0	0	x9	0	x9	0	0	0	0
(42)	0	0	0	0	0	x6	x6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	y2	0	y2
(23)	0	0	0	0	0	4	0	0	0	0	0	x4
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	x2	0	0

and

8	7	7	ι	ι	12	10	7	6	15	ι	13

Table B.6. Intermediate Precedence Matrix, S^5 and Starting Time Vector, T^5.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x3	0	0	0	0	0	0	0	0	0	0
(21)	x8	0	0	0	0	0	0	0	0	8	0	0
(31)	y3	y3	0	0	0	0	0	3	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	y4	y4	y4	4	0	0	0
(22)	0	0	0	0	0	0	x5	x5	0	0	0	0
(32)	0	0	0	0	0	x9	0	x9	0	0	0	0
(42)	0	0	0	0	0	x6	x6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	y2	0	y2
(23)	0	0	0	0	0	4	0	0	0	0	0	x4
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	x2	0	0

and

10	10	7	ι	ι	19	10	7	6	15	ι	13

Table B.7. Intermediate Precedence Matrix, S^6 and Starting Time Vector, T^6.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	x3	0	0	0	0	0	0	0	0	0	0
(21)	x8	0	0	0	0	0	0	0	0	8	0	0
(31)	y3	y3	0	0	0	0	0	3	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	x5	0	0	0	0	0
(32)	0	0	0	0	0	x9	0	0	0	0	0	0
(42)	0	0	0	0	0	y6	y6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	y2	0	y2
(23)	0	0	0	0	0	4	0	0	0	0	0	x4
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	x2	0	0

and

10	10	7	ι	ι	19	10	7	6	18	ι	13

Table B.8. Intermediate Precedence Matrix, S^7 and Starting Time Vector, T^7.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	Y3	0	0	0	0	0	0	0	0	0	0
(21)	0	0	0	0	0	0	0	0	0	8	0	0
(31)	3	0	0	0	0	0	0	3	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	x^5	0	0	0	0	0
(32)	0	0	0	0	0	x^9	0	0	0	0	0	0
(42)	0	0	0	0	0	y^6	y^6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	y^2	0	y^2
(23)	0	0	0	0	0	4	0	0	0	0	0	x^4
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	x^2	0	0

and

10	13	7	ι	ι	22	13	7	6	18	ι	13

Table B.9. Intermediate Precedence Matrix, S^8 and Starting Time Vector, T^8.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	3	0	0	0	0	0	0	0	0	0	0
(21)	0	0	0	0	0	0	0	0	0	8	0	0
(31)	3	0	0	0	0	0	0	3	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	x5	0	0	0	0	0
(32)	0	0	0	0	0	x9	0	0	0	0	0	0
(42)	0	0	0	0	0	y6	y6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	y2	0	y2
(23)	0	0	0	0	0	4	0	0	0	0	0	x4
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	x2	0	0

and

10	13	7	ι	ι	22	13	7	6	21	ι	13

Table B.10. Intermediate Precedence Matrix, S^9 and Starting Time Vector, T^9.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	3	0	0	0	0	0	0	0	0	0	0
(21)	0	0	0	0	0	0	0	0	0	8	0	0
(31)	3	0	0	0	0	0	0	3	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	x^5	0	0	0	0	0
(32)	0	0	0	0	0	x^9	0	0	0	0	0	0
(42)	0	0	0	0	0	y^6	y^6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	0	0	2
(23)	0	0	0	0	0	4	0	0	0	0	0	0
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	y^2	0	0

and

10	13	7	ι	ι	25	13	7	6	21	ι	13

Table B.11. Intermediate Precedence Matrix, S^{10} and Starting Time Vector, T^{10}.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	3	0	0	0	0	0	0	0	0	0	0
(21)	0	0	0	0	0	0	0	0	0	8	0	0
(31)	3	0	0	0	0	0	0	3	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	x5	0	0	0	0	0
(32)	0	0	0	0	0	x9	0	0	0	0	0	0
(42)	0	0	0	0	0	y6	y6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	0	0	2
(23)	0	0	0	0	0	4	0	0	0	0	0	0
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	2	0	0

and

10	13	7	ι	ι	25	13	7	6	21	ι	13

Table B.12. Intermediate Precedence Matrix, S^{11} and Starting Time Vector, T^{11}.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	3	0	0	0	0	0	0	0	0	0	0
(21)	0	0	0	0	0	0	0	0	0	8	0	0
(31)	3	0	0	0	0	0	0	3	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	0	0	0	0	0	0
(32)	0	0	0	0	0	9	0	0	0	0	0	0
(42)	0	0	0	0	0	0	6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	0	0	2
(23)	0	0	0	0	0	4	0	0	0	0	0	0
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	2	0	0

and

10	13	7	ι	ι	25	13	7	6	21	ι	13

Table B.13. Final Precedence Matrix, S^{12} and Starting Time Vector, T^{12}.

	(11)	(21)	(31)	(41)	(12)	(22)	(32)	(42)	(13)	(23)	(33)	(43)
(11)	0	3	0	0	0	0	0	0	0	0	0	0
(21)	0	0	0	0	0	0	0	0	0	8	0	0
(31)	3	0	0	0	0	0	0	3	0	0	0	0
(41)	0	0	7	0	0	0	0	7	0	0	0	0
(12)	0	0	0	0	0	0	0	4	4	0	0	0
(22)	0	0	0	0	0	0	0	0	0	0	0	0
(32)	0	0	0	0	0	9	0	0	0	0	0	0
(42)	0	0	0	0	0	0	6	0	0	0	0	6
(13)	2	0	0	0	0	0	0	0	0	0	0	2
(23)	0	0	0	0	0	4	0	0	0	0	0	0
(33)	0	0	6	0	0	0	0	0	6	0	0	0
(43)	0	0	0	0	0	0	0	0	0	2	0	0

and

10	13	7	ι	ι	25	13	7	6	21	ι	13

B.3. Computational Algorithm:

The final phase of this discussion of the schedule algebra algorithm, will be a formal presentation of the algorithm. The following represents a step by step computational procedure which embodies the concepts presented earlier and formalizes the procedure illustrated in the sample problem.

Step 1: Construct the initial starting vector, T^0.

Form a vector with JM entries, such that

$$\tau(j\ m_\ell) = \begin{cases} \iota, & \text{for all } (j\ m_1), \\ 0, & \text{otherwise.} \end{cases}$$

Step 2: Construct the initial precedence matrix, S^0.

2.1 Partition a (JM x JM) matrix into M machine blocks.

2.2 Label the rows and columns of the matrix by the appropriate nodes.

2.3 Place the entries in the matrix such that

$$s(j\ m_\ell,\ j\ m_s) = \begin{cases} t_{(j\ m_\ell)}, & \text{if } (j\ m_\ell) << (j\ m_s), \\ xt_{(j\ m_\ell)}, & \text{if } (j\ m_\ell) << (j\ m_s) \text{ is possible} \\ 0, & \text{otherwise.} \end{cases}$$

Step 3: Check for null or potentially null columns.

3.1 For each null or potentially null column in the S matrix, mark the column.

3.2 If there is more than one marked column, go to step 4.

3.3 If there is only one marked column, go to step 5.

Step 4: Determine the machine available time.

4.1 For each marked column, scan the corresponding row and determine the minimum entry of the form $xs_{(j\ m_\ell)}$.

4.2 Compute the machine available time, $A(j\ m_\ell)$ such that

$$A(j\ m_\ell) = \tau(j m_\ell) + t(j\ m_\ell).$$

4.3 Select the column having minimum A.

4.3.1. If there is a tie, select a column to next start by a particular rule. Remove the marks from the other columns that were considered, and proceed to step 5.

4.3.2. If there is no tie, remove the marks from the other columns being considered and proceed to step 5.

Step 5: Update the precedence matrix.

5.1 Make all entries in the marked column, of the form $xs_{(j\ m_\ell)}$, equal to zero.

5.2 Make all entries in the marked column, of the form $ys_{(j\ m_\ell)}$, equal to $s_{(j\ m_\ell)}$. Make all other y terms in this row equal to zero.

5.3 Make all entries in the corresponding row of the marked column, which are of the form $xs_{(j\ m_\ell)}$, equal to $ys_{(j\ m_\ell)}$.

Step 6: Update the starting vector.

6.1 Multiply the updated S matrix by the starting vector such that

$$T^{L'} = T^{L-1} \# S^{L}, \quad L = 1, 2, \ldots,$$

and add the starting vector to the resultant vector such that

$$T^{L} = T^{L-1} * T^{L'}$$

Step 7: Repeat steps 3-6 until all columns have been entered.

Step 8: Find the sequence and the corresponding schedule time.

8.1 Order the jobs with respect to their starting times within each machine block of the final starting vector.

8.2 Locate the element in each machine block of the final starting vector that has the latest starting time.

8.3 Add the processing time to the starting time of each chosen element.

8.4 Select the operation which results in the greatest amount of time such that

$$T(S) = \max [\tau_{(j\, m_M)} + t_{(j\, m_M)}], \quad j = 1, 2, \ldots, J,$$

where T(S) is the schedule time for the sequence.

APPENDIX C

A Bounding Procedure

It was pointed out in Chapter II that a particular bounding procedure was used to resolve the conflict that resulted in the machine blocks with respect to the order of entry of nodes. Furthermore, it was pointed out that the bounding procedure used was a composite one; however, no formulation was made at that time. Therefore, this appendix has been included to discuss the composite bound used to resolve the conflicts arising in the network algorithm. Moreover, the organization of this discussion will include two sections. They are (1) formulation of the bound and (2) a sample problem illustrating the use of the bound.

C.1. Composite-Based Bound

The lower bound on schedule time for a node can be defined as the sum of the completion times of the scheduled jobs and the total processing times of the unscheduled jobs in addition to an estimation of the idle time which may be experienced between the unscheduled jobs when they are scheduled. Furthermore, the power of a particular bounding procedure is measured in terms of its ability to produce a lower bound that is close to the actual schedule time.

Lower bounds can be used individually or they can be combined, in which case a composite bound is formed. Such a combination of lower bounds was used in this research. That is, a job-based bound and a machine-based bound have been combined to form a composite-based bound. This composite-based bound is presented more rigorously in Hiremath [11].

