

THE MAN-MACHINE TASK ALLOCATION PROBLEM  
WITH SEQUENCING CONSIDERATIONS

A THESIS

Presented to

The Faculty of the Division of Graduate  
Studies and Research

By

Robert Lee Bulfin, Jr.


In Partial Fulfillment  
of the Requirements for the Degree  
Doctor of Philosophy  
in the School of Industrial and Systems Engineering

Georgia Institute of Technology

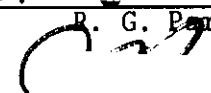
June, 1975

THE MAN-MACHINE TASK ALLOCATION PROBLEM  
WITH SEQUENCING CONSIDERATIONS

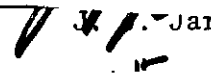
Approved:

  
\_\_\_\_\_

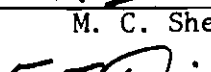
R. G. Parker, Chairman

  
\_\_\_\_\_


J. P. Jarvis

  
\_\_\_\_\_

M. C. Shetty

  
\_\_\_\_\_

F. E. Williams

  
\_\_\_\_\_

V. E. Unger

Date approved by Chairman: May 19, 1975

## ACKNOWLEDGMENTS

I would like to express my gratitude to my advisor, Dr. R. G. Parker, who introduced the topic of this dissertation to me. His guidance, friendship, and scholarly attitude have been beneficial both to the research and me.

I would also like to thank Drs. J. J. Jarvis, M. C. Shetty, and F. E. Williams. Their comments and suggestions as committee members were very helpful.

Finally, I would like to thank Dr. Ed Unger, not only for serving on the dissertation committee but for much more. Throughout my graduate studies his door was always open, and he freely gave his time to advise and assist me on many technical matters. More importantly, his understanding, friendship, and encouragement were major factors in the completion of my graduate studies.

## TABLE OF CONTENTS

|   | Page |
|---|------|
| ACKNOWLEDGMENTS . . . . .   | ii   |
| LIST OF TABLES . . . . .  | v    |
| LIST OF ILLUSTRATIONS . . . . .   | vi   |
| SUMMARY. . . . .  | vii  |
| Chapter   |      |
| I. INTRODUCTION . . . . .   | 1    |
| Combinatorial Problems  |      |
| Problem Description   |      |
| Literature Review   |      |
| II. MATHEMATICAL FORMULATION OF THE MAN-MACHINE TASK<br>ALLOCATION PROBLEM WITH SEQUENCING CONSIDERATIONS . . . . . | 14   |
| Mixed-Integer Programming Model   |      |
| Sample Problem  |      |
| Graph Theoretic Implications  |      |
| Unconstrained Problem   |      |
| III. AN ALGORITHM FOR THE GENERAL PROBLEM . . . . .   | 28   |
| Basic Concepts  |      |
| Computational Algorithm   |      |
| Sample Problem  |      |
| Computational Results   |      |
| IV. THE UNCONSTRAINED SEQUENCE-DEPENDENT PROCESSING<br>TIME PROBLEM . . . . .                                       | 58   |
| Basic Concepts  |      |
| Computational Algorithm   |      |
| Sample Problem  |      |
| Computational Results   |      |

| Chapter  | Page |
|--|------|
| V. TASK ASSIGNMENT AND SEQUENCING OF CREWS . . . . .   | 75   |
| Basic Concepts   |      |
| Computational Algorithm  |      |
| Sample Problem   |      |
| Computational Results  |      |
| VI. CONCLUSIONS AND RECOMMENDATIONS. . . . .   | 89   |
| Conclusions  |      |
| Recommendations  |      |
| APPENDIX   |      |
| A. SOLUTION OF THE SEQUENCING PROBLEMS . . . . .   | 92   |
| B. COMPUTER CODE FOR THE GENERAL PROBLEM. . . . .  | 96   |
| C. COMPUTER CODE FOR THE UNCONSTRAINED SEQUENCE<br>DEPENDENT PROCESSING TIMES PROBLEM. . . . .           | 111  |
| D. COMPUTER CODE FOR TASK ASSIGNMENT AND SEQUENCING<br>OF CREWS . . . . .                                | 119  |
| E. COMPUTER CODE FOR THE SOLUTION OF SEQUENCE DEPENDENT<br>PROCESSING TIME SCHEDULING PROBLEMS . . . . . | 129  |
| BIBLIOGRAPHY. . . . .  | 135  |
| VITA . . . . .   | 137  |