Before any formulation of the bound can be made, certain notation should be considered. This notation is consistent with that used in [11] and can be presented as follows:

L	level at which the conflict occurs. In reference to the network algorithm, L refers to the iteration.
n	set of nodes already selected or scheduled to start
$\bar{n}$	set of nodes not scheduled to start
$c^L j\ m_\ell$	completion time of node $(j\ m_\ell)$ at level L.
s^L	set of nodes that are under conflict at a particular level.
$B^L(j\ m_\ell)$	lower bound for node $(j\ m_\ell)$ at iteration L.
B^L	minimum lower bound on the schedule time at iteration L.

A job-based bound. The job-based bound procedure is a technique which is used to compute the total processing time on each job in the conflict set. The lower bound, $B^L(j\ m_\ell)$ for node $(j\ m_\ell)$ at level or iteration L, can be formulated as follows:

$$B^L(j\ m_\ell) = \max\left[\left[c_j^L\ m_\ell + \sum_{s=\ell+1}^{M} t_j\ m_s\right],\ \max_{\substack{i \\ i\,\varepsilon s^L \\ i \neq j}}\left[c_j^L\ m_\ell + \sum_{s=r}^{M} t_{im_s}\right]\right]$$

The first of the two expressions in the formulation consists of two terms:

$c^L(j\ m_\ell)$ the completion time of node $(j\ m_\ell)$.

$\sum_{s=\ell+1}^{M} t_{j\ m_\ell}$ the sum of the processing times of job j on the remaining machines. That is, the minimum time for the unscheduled nodes.

The second expression the formulation has two components:

$c^L(j\ m_\ell)$ the completion time of node $(j\ m_\ell)$.

$\sum_{s=r}^{M} t_{j\ m_s}$ the sum of the processing times of the other unscheduled nodes in the conflict set.

<u>A machine-based bound</u>. The second bounding procedure used in the composite bound is a machine-based bound, since a lower bound is computed with respect to the total processing time on each machine. The lower bound, $B^L(j\ m_\ell)$ for node $(j\ m_\ell)$ can be formulated as follows:

$$B^L(j\ m_\ell) = \max\left[\left[c^L{}_{j\ m_\ell} + \sum_{\substack{i\varepsilon\bar{n} \\ m = m_\ell}} t_{i\ m}\right],\ \max_{\substack{m \\ m \neq m_\ell}}\left[\min_i\left[c^L_{i\ m} - t_{i\ m}\right] + \sum_{\substack{i=1 \\ i\varepsilon\bar{n}}}^{J} t_{i\ m}\right]\right]$$

The first expression contains two terms:

$c^L(j\ m_\ell)$ the completion time of node $(j\ m_\ell)$.

$\sum_{\substack{i \varepsilon \bar{n} \\ m=m_\ell}} t_{i\ m}$ the sum of the processing times of the unscheduled nodes which include machine m.

The second expression also consists of two components:

$\min_{i \varepsilon \bar{n}} [c^L_{im} - t_{im}]$ the earliest time at which an unscheduled node can be started on machine m.

$\sum_{i=1}^{J} t_{i\ m}$ the sum of the processing times of the unscheduled nodes which involve machine m.

Now that the two lower bounds have been formulated, the composite bound can be presented formally. If the job-based bound is referred to as lower bound I (LB I) and the machine-based bound as lower bound II (LB II), the composite bound can be presented as follows:

$$\text{LB III} = \max\ [\ \text{LB I},\ \ \text{LB II}\ ]\ ,$$

where the composite bound will be referred to as LB III. Obviously, the conflicts are resolved in favor of the node which has the least composite lower bound.

C.2. Sample Problem

By considering the sample problem that has been used consistently in this thesis, the concept of the composite bound will be demonstrated. The conflicts at the first iteration have been resolved by hand and are presented in this section, while the remainder of the resolutions are summarized in Table C.1.

Conflict level one. When the initial precedence matrix, Q^0 was examined for entry candidates in each machine block, it was found that there was a conflict in block one. The two nodes competing for entry were (21) and (41). Consequently, these two nodes constitute the conflict set at iteration one.

Consider first, LB I. The first term in the formulation can be evaluated for node (21) as follows:

$$C^1_{(21)} = 8,$$

and

$$\sum_{s=\ell+1}^{M} t_{jm_s} = t_{23} + t_{22}$$
$$= 9.$$

Consequently, the first term in the bound has a value of 17.

The second term in the bound can be evaluated such that

$$C^1_{(21)} = 8,$$

and

$$\sum_{s=\ell}^{M} t_{im_s} = t_{41} + t_{42} + t_{43}$$
$$= 7 + 6 + 2$$
$$= 15.$$

Therefore, the evaluation of the second expression in the formulation can be given such that

$$\max\ [\ 8 + 15\ ] = \max\ [\ 23\ ]$$
$$= 23.$$

Finally, the lower bound can be computed:

$$B^1(21) = \max [\ 17,\ 23\ ],$$
$$= 23$$

Consider now the evaluation of LB I for node (41). The computation can be presented in the same manner as that used above. The first term in the formulation can be evaluated such that

$$C^1_{(41)} = 7,$$

and

$$\sum_{s=\ell+1}^{M} t_{jm_s} = t_{42} + t_{43}\ ,$$
$$= 8,$$

where, of course, the value of the term becomes 15.

The second term can be computed as follows:

$$C^1_{(41)} = 7,$$

and

$$\sum_{s=\pi}^{M} t_{im_s} = t_{21} + t_{23} + t_{22},$$
$$= 8 + 4 + 5,$$
$$= 17,$$

and finally,

$$\max [\ 7 + 17\ ] = \max [\ 24\ ],$$
$$= 24.$$

Consequently, the value of LB I for node 41 becomes

$$\max [\ 15,\ 24\ ] = 24.$$

Consider the second lower bound in the composite bound, namely LB II. Further, let us consider, first the evaluation of this bound for node (21).

The first term in the formulation of the bound can be computed such that

$$C^{1}_{(21)} = 8,$$

and

$$\begin{aligned}\sum_{\substack{i \varepsilon \bar{n} \\ m=m_{\ell}}} t_{im} &= t_{11} + t_{31} + t_{41}, \\ &= 3 + 3 + 7, \\ &= 13.\end{aligned}$$

The first term in the bound becomes 21.

The evaluation of the second term is made with respect to machines 2 and 3. Considering m = 2, first we can compute

$$\begin{aligned}\min_{i} [\, C^{L}_{im} - t_{im} \,] &= \min [\, C^{1}_{22} - t_{22},\, C_{32} - t_{32},\, C_{42} - t_{42}], \\ &= \min [\, 17 - 5,\, 20 - 9,\, 21 - 6 \,], \\ &= 11.\end{aligned}$$

Note that node (21) is not considered in the above computation because it has already been scheduled.

Continuing with the evaluation of the second term of the bound,

$$\begin{aligned}\sum_{\substack{i = 1 \\ i \,\varepsilon\, \bar{n}}}^{J} t_{im} &= t_{22} + t_{32} + t_{42} \\ &= 5 + 9 + 6, \\ &= 20.\end{aligned}$$

Now consider the case where m = 3.

$$\min_i \left[C^L_{im} - t_{im} \right] = \min \left[C^1_{13} - t_{13},\ C^1_{23} - t_{23},\ C^1_{43} - t_{43} \right]$$

$$= \min \left[8 - 2,\ 12 - 4,\ 23 - 2 \right]$$

$$= 6$$

and

$$\sum_{\substack{i=1 \\ i\varepsilon\bar{n}}}^{J} t_{im} = t_{13} + t_{23} + t_{43}$$

$$= 2 + 4 + 2,$$

$$= 8.$$

Therefore, the second term in LB II becomes

$$\max \left[11+20,\ 6+8 \right] = \max \left[31,\ 14 \right],$$

$$= 31.$$

Finally, the evaluation of the lower bound for node (21), $B^1(21)$ can be made such that

$$\max \left[21,\ 31 \right] = 31.$$

Following the same procedure, the lower bound for node (41) is computed. Consider the following evaluation of the first term of the formulation.

$$C^1_{(41)} = 7$$

and

$$\sum_{\substack{i\varepsilon\bar{n} \\ m=m_\ell}} t_{im} = t_{11} + t_{21} + t_{31},$$

$$= 3 + 8 + 3,$$

$$= 14.$$

The value of the first term is, obviously, 21.

In evaluating the second term, we shall consider the case when m = 2, first, such that

$$\min_i [\, C^L_{im} - t_{im} \,] = \min_i [\, C^1_{22} - t_{22},\ C^1_{32} - t_{32},\ C^1_{42} - t_{42} \,]$$

$$= \min [\, 24 - 5,\ 19 - 9,\ 13 - 6 \,]$$

$$= 7,$$

and

$$\sum_{\substack{i=1 \\ i \in \bar{n}}}^{J} t_{im} = t_{22} + t_{32} + t_{42},$$

$$= 5 + 9 + 6,$$

$$= 20.$$

The final value of the second term for the case, m = 2, becomes 27.

When m = 3, we can compute,

$$\min_i [\, C^L_{im} - t_{im}] = \min [C_{13} - t_{13},\ C_{23} - t_{23},\ C_{43} - t_{43} \,],$$

$$= \min [\, 8 - 2,\ 19 - 4,\ 15 - 2 \,]$$

$$= 6,$$

and

$$\sum_{\substack{i=1 \\ i \in \bar{n}}}^{J} t_{im} = t_{13} + t_{23} + t_{43},$$

$$= 2 + 4 + 2,$$

$$= 8.$$

For m = 3, the second term of the formulation becomes 14. Consequently, the final value of the second term can be computed such that

$$\max [\, 27,\ 14 \,] = 27.$$

Finally, the value of the bound for node (41), $B^1(41)$ can be given as follows:

max [21, 27] = 27.

At this point, we have computed the values of the lower bounds for nodes (21) and (41) using LB I and LB II. The next step, obviously, is to resolve the conflict using the values for the lower bounds obtained. Summarizing, the following can be given:

LB I : $B^1(21) = 23,$

$B^1(41) = 24,$

LB II : $B^1(21) = 31,$

$B^1(41) = 27.$

Application of the composite bound, LB III, yields the following formulation:

LB III = max [LB I, LB II] ,

where for node (21),

LB III = max [23, 31] ,

= 31,

and for node (41),

LB III = max [24, 27],

= 27.

In the above case, node (41) is selected to next start because it exhibits a lower bound that is less than that of node (21). Nonethe less, the remainder of the conflicts in the sample problem and their resolutions are summarized in Table C.1.

It should be noted that the minimum values for the composite bound, 27, are the same as the actual value of the schedule time. This is verified in the sample problem solution in Chapter II.

Table C.1. Summary of Computation of Lower Bounds for Sample Problem.

Conflict Level	Node	Lower Bounds LB I	LB II	LB III	Minimum LB III	Resolution
1	(21)	23	31	31		
	(41)	24	27	27	27	(41)
2	(21)	27	32	32		
	(31)	27	27	27	27	(31)
3	(21)	27	27	27	27	(21)
	(11)	30	30	30		

APPENDIX D

Computer Program

THE FOLLOWING DOCUMENT HAS PRINTING THAT EXTENDS INTO THE BINDING.

THIS IS AS RECEIVED FROM CUSTOMER.

```
C
C     THIS PROGRAM HAS BEEN CONSTRUCTED TO SOLVE A CLASS
C     OF COMBINATORIAL PROBLEMS USING THE NETWORK
C     ALGORITHM.  THE ALGORITHM IS BASED UPON THE
C     SCHEDULE ALGEBRA OPERATORS AS FORMULATED BY
C     B. GIFFLER.  INCLUDED IN THIS ROUTINE BESIDES THE
C     MAIN PROGRAM, ARE TWO SUBROUTINES, COMP AND ICPLT.
C
C
C
C
C                             MAIN VARIABLES
C
C
C     IPROB...........................NUMBER OF PROBLEMS
C                                     TO BE RUN.
C
C     M...............................NUMBER OF MACHINES.
C
C     JOB.............................NUMBER OF JOBS.
C
C     JM..............................NUMBER OF NODES IN THE
C                                     NETWORK, WHERE JM=M * JOB.
C
C     IO..............................MACHINE ORDERING MATRIX.
C
C     NP..............................PROCESSING TIME MATRIX.
C
C     S...............................PRECEDENCE MATRIX.
C
C     T...............................STARTING VECTOR.
C
C     ICMP............................COMPLETION TIME MATRIX.
C
C     ITER............................NUMBER OF ITERATIONS.
C
C     ICONU...........................NUMBER OF CONFLICTS.
C
C     SPROD...........................ELEMENTAL VALUES IN THE
C                                     UPDATED STARTING VECTOR.
C                                     CORRESPONDS TO TK'(I,1)
C                                     IN THE ALGORITHM.
C
C     TITER...........................RESULTANT UPDATE TO THE
C                                     STARTING VECTOR WHEN
C                                     MULTIPLIED BY THE PRE-
C                                     CEDENCE MATRIX.
C                                     CORRESPONDS TO TK' IN THE
C                                     ALGORITHM.
C
C     SEQ.............................SEQUENCING MATRIX.
C
C
C     VARIABLES PERTAINING TO INPUT AND OUTPUT CONTROL
C
C
C
C     JCHNG = 1.......................READ INPUT FROM IO AND NP
```

```
      C                                          AND GENERATE THE PRECEDENCE
      C                                          MATRIX,STARTING TIME VECTOR,
      C                                          PROCESSING TIME VECTOR,AND
      C                                          IDLE TIME VECTOR.
      C
      C             = 0.........................READ INPUT IN FORM OTHER
      C                                          THAN IO.
      C
      C
      C
      C      IPRER = 1..........................PRINT ONLY THE PROBLEM
      C                                          NUMBER,THE SEQUENCING
      C                                          MATRIX, AND THE SCHEDULE TIME.
      C
      C             = 0.........................PRINT ALL INFORMATION AT
      C                                          EVERY ITERATION, INCLUDING
      C                                          VALUES OF THE BOUNDING
      C                                          PROCEDURE.
      C
      C
      C
            DIMENSION S(40,40),T(40,1),WORK(40),PROC(40),IDLE(20),DELET(40),ENRGP00
           1TER(40),TITER(40,1),SPROD(40,1),SCHD(40),IO(15,10),NP(15,10),R(15,RGP00
           210),SEQ(9,15),LONE(15),IDON(15),LBSE(15),LBNE(15),ICOMX(15),ICMP(1RGP00
           35,5),ICTM(15,5)                                                    RGP00
            IPRER=1                                                            RGP00
            IPROB=25                                                           RGP00
            JPROB=0                                                            RGP00
        682 CONTINUE                                                           RGP00
            ICONU=0                                                            RGP00
            JPROB=JPROB+1                                                      RGP00
      C
      C     READ IN THE MACHINE ORDERING AND TIME PROCESSING MATRICES.
      C
       1332 FORMAT(3I4)                                                        RGP00
            READ(1,1332)M,JM,JOB                                               RGP00
            DO 2139 IQX=1,M                                                    RGP00
            DO 2140 IQXX=1,JOB                                                 RGP00
            SEQ(IQX,IQXX)=0                                                    RGP00
       2140 CONTINUE                                                           RGP00
       2139 CONTINUE                                                           RGP00
            JMM=JM-1                                                           RGP00
      C
      C
      C     THIS ROUTINE IS USED TO READ THE INPUT DATA
      C     AND GENERATE THE NEXT PRECEEDS MATRIX, THE                         RGP00
      C     START TIME VECTOR, THE WORK AND PROCESS TIME
      C     VECTORS.
      C
      C
            MM=M-1                                                             RGP00
            JCHNG=1                                                            RGP00
            IF(JCHNG.NE.1) GO TO 1516                                          RGP00
            CONTINUE                                                           RGP00
            DO 1784 I=1,JOB                                                    RGP00
            READ(1,1156)(IO(I,J),J=1,M)                                        RGP00
       1156 FORMAT(14I5)                                                       RGP00
       1784 CONTINUE                                                           RGP00
```

```
      DO 1793 I=1,JOB
      DO 1596 J=1,M
      JQ=IO(I,J)/100
      MQ=IO(I,J)-JQ*100
      IBLCK=MQ-1
      NEWR=IBLCK*JOB
      R(I,J)=NEWR+JQ
 1596 CONTINUE
 1793 CONTINUE
      DO 1619 I=1,JM
      T(I,1)=0
 1619 CONTINUE
      DO 1582 I=1,JOB
      IL=R(I,1)
      T(IL,1)=999
 1582 CONTINUE
      GO TO 1555
 1516 CONTINUE
      DO 1112 I=1,JOB
 1111 FORMAT(24F3.0)
      READ(1,1111)(R(I,J),J=1,M)
 1112 CONTINUE
 1555 CONTINUE
      DO 1113 I=1,JOB
 1114 FORMAT(14I5)
      READ(1,1114)(NP(I,J),J=1,M)
 1113 CONTINUE
C
C     INITIALIZE ICMP.
C
      DO 8668 ILI=1,JOB
      DO 8667 JLI=1,M
      ICMP(ILI,JLI)=NP(ILI,JLI)
      ICTM(ILI,JLI)=0
 8667 CONTINUE
 8668 CONTINUE
      DO 8669 ILI=1,JOB
      IIDAX=0
      DO 8666 JLI=1,M
      IIDA=NP(ILI,JLI)
      ICMP(ILI,JLI)=IIDAX+IIDA
      IIDAX=ICMP(ILI,JLI)
 8666 CONTINUE
 8669 CONTINUE
      IF(IPRER.EQ.1) GO TO 5092
      DO 5036 IIBO=1,JOB
      WRITE(3,5034)(ICMP(IIBO,IIJO),IIJO=1,M)
 5034 FORMAT(3I5)
 5036 CONTINUE
 5092 CONTINUE
      DO 1120 I=1,JM
      DO 1121 J=1,JM
      S(I,J)=0
 1121 CONTINUE
 1120 CONTINUE
      DO 1115 I=1,JOB
      DO 1161 J=1,MM
      IROW=R(I,J)
```

```
      ICOL=R(I,J+1)                                              RGP00
      S(IROW,ICOL)=NP(I,J)                                       RGP00
 1161 CONTINUE                                                   RGP00
 1115 CONTINUE                                                   RGP00
      JX=1                                                       RGP00
      JOBX=JOB                                                   RGP01
      DO 1250 I=1,JM                                             RGP01
      HOLD=0                                                     RGP01
      DO 1219 J=1,JM                                             RGP01
      IF(S(I,J).GT.0) GO TO 1212                                 RGP01
      GO TO 1219                                                 RGP01
 1212 HOLD=S(I,J)                                                RGP01
 1219 CONTINUE                                                   RGP01
      IF(HOLD.EQ.0) GO TO 1216                                   RGP01
      GO TO 1209                                                 RGP01
 1216 DO 6222 IP=1,JOB                                           RGP01
      DO 1181 JP=1,M                                             RGP01
      IF(R(IP,JP).EQ.I) GO TO 1127                               RGP01
      GO TO 1181                                                 RGP01
 1127 HOLD=NP(IP,JP)                                             RGP01
      GO TO 6222                                                 RGP01
 1181 CONTINUE                                                   RGP01
 6222 CONTINUE                                                   RGP01
 1209 IF(I.GT.JOBX) GO TO 6223                                   RGP01
      GO TO 1117                                                 RGP01
 6223 JX=JX+JOB                                                  RGP01
      JOBX=JOBX+JOB                                              RGP01
 1117 DO 1215 JP=JX,JOBX                                         RGP01
      IF(I.EQ.JP) GO TO 1215                                     RGP01
      S(I,JP)=HOLD*1000                                          RGP01
 1215 CONTINUE                                                   RGP01
 1250 CONTINUE                                                   RGP01
      IF(JCHNG.EQ.1) GO TO 1892                                  RGP01
      READ(1,1118)(T(I,1),I=1,JM)                                RGP01
 1118 FORMAT(18F4.0)                                             RGP01
      GO TO 1002                                                 RGP01
 7008 CONTINUE                                                   RGP01
 1892 CONTINUE                                                   RGP01
      IBX=1                                                      RGP01
      IB=JM/M                                                    RGP01
 1863 IWORK=0                                                    RGP01
      ICOUN=0                                                    RGP01
      DO 1861 I=1,JOB                                            RGP01
      DO 1868 J=1,M                                              RGP01
      IF(R(I,J).GE.IBX.AND.R(I,J).LE.IB) GO TO 1152              RGP01
      GO TO 6165                                                 RGP01
 1152 IWORK=NP(I,J)+IWORK                                        RGP01
      ICOUN=ICOUN+1                                              RGP01
      IF(ICOUN.EQ.JOB) GO TO 1157                                RGP01
 6165 GO TO 1868                                                 RGP01
 1157 ID=IB/JOB                                                  RGP01
      PROC(ID)=IWORK                                             RGP01
      IBX=IBX+JOB                                                RGP01
      IB=IB+JOB                                                  RGP01
      IF(IB.GT.JM) GO TO 1862                                    RGP01
      GO TO 1863                                                 RGP01
 1868 CONTINUE                                                   RGP01
 1861 CONTINUE                                                   RGP01
```

```
      1862 CONTINUE                                                    RGP01
           DO 1169 I=1,JOB                                             RGP01
           DO 1172 J=1,M                                               RGP01
           ITEL=R(I,J)                                                 RGP01
           WORK(ITEL)=NP(I,J)                                          RGP01
      1172 CONTINUE                                                    RGP01
      1169 CONTINUE                                                    RGP01
           GO TO 1001                                                  RGP01
      1002 CONTINUE                                                    RGP01
           READ(1,1802)(WORK(I),I=1,JM)                                RGP01
      1802 FORMAT(18F4.0,8X)                                           RGP01
           READ(1,1803)(PROC(K),K=1,M)                                 RGP01
      1803 FORMAT(5F4.0)                                               RGP01
      1001 CONTINUE                                                    RGP01
      1814 FORMAT(10X,'THIS IS PROBLEM',I3)                            RGP01
           WRITE(3,1814) JPROB                                         RGP01
           IF(IPRER.EQ.1) GO TO 5172                                   RGP01
         8 FORMAT(10X,'THE NEXT PRECEEDS MATRIX IS')                   RGP01
           WRITE(3,8)                                                  RGP01
           DO 9 I=1,JM                                                 RGP01
           WRITE(3,10)(S(I,J),J=1,JM)                                  RGP01
        10 FORMAT(12F6.0)                                              RGP01
         9 CONTINUE                                                    RGP01
        11 FORMAT(10X,'THE START VECTOR IS')                           RGP01
           WRITE(3,11)                                                 RGP01
           DO 12 I=1,JM                                                RGP01
           WRITE(3,13)T(I,1)                                           RGP01
        13 FORMAT(F6.0)                                                RGP01
        12 CONTINUE                                                    RGP01
      5172 CONTINUE                                                    RGP01
     C
     C     THE PROBLEM IS PROPERLY FORMULATED AT THIS POINT AND IS
     C     READY FOR SOLUTION.
     C
     C
           DO 15 I=1,JM                                                RGP01
           DELET(I)=0                                                  RGP01
        15 CONTINUE                                                    RGP01