## LIST OF TABLES

| Table  | Page |
|--|------|
| 1. Generator Assembly Breakdown . . . . .  | 19   |
| 2. Data for Sample Problem 1 . . . . .   | 44   |
| 3. Computational Experience for Branch-and-Bound Algorithm .                         | 53   |
| 4. Data for Sample Problem 2 . . . . .   | 67   |
| 5. Summary of Solution Procedure Applied to Sample Problem 2                         | 70   |
| 6. Computational Results for the Sequence-Dependent Set-up<br>Time Problem . . . . . | 72   |
| 7. Data for Sample Problem 3 . . . . .   | 81   |
| 8. Computational Results for Task Assignment and<br>Sequencing of Crews . . . . .    | 87   |

## LIST OF ILLUSTRATIONS

| Figure   | Page |
|--|------|
| 1. Gantt Chart Depiction of the Generator Assembly<br>Solution . . . . .             | 22   |
| 2. Graph Theoretic Depiction of the Generator Assembly<br>Problem . . . . .          | 25   |
| 3. Graph Theoretic Depiction of the Generator Assembly<br>Problem Solution . . . . . | 26   |
| 4. Solution Tree for Sample Problem 1 . . . . .                                      | 51   |
| 5. Graph Theoretic Depiction of the Solution to Sample<br>Problem 2. . . . .         | 71   |
| 6. Solution Tree for Sample Problem 3 . . . . .                                      | 85   |

## SUMMARY

This dissertation is concerned with a particular type of scheduling problem, one involving a human as well as a machine component. Each component of the system must process certain operations of some project but, in addition, there are other operations that can be performed by either. It is assumed that all operation processing times are sequence independent but do depend on which component processes the operation. Precedence constraints, defined as conditional and absolute, are allowed. A branch-and-bound approach is developed to treat the problem where minimum total completion time is the measure of performance. Suitable computational experience is given.

A variation of this problem, called the sequence dependent problem, is also investigated. In this problem, precedence constraints are not present, but processing times are sequence dependent. An algorithm, which minimizes project duration, is developed for this problem, and computational results reported.

A second variation of the problem is also discussed. This problem has multiple distinct processors and processing times are again sequence dependent. An algorithm is developed, and computational experience is given.

## CHAPTER I

### INTRODUCTION

The discipline referred to as Operations Research is a relatively new one. It differs from most other scientific disciplines in that it is usually prescriptive in nature, rather than descriptive. Moreover, the impact of the so-called Operations Research approach has been realized in a multitude of areas, varying from traditional industrial production environments all the way to more contemporary realms in health care and social systems.

While it is often the case that one defines the applications of Operations Research techniques by sectors, e.g., manufacturing, social, etc., it is also common to identify problem types, which often arise across such sector boundaries. This research centers about a problem which belongs to the class of problems usually referred to as combinatorial problems. A discussion of these problems follows.

#### Combinatorial Problems

The problems attacked in this research are of a class referred to as combinatorial problems. In subsequent chapters, referral to such problems will be made relative to the systems in which the problems arise, e.g., scheduling problem, sequencing problem, assignment problem, and so forth. However, it remains that the models developed herein deal with, at a basic level, a problem in combinatorics. It is important to consider some general concepts of combinatorics at this

point, since many such problems have similar theoretical bases, even though their corresponding models may reflect seemingly different system applications.

Ryser [16] defines combinatorics as the analysis of the arrangement of elements into sets where the elements are usually finite and the arrangements are restricted by constraints upon specific problems. One may conclude that there are four general classes of combinatorial problems. The classes are: (1) existence problems, (2) evaluation problems, (3) enumeration problems, and (4) extremization problems. A problem of the first class above is one of determination of the existence of an arrangement. If one has ascertained the existence of an arrangement, then its identification follows. A problem of enumeration is one of determining all existing arrangements, and finally an extremization problem arises when the "best" arrangement is sought. The latter problem defines the interest of this research, relative to a class of combinatorial problems.

Most combinatorial problems can be solved by total enumeration, since there are usually a finite number of solutions. However, this number becomes large for even problems with relatively few variables. Combinatorial problems are infamous for their explosive nature; the marginal contribution of an additional variable to the computational effort of solution may be staggering. It is such a property that has rendered most analyses to practical problems unwieldy.

Consider the celebrated traveling salesman problem. Suppose that a salesman wishes to start from his home city and visit  $n$  other cities, and then return to his home city. Further suppose that he will

visit each intermediate city exactly once, and that he would like to minimize the total distance he travels. For this combinatorial problem, there are exactly  $(n-1)!$  possible routes, or tours, satisfying the above conditions. Thus for a problem with only eight cities, there are 5,040 possible tours. Adding a single city increases the number of tours to 40,320.

Another well-known problem is that of job shop sequencing. Consider a batch type shop with  $J$  jobs and  $M$  machines, where every job is to be processed once and only once on each machine. There are  $(J!)^M$  possible solutions to such a problem. Of course, some solutions may be infeasible, but the enormity of the problem remains.

There are many other problems of the combinatorial variety. It is the subject of this research to deal with such a problem, the description of which follows in the subsequent section.

#### Problem Description

Consider a job, or project to be completed, which consists of a set of tasks or operations. In the simple case, suppose there is a single man and a single machine to complete the job. (Rather than a man and a machine, one may consider two facilities.) Further, suppose that of this set of tasks, some must be done by the machine, others must be done by the human, and still others, due primarily to the flexibility of the human, could conceivably be done by either the machine or the human. The objective then, is to assign those tasks that can be done by either the machine or the human to one or the other, and then sequence all tasks for both the machine and the human in such a manner so as to optimize some measure of performance.

The problem is enhanced by the consideration of constraints on the sequence of tasks for both the machine and human. Constraints of this type are called precedence constraints. If task *i* precedes task *j*, then task *i* must be completed before task *j* may be started. However, task *j* need not immediately follow task *i*. Precedence constraints are further complicated by being of two types. One type will be referred to as unconditional precedence, and must always be satisfied. The second type, conditional precedence, is dependent upon the assignment of one or both of the tasks involved in the precedence.

Although the description of the primary problem considered in this research is man-machine, it should be specified that the system may not necessarily involve a single man and a single machine. One may use "machine" in the sense of a group of machines or a work center. Similarly, the problem may be one with two men, or only two machines. While the special case of multiple men systems or crew served systems is addressed in a later chapter, the primary description of man-machine will be used continuously to reflect the aspect of a system involving two, rather different, facilities, where such a system has inherent to it, a high degree of flexibility with reference to its performance of certain tasks. It may be of interest to demonstrate a non-intuitive system involving attributes that would lend itself to the verbal model above.

Consider an assembly operation that is done entirely by hand. Let the two facilities be the right and left hand. Certainly, many of the assembly operations could be done by either hand. There may also exist different times for doing the task with the right or left hand,

since most people have a dominant hand. Also, it is not inconceivable that constraints exist on the order in which the assembly is carried out, some of which would depend on the hand which performed the task. Recognizable as traditional right hand-left hand analysis in human factors engineering, such a system fits nicely into the framework of the problem described thus far.