           DO 16 J=1,JM                                                RGP01
           ENTER(J)=0                                                  RGP01
        16 CONTINUE                                                    RGP01
           ITER=0                                                      RGP01
           GO TO 476                                                   RGP01
       416 CONTINUE                                                    RGP01
           IF(IPRER.EQ.1) GO TO 5060                                   RGP01
        21 FORMAT(10X,'AFTER ITERATION',I3,'THE S/MATRIX IS')          RGP01
           WRITE(3,21) ITER                                            RGP01
           DO 22 I=1,JM                                                RGP01
           WRITE(3,23)(S(I,J),J=1,JM)                                  RGP01
        23 FORMAT(12F6.0)                                              RGP01
        22 CONTINUE                                                    RGP01
      5060 CONTINUE                                                    RGP01
       476 ITER=ITER+1                                                 RGP02
           DO 9921 IQRG=1,M                                            RGP02
           IDON(IQRG)=0                                                RGP02
      9921 CONTINUE                                                    RGP02
           L=1                                                         RGP02
           MM=JM/M                                                     RGP02
```

```
  926 CONTINUE                                                          RGP02
      IJACK=0                                                           RGP02
      NULL=0                                                            RGP02
      DO 825 J=L,MM                                                     RGP02
      IF(J.EQ.ENTER(J)) GO TO 825                                       RGP02
      DO 826 I=1,JM                                                     RGP02
      IF(S(I,J).GT.0.AND.S(I,J).LT.99) GO TO 859                        RGP02
      IF(I.LT.JM) GO TO 826                                             RGP02
   29 CONTINUE                                                          RGP02
      IF(IPRER.EQ.1) GO TO 5061                                         RGP02
      WRITE(3,47)ITER,J                                                 RGP02
   47 FORMAT(10X,'POTENTIALLY NULL AT ITERATION ',I3,'IS ',I3)          RGP02
 5061 CONTINUE                                                          RGP02
      NULL=NULL+1                                                       RGP02
      IJACK=J                                                           RGP02
      GO TO 825                                                         RGP02
  859 IF(I.EQ.DELET(I)) GO TO 30                                        RGP02
      GO TO 825                                                         RGP02
   30 IF(I.EQ.JM) GO TO 29                                              RGP02
  826 CONTINUE                                                          RGP02
  825 CONTINUE                                                          RGP02
      IF(NULL.EQ.1) GO TO 1751                                          RGP02
      GO TO 1752                                                        RGP02
 1751 ENTER(IJACK)=IJACK                                                RGP02
      IWWD=MM/JOB                                                       RGP02
      IDON(IWWD)=IWWD                                                   RGP02
C
C     ENTER IJACK INTO THE JOB SEQUENCING MATRIX, WHERE IJACK
C     REFERS TO COLUMNS WHICH CAN ENTER THE SOLUTION WITHOUT RESOLUTION
C     OF A CONFLICT.
C
      IPREY=IJACK                                                       RGP02
      CALL ICPLT(ICMP,IPREY,JOB,M,IO,ENTER,NP,ICTM)                     RGP02
      DO 5043 IBIDO=1,JOB                                               RGP02
      DO 5042 IIBDO=1,M                                                 RGP02
      ICMP(IBIDO,IIBDO)=ICTM(IBIDO,IIBDO)                               RGP02
 5042 CONTINUE                                                          RGP02
 5043 CONTINUE                                                          RGP02
      JNOW=ENTER(IJACK)                                                 RGP02
      IPICK=JNOW/JOB                                                    RGP02
      INUM=IPICK*JOB                                                    RGP02
      JPICK=JNOW-INUM                                                   RGP02
      IF(JPICK.EQ.0) GO TO 1829                                         RGP02
      MXX=IPICK+1                                                       RGP02
      GO TO 1830                                                        RGP02
 1829 JPICK=JOB                                                         RGP02
      MXX=IPICK                                                         RGP02
 1830 CONTINUE                                                          RGP02
      DO 4362 IKL=1,JOB                                                 RGP02
      IF(SEQ(MXX,IKL).NE.0) GO TO 4362                                  RGP02
      SEQ(MXX,IKL)=JPICK                                                RGP02
      GO TO 6527                                                        RGP02
 4362 CONTINUE                                                          RGP02
 6527 CONTINUE                                                          RGP02
 1752 IF(MM.EQ.JM) GO TO 1637                                           RGP02
      MM=MM+JM/M                                                        RGP02
      L=L+JM/M                                                          RGP02
      GO TO 926                                                         RGP02
```

```
 1637 L=1                                                              RGP02
      MM=JM/M                                                          RGP02
 9926 CONTINUE                                                         RGP02
      IEXTD=JOB+1                                                      RGP02
      DO 9214 ICRP=1,IEXTD                                             RGP02
      LONE(ICRP)=0                                                     RGP02
      LBNE(ICRP)=0                                                     RGP02
      LBSE(ICRP)=0                                                     RGP02
      ICOMX(ICRP)=0                                                    RGP02
 9214 CONTINUE                                                         RGP02
      NULL=0                                                           RGP02
      IGOO=MM/JOB                                                      RGP02
      IF(IDON(IGOO).NE.0) GO TO 9489                                   RGP02
      DO 9825 J=L,MM                                                   RGP02
      IF(J.EQ.ENTER(J)) GO TO 9825                                     RGP02
      DO 9826 I=1,JM                                                   RGP02
      IF(S(I,J).GT.0.AND.S(I,J).LT.99) GO TO 9859                      RGP02
      IF(I.LT.JM) GO TO 9826                                           RGP02
 9829 CONTINUE                                                         RGP02
      IF(IPRER.EQ.1) GO TO 5062                                        RGP02
      WRITE(3,9847) ITER,J                                             RGP02
 9847 FORMAT(10X,'POTENTIALLY NULL AT ITERATION ',I3,'IS ',I3)         RGP02
 5062 CONTINUE                                                         RGP02
      NULL=NULL+1                                                      RGP02
      IBBZ=NULL                                                        RGP02
      LONE(IBBZ)=J                                                     RGP02
      GO TO 9825                                                       RGP02
 9859 IF(I.EQ.DELET(I)) GO TO 9830                                     RGP02
      GO TO 9825                                                       RGP02
 9830 IF(I.EQ.JM) GO TO 9829                                           RGP02
 9826 CONTINUE                                                         RGP02
 9825 CONTINUE                                                         RGP02
      IF(LONE(1).EQ.0) GO TO 9874                                      RGP02
      GO TO 9875                                                       RGP02
 9874 GO TO 9489                                                       RGP02
 6692 JENT=0                                                           RGP02
C
C     RESOLVE CONFLICTS IN THE MACHINE BLOCKS WITH LOWER BOUND I (COMP).
C
 9875 CALL COMP(LONE,JOB,M,NP,IO,ENTER,LBSE,LBNE,ICOMX,JENT,ICMP,ICTM,IPRGP02
     1RER)                                                             RGP02
      ICONU=ICONU+1                                                    RGP03
      IF(IPRER.EQ.1) GO TO 5063                                        RGP03
      WRITE(3,9807)(ICOMX(ICB),ICB=1,JOB)                              RGP03
 9807 FORMAT(10X,'THE LOWER BOUND IS ',I4)                             RGP03
      WRITE(3,9808) JENT                                               RGP03
 9808 FORMAT(10X,'SELECT NODE ',I4)                                    RGP03
 5063 CONTINUE                                                         RGP03
      ENTER(JENT)=JENT                                                 RGP03
C                                                                      RGP03
C     UPDATE ICMP, BASED UPON THE SELECTED NODE, JENT.                 RGP03
C                                                                      RGP03
      NMMN=JENT/JOB                                                    RGP03
      NMMC=NMMN*JOB                                                    RGP03
      IF(NMMC.EQ.JENT) GO TO 2500                                      RGP03
      NMMX=NMMN+1                                                      RGP03
      JMMX=NMMN*JOB                                                    RGP03
      IF(JMMX.GT.JENT) GO TO 2531                                      RGP03
```

```
      GO TO 2532
 2531 JOBTX=JENT
      GO TO 2533
 2532 JOBTX=JENT-JMMX
 2533 NSEL=JOBTX*100
      NSEX=NSEL+NMMX
      GO TO 2501
 2500 NSEL=JOB*100
      NSEX=NSEL+NMMN
 2501 DO 6600 IUUX=1,JOB
      DO 6601 IUUU=1,M
      IF(IO(IUUX,IUUU).EQ.NSEX) GO TO 6605
      GO TO 6601
 6605 IIDA=ICMP(IUUX,IUUU)
 6601 CONTINUE
 6600 CONTINUE
      IKSU=NSEX/100
      IISU=IKSU*100
      IIKU=NSEX-IISU
      DO 6640 IKUU=1,JOB
      DO 6641 IIIK=1,M
      IF(IKUU.EQ.IKSU) GO TO 6640
 6642 IIMU=IO(IKUU,IIIK)/100
      IICU=IO(IKUU,IIIK)-IIMU*100
      IF(IICU.EQ.IIKU) GO TO 6643
      GO TO 6641
 6643 IIOU=IICU-1
      IOSU=IIOU*JOB
      IINU=IOSU+IIMU
      IF(ENTER(IINU).EQ.0) GO TO 6644
      GO TO 6640
 6644 IUPTE=0
      DO 6347 JCCCD=IIIK,M
      IF(JCCCD.EQ.IIIK) GO TO 6317
      GO TO 6327
 6317 KIJCK=NP(IKUU,JCCCD)+IIDA
      IF(KIJCK.LT.ICMP(IKUU,JCCCD)) GO TO 6315
      ICMP(IKUJ,JCCCD)=NP(IKUU,JCCCD)+IIDA
      GO TO 6316
 6315 IUPTE=ICMP(IKUU,JCCCD)
      GO TO 6347
 6316 IUPTE=ICMP(IKUU,JCCCD)
      GO TO 6347
 6327 ICMP(IKUU,JCCCD)=IUPTE+NP(IKUU,JCCCD)
      IUPTE=ICMP(IKUU,JCCCD)
 6347 CONTINUE
 6641 CONTINUE
 6640 CONTINUE
      IF(IPRER.EQ.1) GO TO 5064
      DO 5097 IBOQ=1,JOB
      WRITE(3,5098)(ICMP(IBOQ,JBOQ),JBOQ=1,M)
 5098 FORMAT(3I5)
 5097 CONTINUE
 5064 CONTINUE
C
C     ENTER NODE JENT INTO THE SEQUENCING MATRIX.
C
C
```

```
C
C     NOTE THAT THE ONLY DIFFERENCE BETWEEN NODES SIGNIFIED AS
C     IJACK AND THOSE AS JENT IS THAT THE FORMER ARE DETERMINED
C     WITHOUT RESOLUTION OF CONFLICTS WHILE THE LATTER ARE DE-
C     TERMINED BY SAID RESOLUTION.
C
      JNOW=ENTER(JENT)                                          RGP03
      IPICK=JNOW/JOB                                            RGP03
      INUM=IPICK*JOB                                            RGP03
      JPICK=JNOW-INUM                                           RGP03
      IF(JPICK.EQ.0) GO TO 9429                                 RGP03
      MXX=IPICK+1                                               RGP03
      GO TO 9430                                                RGP03
 9429 JPICK=JOB                                                 RGP03
      MXX=IPICK                                                 RGP03
 9430 CONTINUE                                                  RGP03
      DO 9462 IKL=1,JOB                                         RGP03
      IF(SEQ(MXX,IKL).NE.0) GO TO 9462                          RGP03
      SEQ(MXX,IKL)=JPICK                                        RGP03
      GO TO 9427                                                RGP03
 9462 CONTINUE                                                  RGP03
 9427 CONTINUE                                                  RGP03
 9489 IF(MM.EQ.JM) GO TO 800                                    RGP03
      MM=MM+JM/M                                                RGP03
      L=L+JM/M                                                  RGP03
      GO TO 9926                                                RGP03
C
C     ENTER THE SELECTED NODES IN THE PRECEDENCE MATRIX.
C
  800 DO 801 J=1,JM                                             RGP03
      NO=0                                                      RGP03
      IF(J.EQ.ENTER(J)) GO TO 802                               RGP03
      GO TO 801                                                 RGP03
  802 DO 803 IX=1,JM                                            RGP03
      IF(NO.GT.0) GO TO 803                                     RGP03
      IF(S(IX,J).GT.999) GO TO 804                              RGP04
      GO TO 803                                                 RGP04
  804 NO=NO+1                                                   RGP04
  803 CONTINUE                                                  RGP04
      IF(NO.EQ.0) GO TO 801                                     RGP04
      DO 810 I=1,JM                                             RGP04
      IF(S(I,J).GT.999) GO TO 811                               RGP04
      GO TO 810                                                 RGP04
  811 IF(I.EQ.DELET(I)) GO TO 821                               RGP04
      S(I,J)=0                                                  RGP04