The last consideration for the problem defined to this point is the measure of performance. There are several possible measures of performance which might be used. All relevant data might be converted to dollars, and then total profit could be maximized (or total cost minimized). The major drawback of such a measure would be the difficulty in quantifying many of the costs involved. A second measure might be to minimize the total processing time of all tasks. This could lead to solutions where either the machine or the human possess large amounts of idle time. The objective adopted in this research will be the minimization of project duration, or schedule time [3]. This measure makes the problem more difficult to solve, since project duration is the maximum of the completion times for all tasks comprising the project. Hence the problem becomes one of a min-max nature.

The general problem described above, as well as its specific extensions, will be treated in detail in subsequent chapters. However, prior to such developments, work that has been done on similar problems will be reviewed. Included is an overview of those problems that are related to the extent that they possess characteristics that arise in the problem addressed in this work.

### Literature Review

The man-machine task allocation problem with sequencing considerations (MMTAWSC) has four main characteristics. These characteristics are: (1) minimizing project completion, or makespan; (2) assignment of tasks to distinct processors; (3) sequencing two interrelated processors; and (4) the existence of conditional and unconditional precedence constraints both across and within processors.

There are several classes of problems which possess similar characteristics. These classes of problems will now be discussed.

#### Shop Scheduling Problems

The shop scheduling problem [4], or job shop problem, arises in the context of scheduling a set of  $J$  jobs on  $M$  machines, so that some measure of performance is optimized. Quite often, the measure of performance is to minimize makespan. In addition, each job consists of operations which must be done in a specified order on particular machines. However, the sequence in which the jobs are processed is not constrained.

The measure of performance is the same as the measure of performance used in the (MMTAWSC). The sequencing aspect is also similar to that of (MMTAWSC). It differs in that there is really no assignment to be made, since each job has a specified order of machines by which it must be processed. The technological constraints are similar to unconditional precedence, but only apply to operations, and not to jobs. No form of conditional precedence is considered.

Approaches for solving the shop scheduling problem run the gamut from single pass heuristics to mixed integer programming

formulations. Single pass procedures are generally based on a priority for each operation, which may be computed from the processing time, job completion time, or the machine completion time. The operation with the best priority is sequenced first on each machine. Giffler and Thompson [6] were among the first to propose a procedure of this type.

Another approach is to compute priorities that are legitimate bounds on the measure of performance and then incorporate a backtrack procedure in a branch-and-bound routine. This procedure will produce optimal sequences. Ashour and Hiremath [1] have developed such a procedure.

Conway et al. [4] present a mixed-integer programming model for the problem. In theory, any shop scheduling problem could be solved by applying an existing mixed-integer solution technique to this model. However, the extremely large numbers of variables and constraints required by this formulation makes this approach untenable.

#### Assembly Line Balancing Problems

An assembly line is characterized by the workpiece moving from work station to work station, with each work station performing the same operations on each workpiece. The assembly line balancing problem consists of assigning individual tasks to the work stations in such a way that some measure of performance is optimized. There may exist precedence relationships for the order in which the tasks are performed. Usual measures of performance relate to idle time, such as minimizing idle time or the number of work stations needed.

Obviously, the measures of performance used in the balancing problem and in (MMTAWSC) are quite different. The assignment decision

exists in both problems, but since all work stations are assumed "identical," then assignment in the balancing problem is not made with respect to distinct processors. The sequencing aspect of the balancing problem is trivial, since the sequencing of work stations are independent. The precedence constraints are similar to the unconditional precedence of (MMTAWSC), but conditional precedence is not considered.

Heuristic procedures based upon the precedence relationships are the most common approach. Kilbridge and Wester [8] proposed a solution procedure whereby tasks with many successors are placed in the first available station. After all tasks are assigned to a work station in this manner, tasks are traded between stations in an attempt to improve the solution.

Thangavelu and Shetty [17] have developed an integer programming formulation which minimizes the number of work stations. An adaptation of Balas' additive algorithm was used to solve this model. Due to the advantage taken of the structure of the problem, they were able to solve 50 task problems in a few seconds of computer time.