      GO TO 810                                                 RGP04
  821 S(I,J)=S(I,J)/1000                                        RGP04
  820 DO 815 IP=1,JM                                            RGP04
      IF(S(I,IP).GT.999) GO TO 816                              RGP04
      GO TO 815                                                 RGP04
  816 S(I,IP)=0                                                 RGP04
  815 CONTINUE                                                  RGP04
  810 CONTINUE                                                  RGP04
C
C     NOW WE MUST DELETE THE CORRESPONDING ROW.                 RGP04
C
      I=J                                                       RGP04
      DELET(I)=I                                                RGP04
```

```
  801 CONTINUE                                                    RGP04
C
C     CHECK TO SEE IF ALL NODES HAVE BEEN ENTERED.
C
      PLUS=0                                                      RGP04
      DO 640 I=1,JM                                               RGP04
      DO 641 J=1,JM                                               RGP04
      IF(S(I,J).GT.99) GO TO 642                                  RGP04
      GO TO 641                                                   RGP04
  642 PLUS=PLUS+1                                                 RGP04
  641 CONTINUE                                                    RGP04
  640 CONTINUE                                                    RGP04
      IF(PLUS.GT.0) GO TO 416                                     RGP04
C
C     UPDATE THE STARTING VECTOR.
C
  977 CONTINUE                                                    RGP04
      TCOL=1                                                      RGP04
      DO 905 J=1,JM                                               RGP04
      DO 904 I=1,JM                                               RGP04
      IF(T(I,TCOL).EQ.999) GO TO 963                              RGP04
      IF(T(I,TCOL).EQ.0) GO TO 902                                RGP04
      IF(T(I,TCOL).GT.0.AND.T(I,TCOL).LT.999) GO TO 969           RGP04
  963 IF(S(I,J).GT.0.AND.S(I,J).LT.99) GO TO 901                  RGP04
      IF(S(I,J).EQ.0.OR.S(I,J).GT.999) GO TO 900                  RGP04
      IF(S(I,J).GT.99) GO TO 49                                   RGP04
  901 SPROD(I,TCOL)=S(I,J)                                        RGP04
      GO TO 904                                                   RGP04
   49 SPROD(I,TCOL)=S(I,J)/100                                    RGP04
      GO TO 904                                                   RGP04
  900 SPROD(I,TCOL)=0                                             RGP04
      GO TO 904                                                   RGP04
  902 SPROD(I,TCOL)=0                                             RGP04
      GO TO 904                                                   RGP04
  969 IF(S(I,J).EQ.0.OR.S(I,J).GT.999) GO TO 770                  RGP04
      IF(S(I,J).GT.99.AND.S(I,J).LT.999) GO TO 778                RGP04
      SPROD(I,TCOL)=T(I,TCOL)+S(I,J)                              RGP04
      GO TO 904                                                   RGP04
  778 SPROD(I,TCOL)=T(I,TCOL)+S(I,J)/100                          RGP04
      GO TO 904                                                   RGP04
  770 SPROD(I,TCOL)=0                                             RGP04
  904 CONTINUE                                                    RGP04
      MAL=M                                                       RGP04
      M=1                                                         RGP04
      MAXT=SPROD(M,TCOL)                                          RGP04
      DO 709 M=1,JMM                                              RGP04
      IF(MAXT.GE.SPROD(M+1,TCOL)) GO TO 709                       RGP04
      MAXT=SPROD(M+1,TCOL)                                        RGP04
  709 CONTINUE                                                    RGP04
      M=MAL                                                       RGP04
      I=J                                                         RGP04
      TITER(I,TCOL)=MAXT                                          RGP04
  905 CONTINUE                                                    RGP04
      AGAIN=0                                                     RGP04
      DO 797 I=1,JM                                               RGP04
      IF(T(I,TCOL).EQ.999) GO TO 796                              RGP04
      IF(T(I,TCOL).LT.TITER(I,TCOL)) GO TO 795                    RGP04
      GO TO 797                                                   RGP04
```

```
  795 T(I,TCOL)=TITER(I,TCOL)                                           RGP04
      AGAIN=AGAIN+1                                                     RGP04
      GO TO 797                                                         RGP04
  796 IF(TITER(I,TCOL).GT.0) GO TO 339                                  RGP04
      GO TO 797                                                         RGP04
  339 T(I,TCOL)=TITER(I,TCOL)                                           RGP04
      AGAIN=AGAIN+1                                                     RGP04
  797 CONTINUE                                                          RGP04
C
C     THE ABOVE SECTION OF THE PROGRAM WILL UPDATE BOTH
C     THE T-VECTOR AND THE S-MATRIX.
C                                                                       RGP04
      IF(AGAIN.NE.0) GO TO 977                                          RGP04
  782 CONTINUE                                                          RGP04
      IF(IPRER.EQ.1) GO TO 5173                                         RGP04
  781 WRITE(3,913)                                                      RGP04
  913 FORMAT(10X,'THE FINAL START TIME VECTOR IS')                      RGP04
      WRITE(3,912)(T(I,TCOL),I=1,JM)                                    RGP04
  912 FORMAT(F6.0)                                                      RGP04
      WRITE(3,827)                                                      RGP04
  827 FORMAT(10X,'THE FINAL S-MATRIX IS')                               RGP04
      DO 828 I=1,JM                                                     RGP04
      WRITE(3,829)(S(I,J),J=1,JM)                                       RGP04
  829 FORMAT(12F6.0)                                                    RGP04
  828 CONTINUE                                                          RGP04
 5173 CONTINUE                                                          RGP05
      N=1                                                               RGP05
      K=1                                                               RGP05
C
C     THE REMAINDER OF THE ROUTINE COMPUTES SCHEDULE TIME AND FACILITY
C     IDLE TIME.
C
      DO 929 N=1,JM                                                     RGP05
      SCHD(N)=WORK(N)+T(N,TCOL)                                         RGP05
      IF(SCHD(N).GT.999) SCHD(N)=SCHD(N)-999                            RGP05
  929 CONTINUE                                                          RGP05
      CHEK1=SCHD(1)                                                     RGP05
      DO 939 I=1,JMM                                                    RGP05
      IF(CHEK1.GE.SCHD(I+1)) GO TO 939                                  RGP05
      CHEK1=SCHD(I+1)                                                   RGP05
  939 CONTINUE                                                          RGP05
      WRITE(3,919) CHEK1                                                RGP05
  919 FORMAT(10X,'THE SCHEDULE TIME IS',F4.0)                           RGP05
C     NOW, USING CHEK1, WE CAN CALCULATE MACHINE IDLE TIME.             RGP05
      DO 808 I=1,M                                                      RGP05
      IDLE(I)=CHEK1-PROC(I)                                             RGP05
  808 CONTINUE                                                          RGP05
      WRITE(3,812)                                                      RGP05
      WRITE(3,809)(IDLE(I),I=1,M)                                       RGP05
  812 FORMAT(10X,'THE FACILITY IDLE TIMES ARE')                         RGP05
  809 FORMAT(I4)                                                        RGP05
      DO 9032 IPQ=1,M                                                   RGP05
      WRITE(3,9041)(SEQ(IPQ,IPQX),IPQX=1,JOB)                           RGP05
 9041 FORMAT(10F5.0)                                                    RGP05
 9032 CONTINUE                                                          RGP05
      WRITE(2,5067) ITER,ICONU,CHEK1                                    RGP05
 5067 FORMAT(2I5,F4.0)                                                  RGP05
      WRITE(3,5068) ITER,ICONU                                          RGP05
```

```
      5068 FORMAT(10X,'THE NO. OF ITER. AND CONFLICTS ARE',2I5)          RGP05
           IF(JPROB.NE.IPROB) GO TO 682                                  RGP05
           STOP                                                          RGP05
           END                                                           RGP05
```

```
      SUBROUTINE COMP(LONE,JOB,M,NP,IO,ENTER,LBSE,LBNE,ICOMX,JENT,ICMP,IRGP05
     1CTM,IPRER)                                                        RGP05
      DIMENSION LONE(15),NP(15,5),IO(15,5),ENTER(50),LBSE(15),LBNE(15),IRGP05
     1COMX(15),IUNSC(40),MIN(15),MINK(15),ISECT(15),ISEC(15),ICMP(15,5),RGP05
     2ICTM(15,5)                                                        RGP05
C                                                                       RGP05
C     THIS ROUTINE COMPUTES THE LOWER BOUND FOR EACH                    RGP05
C     NODE IN THE CONFLICT SET IN EACH MACHINE BLOCK                    RGP05
C     AT ALL ITERATIONS-USING LOWER BOUND ONE.                          RGP05
C     LOWER BOUND I IS A COMPOSITE BOUND CONSISTING OF TWO
C     INDIVIDUAL BOUNDS.
C                                                                       RGP05
C     COMPUTE FIRST, THE VALUE OF THE FIRST BOUND IN THE                RGP05
C     COMPOSITE LOWER BOUND.                                            RGP05
C                                                                       RGP05
      JM=JOB*M                                                          RGP05
      DO 8888 IPA=1,JOB                                                 RGP05
      ISEC(IPA)=0                                                       RGP05
      ISECT(IPA)=0                                                      RGP05
 8888 CONTINUE                                                          RGP05
      IQB=0                                                             RGP05
 5001 IQB=IQB+1                                                         RGP05
      IQD=IQB                                                           RGP05
      IF(LONE(IQD).EQ.0) GO TO 5002                                     RGP05
C                                                                       RGP05
C     COMPUTE THE COMPLETION TIME                                       RGP05
C                                                                       RGP05
      IPREY=LONE(IQD)                                                   RGP05
      IMCH=IPREY/JOB                                                    RGP05
      IMC=IMCH*JOB                                                      RGP05
      IF(IMC.EQ.IPREY) GO TO 7500                                       RGP05
      IMCX=IMCH+1                                                       RGP05
      JOBM=IMCH*JOB                                                     RGP05
      IF(JOBM.GT.IPREY) GO TO 6031                                      RGP05
      GO TO 6032                                                        RGP05
 6031 JOBP=IPREY                                                        RGP05
      GO TO 6033                                                        RGP05
 6032 JOBP=IPREY-JOBM                                                   RGP05
 6033 ISEL=JOBP*100                                                     RGP05
      ISEX=ISEL+IMCX                                                    RGP05
      GO TO 7501                                                        RGP05
 7500 ISEL=JOB*100                                                      RGP05
      ISEX=ISEL+IMCH                                                    RGP05
 7501 I=ISEX/100                                                        RGP05
      DO 5011 J=1,M                                                     RGP05
      IF(IO(I,J).EQ.ISEX) GO TO 5012                                    RGP05
      GO TO 5011                                                        RGP05
 5012 IHOL=J                                                            RGP05
 5011 CONTINUE                                                          RGP05
      ISUMX=ICMP(I,IHOL)                                                RGP05
C
C     COMPUTE THE SECOND COMPONENT IN THE FIRST TERM OF THE BOUND.
C
      JHOL=IHOL+1                                                       RGP05
      IJSUM=0                                                           RGP05
      IF(JHOL.GT.M) GO TO 6608                                          RGP05
      DO 5014 JZX=JHOL,M                                                RGP05
      JISUM=NP(I,JZX)                                                   RGP05
```

```
      IJSUM=IJSUM+JISUM                                                 RGP05
 5014 CONTINUE                                                          RGP05
 6608 CONTINUE                                                          RGP05
      IFIRS=ISUMX+IJSUM                                                 RGP05
      IF(IPRER.EQ.1) GO TO 4370                                         RGP05
      WRITE(3,3901) ISUMX                                               RGP05
 3901 FORMAT(10X,'COMPLETION TIME IS ',I4)                              RGP05
      WRITE(3,3900) IFIRS                                               RGP05
 3900 FORMAT(10X,'FIRST TERM IS ',I4)                                   RGP05
 4370 CONTINUE                                                          RGP05
C                                                                       RGP05