#### Parallel Processor Problems

Baker [3] describes the parallel processor problem as the problem of assigning a set of  $J$  single operation jobs to  $M$  identical machines available for processing these jobs so that makespan is minimized. In addition, there may be precedence restrictions between jobs.

In this case, the measure of performance is identical to that of (MMTAWSC), and the sequencing decision is compounded by the assignment. Also unconditional precedence can arise in both problems.

However, in the parallel processor problem, the processors are not distinct, which makes assignment much easier. Conditional precedence is not considered in the parallel processor problem. Furthermore, in all solution procedures, certain restrictive assumptions about the processing times, precedence constraints, preemption of jobs or processor sharing are made.

Hu [7] has presented a labeling scheme which produces minimum makespan solutions to the problem under certain restrictions. These restrictions are that the processing times for all jobs are the same, and that each job precedes at most one other job. A label, which corresponds to the total time for all jobs which follow the job, is computed for each job. Then the  $M$  jobs with the largest labels are sequenced on the  $M$  machines, then the next  $M$ , and so on until all jobs are sequenced. If either the processing time or precedence assumption is violated, this procedure may give poor results.

This procedure has been generalized to the case where the processing times for different jobs may be different. However, the generalizations impose other restrictive assumptions on the problem. Muntz and Coffman [12] have proposed an algorithm for the case where job splitting or preemption is allowed. They have also developed a generalization when the machines can process more than one job simultaneously [13]. Both of these assumptions are too restrictive to be usable for (MMTAWSC).

A slightly different definition of the parallel processor problem was investigated by Marsh [11]. His formulation allowed for distinct processors, but the processing times were sequence dependent.

No precedence constraints were allowed and the measure of performance was to minimize total processing time. The solution procedure used was to compute bounds for each job being sequenced on each machine, and schedule the job with the smallest bound. Included in the algorithm was a backtrack scheme similar to the one used by Little et al. [10]. This formulation is similar to the formulation of the traveling salesman problem with  $M$  salesmen.

#### Project Scheduling with Resource Constraints

Suppose there is a project, which consists of a set of operations, or tasks, to be completed. Each task requires a specified amount of a particular resource. In addition, there are limited amounts of each resource available. Unconditional precedence may exist between tasks, and it is desired to minimize project duration. This problem is called the project scheduling problem with resource constraints.

The objective of this problem, and the existence of the unconditional precedence constraints are the same as those of (MMTAWSC). The sequencing decision is compounded by the limited resources available, but is not exactly the same as found in (MMTAWSC). In this case, the assignment decision is absent, as is the existence of conditional precedence. Further, each operation of the project scheduling problem is assumed to have an earliest start time and a latest finish time. (These may be thought of as arrival times and due dates respectively.)

Many heuristics have been developed for this problem. Most of them start out by computing the earliest start and latest finish time for each task. Then, each task that can be scheduled, i.e., each task whose earliest start time is less than the current time and has all

predecessors scheduled, is placed in a set ST. Then, each element of ST is ranked. The rank may be determined by processing time, latest start time, etc. Tasks in ST are then scheduled in rank order until all the resources are used up, or ST becomes empty. The current time is now advanced and a new ST determined. This continues until all tasks are scheduled.

Pritsker, Watters, and Wolfe [14] have developed an integer-programming model for this problem. The integer variables are of the form  $X_{jt}$ , where  $j$  represents the task, and  $t$  the time period. The variable takes on the value one if task  $j$  is completed in period  $t$ , and zero otherwise. The objective is to minimize project duration subject to the restrictions on the availability of resources. The earliest start times and due dates are used to reduce the number of zero-one variables that must be considered. However, the number of jobs and the number of periods to be considered (called the planning horizon) greatly affect the number of variables and constraints.

A further extension discussed is substitutability of resources. That is, if task  $i$  requires one unit of resource  $K$ , and it is not available, then some amount of resource  $j$  may be substituted for resource  $K$  in the processing of task  $i$ . This can be construed as an assignment of task  $i$  to one of two processors. This will be discussed further in Chapter II.

Redwine and Wismer [15] have investigated a slight variation of the above model. Their model allows preemption of tasks (i.e., a task may be started in period  $j$ , nothing done on it in period  $j+1$ , and then processing resumed in period  $j+2$ ). Also, the only precedence relations

that are considered are technological in that operation  $j+1$  cannot start until operation  $j$  has started, and the percent of  $j+1$  completed must be no more than the percent of  $j$  completed. They report solving a problem with 102 orders in ten minutes on an IBM 360/91. The objective function was not to minimize makespan.

### Scope of the Research

The nature of this work is organized such that each chapter deals with developments pertaining either to the general problem described thus far or with an extension of same. For the most part, each chapter is self-contained in that algorithmic developments and respective computational experience are included.

The next chapter is devoted to further definition of the general problem. A mathematical formulation is given, as is a graph theoretic interpretation. Finally, a simplification of the general problem is discussed.

Chapter III consists of the development of a solution procedure for the general problem which utilizes the simplification discussed in the second chapter. A computational algorithm is then stated, and computational results discussed.

The fourth chapter deals with an extension of the general problem; namely dropping the precedence constraints, and assuming that the processing times for tasks are sequence dependent. A solution procedure is developed, stated, and computational results discussed.

Chapter V further extends the problem discussed in Chapter IV to allow for more than two facilities. Again, a solution procedure

is developed, stated, and computational results discussed.

The final chapter presents an overall summary of the results of the work. Included in this chapter are conclusions and extensions of the research.

## CHAPTER II

## MATHEMATICAL FORMULATION OF THE MAN-MACHINE TASK ALLOCATION

## PROBLEM WITH SEQUENCING CONSIDERATIONS

The general problem introduced in Chapter I can now be presented in a quantitative context. Let  $O$  be the set of operations or tasks which comprise the project or job in question. Further, the set  $O$  can be decomposed into three distinct subsets,  $O_{MH}$ ,  $O_M$ , and  $O_H$ . The set  $O_{MH}$  consists of those tasks which could conceivably be done by either the machine or the human, while  $O_M$  represents those tasks which can only be done by the machine, and  $O_H$  those by the human. Let  $I$ ,  $I_{MH}$ ,  $I_M$ , and  $I_H$  denote the cardinality of  $O$ ,  $O_{MH}$ ,  $O_M$ , and  $O_H$  respectively. Then the tasks may be numbered such that

$$O_{MH} = \{1, 2, \dots, I_{MH}\}$$

$$O_M = \{I_{MH}+1, I_{MH}+2, \dots, I_{MH}+I_M\}$$

$$O_H = \{I_{MH}+I_M+1, I_{MH}+I_M+2, \dots, I\}$$

Note that  $O_{MH} \subseteq O$ ,  $O_M \subseteq O$ , and  $O_H \subseteq O$ . Moreover,  $O_{MH} \cup O_M \cup O_H = O$  and  $O_{MH} \cap O_M \cap O_H = \emptyset$ . It is assumed that each task possesses an integer valued processing time that is known and deterministic. These times can be denoted by  $t_i^M \forall i \in O_M$  and  $t_i^H \forall i \in O_H$ . Tasks in  $O_{MH}$  possess two processing times:  $t_i^M$  if  $i$  is assigned to the machine, and  $t_i^H$  if  $i$  is assigned to the human.