C     COMPUTE THE SECOND TERM                                           RGP05
C                                                                       RGP05
      DO 8846 IBT=1,JOB                                                 RGP05
      ISEC(IBT)=0                                                       RGP06
 8846 CONTINUE                                                          RGP06
      DO 5020 IXD=1,JOB                                                 RGP06
      IF(IXD.EQ.IQD) GO TO 5020                                         RGP06
      IF(LONE(IXD).EQ.0) GO TO 5020                                     RGP06
      IPREY=LONE(IXD)                                                   RGP06
      IMCH=IPREY/JOB                                                    RGP06
      IMC=IMCH*JOB                                                      RGP06
      IF(IMC.EQ.IPREY) GO TO 8500                                       RGP06
      IMCX=IMCH+1                                                       RGP06
      JOBM=IMCH*JOB                                                     RGP06
      IF(JOBM.GT.IPREY) GO TO 6041                                      RGP06
      GO TO 6042                                                        RGP06
 6041 JOBP=IPREY                                                        RGP06
      GO TO 6043                                                        RGP06
 6042 JOBP=IPREY-JOBM                                                   RGP06
 6043 ISEL=JOBP*100                                                     RGP06
      ISEX=ISEL+IMCX                                                    RGP06
      GO TO 8501                                                        RGP06
 8500 ISEL=JOB*100                                                      RGP06
      ISEX=ISEL+IMCH                                                    RGP06
 8501 I=ISEX/100                                                        RGP06
      DO 6090 J=1,M                                                     RGP06
      IF(IO(I,J).EQ.ISEX) GO TO 6091                                    RGP06
      GO TO 6090                                                        RGP06
 6091 JHOL=J                                                            RGP06
 6090 CONTINUE                                                          RGP06
      JISUM=0                                                           RGP06
      DO 6080 JZZ=JHOL,M                                                RGP06
      JXSUM=NP(I,JZZ)                                                   RGP06
      JISUM=JXSUM+JISUM                                                 RGP06
 6080 CONTINUE                                                          RGP06
      IF(IPRER.EQ.1) GO TO 4371                                         RGP06
      WRITE(3,3902) JISUM                                               RGP06
 3902 FORMAT(10X,'FOURTH TERM IS ',I4)                                  RGP06
 4371 CONTINUE                                                          RGP06
      ISEC(IXD)=ISUMX+JISUM                                             RGP06
 5020 CONTINUE                                                          RGP06
      JOZQ=JOB-1                                                        RGP06
      DO 6070 IXD=1,JOZQ                                                RGP06
      IF(IXD.GT.1) GO TO 6072                                           RGP06
      KEPE=ISEC(1)                                                      RGP06
 6072 IF(KEPE.GT.ISEC(IXD+1)) GO TO 6070                                RGP06
      KEPE=ISEC(IXD+1)
```

```
      6070 CONTINUE                                                   RGP06
           IFISC=KEPE                                                 RGP06
           IDIFF=IFIRS-IFISC                                          RGP06
           IF(IDIFF.GT.0) GO TO 9539                                  RGP06
           GO TO 9538                                                 RGP06
      9539 LBNE(IQD)=IFIRS                                            RGP06
           GO TO 9537                                                 RGP06
      9538 LBNE(IQD)=IFISC                                            RGP06
      9537 CONTINUE                                                   RGP06
           GO TO 5001                                                 RGP06
      5002 CONTINUE                                                   RGP06
           IF(IPRER.EQ.1) GO TO 4372                                  RGP06
           WRITE(3,8823)(LBNE(IQD),IQD=1,JOB)                         RGP06
      8823 FORMAT(10X,'THE VALUE OF THE LB I IS ',I5)                 RGP06
      4372 CONTINUE                                                   RGP06
     C
     C
     C                                                                RGP06
     C     COMPUTE THE VALUE OF LOWER BOUND USING LOWER               RGP06
     C     BOUND TWO.                                                 RGP06
     C                                                                RGP06
     C                                                                RGP06
     C
     C
     C     COMPUTE THE FIRST TERM IN THE FORMULATION                  RGP06
     C                                                                RGP06
           IQR=0                                                      RGP06
      8001 IQR=IQR+1                                                  RGP06
           IQW=IQR                                                    RGP06
           IF(LONE(IQW).EQ.0) GO TO 8002                              RGP06
     C                                                                RGP06
     C     COMPUTE THE COMPLETION TIME                                RGP06
     C                                                                RGP06
           IPREY=LONE(IQW)                                            RGP06
           IMCH=IPREY/JOB                                             RGP06
           IMC=IMCH*JOB                                               RGP06
           IF(IMC.EQ.IPREY) GO TO 9500                                RGP06
           IMCX=IMCH+1                                                RGP06
           JOBM=IMCH*JOB                                              RGP06
           IF(JOBM.GT.IPREY) GO TO 9031                               RGP06
           GO TO 9032                                                 RGP06
      9031 JOBP=IPREY                                                 RGP06
           GO TO 9033                                                 RGP06
      9032 JOBP=IPREY-JOBM                                            RGP06
      9033 ISEL=JOBP*100                                              RGP06
           ISEX=ISEL+IMCX                                             RGP06
           GO TO 9501                                                 RGP06
      9500 ISEL=JOB*100                                               RGP06
           ISEX=ISEL+IMCH                                             RGP06
      9501 I=ISEX/100                                                 RGP06
           DO 9511 J=1,M                                              RGP06
           IF(IO(I,J).EQ.ISEX) GO TO 9512                             RGP06
           GO TO 9511                                                 RGP06
      9512 IHOL=J                                                     RGP06
      9511 CONTINUE                                                   RGP06
           ISUMX=ICMP(I,IHOL)                                         RGP06
           IF(IPRER.EQ.1) GO TO 4373                                  RGP06
           WRITE(3,3903) ISUMX                                        RGP06
```

```
 3903 FORMAT(10X,'COMPLETION TIME IS ',I4)                              RGP06
 4373 CONTINUE                                                          RGP06
C                                                                       RGP07
C     COMPUTE PROCESSING TIME OF OTHER UNSCHEDULED JOBS                 RGP07
C     ON THE SAME MACHINE.                                              RGP07
C                                                                       RGP07
      IOTHR=IPREY/JOB                                                   RGP07
      IOT=IOTHR*JOB                                                     RGP07
      IF(IOT.EQ.IPREY) GO TO 9560                                       RGP07
      GO TO 9561                                                        RGP07
 9560 IMCKI=IPREY                                                       RGP07
      GO TO 9562                                                        RGP07
 9561 INMC=IOTHR*JOB                                                    RGP07
      JKZ=INMC+1                                                        RGP07
      JKX=INMC+JOB                                                      RGP07
      GO TO 6939                                                        RGP07
 9562 JOBK=JOB-1                                                        RGP07
      JKZ=IPREY-JOBK                                                    RGP07
      JKX=IPREY                                                         RGP07
C                                                                       RGP07
C     CHECK FOR UNSCHEDULED NODES.                                      RGP07
C                                                                       RGP07
 6939 CONTINUE                                                          RGP07
      DO 4908 IOO=1,JM                                                  RGP07
      IUNSC(IOO)=0                                                      RGP07
 4908 CONTINUE                                                          RGP07
 9563 DO 9567 IUJ=JKZ,JKX                                               RGP07
      IF(IUJ.EQ.LONE(IQW)) GO TO 9567                                   RGP07
      IF(ENTER(IUJ).EQ.0) GO TO 9568                                    RGP07
      GO TO 9567                                                        RGP07
 9568 IUNSC(IUJ)=IUJ                                                    RGP07
 9567 CONTINUE                                                          RGP07
      IUNSM=0                                                           RGP07
      DO 9570 IUJ=JKZ,JKX                                               RGP07
      IF(IUNSC(IUJ).EQ.0) GO TO 9570                                    RGP07
      IPREY=IUNSC(IUJ)                                                  RGP07
      IMCH=IPREY/JOB                                                    RGP07
      IMC=IMCH*JOB                                                      RGP07
      IF(IMC.EQ.IPREY) GO TO 9571                                       RGP07
      IMCX=IMCH+1                                                       RGP07
      JOBM=IMCH*JOB                                                     RGP07
      IF(JOBM.GT.IPREY) GO TO 9572                                      RGP07
      GO TO 9573                                                        RGP07
 9572 JOBP=IPREY                                                        RGP07
      GO TO 9574                                                        RGP07
 9573 JOBP=IPREY-JOBM                                                   RGP07
 9574 ISEL=JOBP*100                                                     RGP07
      ISEX=ISEL+IMCX                                                    RGP07
      GO TO 9575                                                        RGP07
 9571 ISEL=JOB*100                                                      RGP07
      ISEX=ISEL+IMCX                                                    RGP07
 9575 I=ISEX/100                                                        RGP07
      DO 6511 JA=1,M                                                    RGP07
      IF(IO(I,JA).EQ.ISEX) GO TO 6512                                   RGP07
      GO TO 6511                                                        RGP07
 6512 IHOL=JA                                                           RGP07
 6511 CONTINUE                                                          RGP07
      IPRSS=NP(I,IHOL)                                                  RGP07
```

```
      IUNSM=IUNSM+IPRSS                                                 RGP07
 9570 CONTINUE                                                          RGP07
      IUNXM=IUNSM+ISUMX                                                 RGP07
      IF(IPRER.EQ.1) GO TO 4374                                         RGP07
      WRITE(3,3904) IUNXM                                               RGP07
 3904 FORMAT(10X,'FIRST TERM IS ',I4)                                   RGP07
 4374 CONTINUE                                                          RGP07
C                                                                       RGP07
C     COMPUTE THE SECOND TERM OF THE LOWER BOUND.                       RGP07
C                                                                       RGP07
C                                                                       RGP07
C                                                                       RGP07
C     IDENTIFY THE OTHER MACHINES                                       RGP07
C                                                                       RGP07
      IPREY=LONE(IQR)                                                   RGP07
      CALL ICPLT(ICMP,IPREY,JOB,M,IO,ENTER,NP,ICTM)                     RGP07
      IMCH=IPREY/JOB                                                    RGP07
      IMC=IMCH*JOB                                                      RGP07
      IF(IMC.EQ.IPREY) GO TO 9430                                       RGP07
      IMCH=IMCH+1                                                       RGP07
      GO TO 9431                                                        RGP07
 9430 IMCH=IPREY/JOB                                                    RGP07
 9431 CONTINUE                                                          RGP07
      DO 4900 JZZZ=1,M                                                  RGP07
      IF(JZZZ.EQ.IMCH) GO TO 4900                                       RGP07
C                                                                       RGP07