The final consideration surrounding the problem and the one contributing most heavily to its computational severity, is that involving task sequencing or specifically, restrictions placed upon such sequencing. These restrictions, referred to as precedence constraints, fall into two categories. The first consists of absolute precedence, which implies that one operation, say  $i$ , must precede another operation, say  $j$ . This relationship can be denoted as  $i \rightarrow j$ . The following sets define all such constraints:

$$G_1^M = \{(i,j) | i \rightarrow j, i \in O_M, j \in O\}$$

$$G_1^H = \{(i,j) | i \rightarrow j, i \in O_H, j \in O\}$$

$$G_1^{MH} = \{(i,j) | i \rightarrow j, i \in O_{MH}, j \in O\}.$$

A second form of precedence, conditional precedence, also arises. Conditional precedence between task  $i$  and task  $j$ , denoted by  $i \leftrightarrow j$ , holds only if both tasks  $i$  and  $j$  are done by the human, or both by the machine. The following sets define all forms of conditional precedence considered.

$$G_2^M = \{(i,j) | i \leftrightarrow j, i \in O_M, j \in O_{MH}\}$$

$$G_2^H = \{(i,j) | i \leftrightarrow j, i \in O_H, j \in O_{MH}\}$$

$$G_2^{MH} = \{(i,j) | i \leftrightarrow j, i, j \in O_{MH}\}$$

$$\bar{G}_2^M = \{(i,j) | i \leftrightarrow j, i \in O_{MH}, j \in O_M\}$$

$$\overline{G}_2^H = \{(i,j) | i \leftrightarrow j, i \in O_{MH}, j \in O_H\}.$$

Consider an ordered pair  $(i,j)$ , such that  $(i,j) \in G_1^M \cup G_1^H \cup G_1^{MH}$ . In this case, task  $i$  must precede task  $j$ . However, if  $(i,j) \in G_2^M$ , task  $i$  must precede if and only if task  $j$  is assigned to the machine. Note that  $G_2^H$  is similar to  $G_2^M$  except that precedence holds if and only if task  $j$  is assigned to the human. The precedence set  $G_2^{MH}$  represents precedence which is contingent on both task  $i$  and task  $j$  being assigned to the machine, or both to the human. The sets  $\overline{G}_2^M$  and  $\overline{G}_2^H$  are analogous to  $G_2^M$  and  $G_2^H$  except that precedence holds with reference to the assignment of task  $i$  to the machine and human respectively.

#### Mixed-Integer Programming Model

Using the conventions adopted above, the general problem can be formulated as a mixed-integer programming model. It should be noted that the model is conceptually similar to those previously formulated for related problems [4]. Define

$$s_i = \text{start time of task } i$$

$$x_i = \begin{cases} 1 & \text{if } i \in O_{MH} \text{ and task } i \text{ is assigned to the machine} \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ij} = \begin{cases} 1 & \text{if } i \text{ is sequenced before } j \\ 0 & \text{otherwise} \end{cases}$$

The objective is to minimize project completion time, which shall be denoted by  $F$ . The following formulation can now be given.

Minimize F  
Subject to

$$s_i + t_i^M x_i + t_i^H (1 - x_i) \leq F ; \quad i \in O_{MH}$$

$$s_i + t_i^M \leq F ; \quad i \in O_M$$

$$s_i + t_i^H \leq F ; \quad i \in O_H$$

$$s_i + t_i^M \leq s_j ; \quad (i,j) \in G_1^M$$

$$s_i + t_i^H \leq s_j ; \quad (i,j) \in G_1^H$$

$$s_i + t_i^M x_i + t_i^H (1 - x_i) \leq s_j ; \quad (i,j) \in G_1^{MH}$$

$$s_i + t_i^M \leq s_j + Q(1 - x_j) ; \quad (i,j) \in G_2^M$$

$$s_i + t_i^H \leq s_j + Qx_j ; \quad (i,j) \in G_2^H$$

$$\left. \begin{array}{l} s_i + t_i^M \leq s_j + Q(1 - x_i) + Q(1 - x_j) \\ s_i + t_i^H \leq s_j + Qx_i + Qx_j \end{array} \right\} (i,j) \in G_2^{MH}$$