C     COMPUTE THE FIRST COMPONENT OF THE SECOND TERM OF THE BOUND.
C                                                                       RGP07
      DO 7662 IKK=1,JOB                                                 RGP07
      MIN(IKK)=0                                                        RGP07
      MINK(IKK)=0                                                       RGP07
 7662 CONTINUE                                                          RGP07
      DO 4901 JZZX=1,JOB                                                RGP07
      JZZK=JZZX*100                                                     RGP07
      JZZP=JZZK+JZZZ                                                    RGP07
      JXOX=JZZP/100                                                     RGP07
      JZZXX=JZZP-JXOX*100                                               RGP07
      JZXXX=JZZXX-1                                                     RGP07
      JZXN=JZXXX*JOB                                                    RGP07
      JTRY=JZXN+JXOX                                                    RGP07
C                                                                       RGP07
C     CHECK IF THIS NODE HAS ALREADY BEEN ENTERED.                      RGP07
C                                                                       RGP07
      IF(ENTER(JTRY).EQ.0) GO TO 4902                                   RGP07
      GO TO 4901                                                        RGP08
 4902 MLK=JZZX                                                          RGP08
      DO 4903 JOW=1,M                                                   RGP08
      IF(IO(MLK,JOW).EQ.JZZP) GO TO 4904                                RGP08
      GO TO 4903                                                        RGP08
 4904 JQD=JOW                                                           RGP08
 4903 CONTINUE                                                          RGP08
      KOOL=ICTM(MLK,JQD)                                                RGP08
      NKOOL=NP(MLK,JQD)                                                 RGP08
      MIN(MLK)=KOOL-NKOOL                                               RGP08
      MINK(MLK)=NKOOL                                                   RGP08
 4901 CONTINUE                                                          RGP08
      IOKP=100                                                          RGP08
      DO 4910 MLK=1,JOB                                                 RGP08
```

```
      IF(MIN(MLK).EQ.0) GO TO 4910                                    RGP08
      IF(IOKP.LE.MIN(MLK)) GO TO 4910                                 RGP08
 4911 IOKP=MIN(MLK)                                                   RGP08
 4910 CONTINUE                                                        RGP08
C                                                                     RGP08
C     ADD THIS MINIMUM TO THE SECOND COMPONENT OF THE  SECOND TERM
C     OF THE BOUND, WHERE THE SECOND COMPONENT IS IDENTIFIED AS
C     MITOL.
C                                                                     RGP08
      MITOL=0                                                         RGP08
      DO 4920 MLK=1,JOB                                               RGP08
      IF(MINK(MLK).EQ.0) GO TO 4920                                   RGP08
      MITOL=MINK(MLK)+MITOL                                           RGP08
      IF(IPRER.EQ.1) GO TO 4377                                       RGP08
      WRITE(3,3907) MITOL                                             RGP08
 3907 FORMAT(10X,'MITOL EQUALS ',I4)                                  RGP08
 4377 CONTINUE                                                        RGP08
 4920 CONTINUE                                                        RGP08
      ISECT(JZZZ)=IOKP+MITOL                                          RGP08
 4900 CONTINUE                                                        RGP08
      ILAKS=0                                                         RGP08
      DO 5650 JZZZ=1,M                                                RGP08
      IF(JZZZ.EQ.IMCH) GO TO 5650                                     RGP08
      IF(ILAKS.GT.ISECT(JZZZ)) GO TO 5650                             RGP08
      ILAKS=ISECT(JZZZ)                                               RGP08
      IF(IPRER.EQ.1) GO TO 4375                                       RGP08
      WRITE(3,3908) ILAKS                                             RGP08
 3908 FORMAT(10X,'ILAKS EQUALS ',I4)                                  RGP08
 4375 CONTINUE                                                        RGP08
 5650 CONTINUE                                                        RGP08
C                                                                     RGP08
C     COMPUTE THE VALUE OF THE SECOND LOWER BOUND                     RGP08
C     WHERE THE FIRST TERM IS IUNXM AND THE SECOND
C     TERM IS ILAKS.                                                  RGP08
C                                                                     RGP08
C                                                                     RGP08
C                                                                     RGP08
      IWICH=IUNXM-ILAKS                                               RGP08
      IF(IWICH.GE.0) GO TO 7735                                       RGP08
      LBSE(IQW)=ILAKS                                                 RGP08
      GO TO 7736                                                      RGP08
 7735 LBSE(IQW)=IUNXM                                                 RGP08
 7736 GO TO 8001                                                      RGP08
 8002 CONTINUE                                                        RGP08
      IF(IPRER.EQ.1) GO TO 4376                                       RGP08
      WRITE(3,8822)(LBSE(IQW),IQW=1,JOB)                              RGP08
 8822 FORMAT(10X,'THE VALUE OF THE LB II IS ',I5)                     RGP08
 4376 CONTINUE                                                        RGP08
C                                                                     RGP08
C     COMPUTE THE COMPOSITE LOWER BOUND FOR THE                       RGP08
C     NODE LONE(X), WHERE X = 1, 2,..., AND REFERS TO THE 1ST, 2ND,
C     ETC. NODE IN THE CONFLICT SET.
C                                                                     RGP08
      DO 9635 ICB=1,JOB                                               RGP08
      IF(LONE(ICB).EQ.0) GO TO 9635                                   RGP08
      ICOMP=LBNE(ICB)-LBSE(ICB)                                       RGP08
      IF(ICOMP.GE.0) GO TO 9636                                       RGP08
      ICOMX(ICB)=LBSE(ICB)                                            RGP08
```

```
          GO TO 9635                                              RGP08
    9636  ICOMX(ICB)=LBNE(ICB)                                    RGP08
    9635  CONTINUE                                                RGP08
C                                                                 RGP08
C         SELECT WHICH NODE WILL ENTER (RESOLVE THE CONFLICT).
C                                                                 RGP08
          IHOH=1000                                               RGP08
          DO 8945 ICB=1,JOB                                       RGP08
          IF(ICOMX(ICB).EQ.0) GO TO 8945                          RGP08
          IF(IHOH.LE.ICOMX(ICB)) GO TO 8945                       RGP08
          IHOH=ICOMX(ICB)                                         RGP08
          JENTX=ICB                                               RGP08
    8945  CONTINUE                                                RGP08
          JENT=LONE(JENTX)                                        RGP08
          RETURN                                                  RGP08
          END                                                     RGP08
```

```
      SUBROUTINE ICPLT(ICMP,IPREY,JOB,M,IO,ENTER,NP,ICTM)                RGP08
C                                                                        RGP08
C     THIS ROUTINE COMPUTES THE COMPLETION TIME MATRIX.
C                                                                        RGP08
      DIMENSION ICTM(15,5),ICMP(15,5),IO(15,5),ENTER(40),NP(15,5)        RGP08
 6551 DO 5800 IWI=1,JOB                                                  RGP08
      DO 5801 IWJ=1,M                                                    RGP08
      ICTM(IWI,IWJ)=ICMP(IWI,IWJ)                                        RGP08
 5801 CONTINUE                                                           RGP08
 5800 CONTINUE                                                           RGP08
C                                                                        RGP08
C     UPDATE ICTM W.R.T. THE NODE UNDER RESOLUTION, WHERE ICTM IS
C     THE TRANSITORY FORM OF THE COMPLETION TIME MATRIX
C     UNTIL A CONFLICT HAS BEEN RESOLVED AFTER WHICH TIME THE
C     MATRIX IS IDENTIFIED AS ICMP.
C                                                                        RGP08
      MMCH=IPREY/JOB                                                     RGP08
      MMC=MMCH*JOB                                                       RGP09
      IF(MMC.EQ.IPREY) GO TO 5802                                        RGP09
      MMCX=MMCH+1                                                        RGP09
      MOBM=MMCH*JOB                                                      RGP09
      IF(MOBM.GT.IPREY) GO TO 5803                                       RGP09
      GO TO 5804                                                         RGP09
 5803 MOBP=IPREY                                                         RGP09
      GO TO 5805                                                         RGP09
 5804 MOBP=IPREY-MOBM                                                    RGP09
 5805 MSEL=MOBP*100                                                      RGP09
      MSEX=MSEL+MMCX                                                     RGP09
      GO TO 5806                                                         RGP09
 5802 MSEL=JOB*100                                                       RGP09
      MSEX=MSEL+MMCH                                                     RGP09
 5806 CONTINUE                                                           RGP09
      DO 3429 IOOB=1,JOB                                                 RGP09
      DO 3428 JOOB=1,M                                                   RGP09
      IF(IO(IOOB,JOOB).EQ.MSEX) GO TO 3427                               RGP09
      GO TO 3428                                                         RGP09
 3427 KNOX=ICMP(IOOB,JOOB)                                               RGP09
 3428 CONTINUE                                                           RGP09
 3429 CONTINUE                                                           RGP09
C                                                                        RGP09
C     WE'VE NOW IDENTIFIED THE NODE WE ARE INVESTIGATING.                RGP09
C                                                                        RGP09
C     NOW, WE UPDATE ICTM W.R.T. THE OTHER NODES IN                      RGP09
C     THE CONFLICT SET.                                                  RGP09
C                                                                        RGP09
      DO 7050 IKEY=1,JOB                                                 RGP09
      DO 7051 JKEY=1,M                                                   RGP09
      MSEF=MSEX/100                                                      RGP09
      IF(IKEY.EQ.MSEF) GO TO 7050                                        RGP09
      MSSX=MSEX/100                                                      RGP09
      MSSL=MSSX*100                                                      RGP09
      MSMS=MSEX-MSSL                                                     RGP09
      MXSX=IO(IKEY,JKEY)/100                                             RGP09
      MSLS=MXSX*100                                                      RGP09
      MSSM=IO(IKEY,JKEY)-MSLS                                            RGP09
      IF(MSSM.EQ.MSMS) GO TO 7049                                        RGP09
      GO TO 7051                                                         RGP09
 7049 MXOL=IO(IKEY,JKEY)/100                                             RGP09
```

```
      MXZZP=IO(IKEY,JKEY)-MXOL*100                                      RGP09
      MZZZP=MXZZP-1                                                     RGP09
      MZNX=MZZZP*JOB                                                    RGP09
      JTRY=MZNX+MXOL                                                    RGP09
      IF(ENTER(JTRY).EQ.0) GO TO 7048                                   RGP09
      GO TO 7050                                                        RGP09
 7048 IUPTE=0                                                           RGP09
      DO 7047 JCCCD=JKEY,M                                              RGP09
      IF(JCCCD.EQ.JKEY) GO TO 7017                                      RGP09
      GO TO 7027                                                        RGP09
 7017 KIJCK=NP(IKEY,JCCCD)+KNOX                                         RGP09
      IF(KIJCK.LT.ICMP(IKEY,JCCCD)) GO TO 7015                          RGP09
      ICTM(IKEY,JCCCD)=NP(IKEY,JCCCD)+KNOX                              RGP09
      GO TO 7016                                                        RGP09
 7015 IUPTE=ICMP(IKEY,JCCCD)                                            RGP09
      GO TO 7047                                                        RGP09
 7016 IUPTE=ICTM(IKEY,JCCCD)                                            RGP09
      GO TO 7047                                                        RGP09
 7027 ICTM(IKEY,JCCCD)=IUPTE+NP(IKEY,JCCCD)                             RGP09
      IUPTE=ICTM(IKEY,JCCCD)                                            RGP09
 7047 CONTINUE                                                          RGP09
 7051 CONTINUE                                                          RGP09
 7050 CONTINUE                                                          RGP09
      RETURN                                                            RGP09
      END                                                               RGP09
```

DEVELOPMENT OF A NETWORK ALGORITHM AND ITS APPLICATION TO COMBINATORIAL PROBLEMS

by

ROBERT GARY PARKER

B. S., Kansas State University, 1968

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

This thesis is concerned with the development of an algorithm which can be used to solve combinatorial problems. The algorithm employs a network approach and is based upon the use of the schedule algebra operators. The basic concepts of the approach as well as a sample problem and formal presentation of the computational algorithm are presented.

The main application of the algorithm is made with respect to the general job shop scheduling problem. However, the extension of its applicability is demonstrated by considering three other classes of problems. These are the traveling salesman, project scheduling, and explosion problems.

A wide range of computational experiments were conducted with respect to the job shop problem. In addition, four traveling salesman problems are solved. Three main factors were considered in the evaluation of the performance of the network algorithm as it pertains to the job shop problem. They are (1) computation time, (2) quality of the solution, and (3) the number of iterations and conflicts encountered in obtaining a solution.

From the computational results, it is evident that computational time increases rapidly as problem size increases. The quality of the solution which is a measure of efficiency, appears to decrease as the number of conflicts increase. In addition, as the problem size increases, the number of iterations seems to approach the minimum of J.

Finally, further research is suggested in certain directions, the most important of which, lies in the area of increased applicability of the algorithm.