$$s_i + t_i^M \leq s_j + Q(1 - x_i) ; \quad (i,j) \in \overline{G}_2^M$$

$$s_i + t_i^H \leq s_j + Qx_i ; \quad (i,j) \in \overline{G}_2^H$$

$$\left. \begin{array}{l} s_i + t_i^M \leq s_j + Q(1 - y_{ij}) + Qx_i + Qx_j \\ s_j + t_j^M \leq s_i + Qy_{ij} + Qx_i + Qx_j \end{array} \right\} \begin{array}{l} i,j \in O_{MH} \\ i < j \end{array}$$

$$\left. \begin{array}{l} s_i + t_i^H \leq s_j + Q(1 - y_{ij}) + Q(1 - x_i) + Q(1 - x_j) \\ x_j + t_j^H \leq s_i + Qy_{ij} + Q(1 - x_i) + Q(1 - x_j) \end{array} \right\} \begin{array}{l} i,j \in O_{MH} \\ i < j \end{array}$$

$$\left. \begin{array}{l} s_i + t_i^M \leq s_j + Q(1 - y_{ij}) \\ s_j + t_j^M \leq s_i + Qy_{ij} \end{array} \right\} \begin{array}{l} i, j \in O_M \\ i < j \end{array}$$

$$\left. \begin{array}{l} s_i + t_i^H \leq s_j + Q(1 - y_{ij}) \\ s_j + t_j^H \leq s_i + Qy_{ij} \end{array} \right\} \begin{array}{l} i, j \in O_H \\ i < j \end{array}$$

$$\left. \begin{array}{l} s_i + t_i^M \leq s_j + Q(1 - y_{ij}) + Q(1 - x_i) \\ s_j + t_j^M \leq s_i + Qy_{ij} + Q(1 - x_i) \end{array} \right\} \begin{array}{l} i \in O_{MH} \\ j \in O_M \end{array}$$

$$\left. \begin{array}{l} s_i + t_i^H \leq s_j + Q(1 - y_{ij}) + Qx_i \\ s_j + t_j^H \leq s_i + Qy_{ij} + Qx_i \end{array} \right\} \begin{array}{l} i \in O_{MH} \\ j \in O_H \end{array}$$

$$s_i \geq 0 ; i \in O$$

$$x_i \in \{0,1\} ; i \in O_{MH}$$

$$y_{ij} \in \{0,1\} ; i, j \in O, i < j .$$

Note that  $Q = I \cdot \max_{i \in O} \{t_i^M, t_i^H\}$ . The above model can be illustrated in a specific context by considering a small sample problem.

#### Sample Problem

Consider the project of constructing an electric generator unit. The following table can be given which provides operation breakdown and appropriate processing times. Note in Table 1, that each operation is placed into one of three sets,  $O_{MH}$ ,  $O_M$ , or  $O_H$ . Moreover, the processing times have been scaled by a factor of 1/10.

Table 1. Generator Assembly Breakdown

| Operation Index, i | Operation Description           | Set Classification | Processing Time (min/10) |
|--------------------|---------------------------------|--------------------|--------------------------|
| 1                  | Test completed assembly         | $0'_{MH}$          | $3(0'_H); 2(0'_M)$       |
| 2                  | Paint the base                  | $0_M$              | 5                        |
| 3                  | Cut channel irons for base      | $0_H$              | 3                        |
| 4                  | Drill mounting holes in base    | $0_H$              | 2                        |
| 5                  | Weld base together              | $0_H$              | 2                        |
| 6                  | Assemble generator unit on base | $0_H$              | 30                       |

The constraints for this problem can be specified such that,

$$G_1^H = \{(3,1), (3,5), (3,6), (4,1), (4,6), (5,1), (5,2), (5,6), (6,1)\}$$

and

$$G_1^M = G_1^{MH} = G_2^M = G_2^H = G_2^{MH} = \overline{G}_2^M = \overline{G}_2^H = \emptyset$$

The complete model then is given as follows:

Minimize F  
Subject to

$$s_1 + 3x_1 + 2(1 - x_1) \leq F$$

$$s_2 + 5 \leq F$$

$$s_3 + 3 \leq F$$

$$s_4 + 2 \leq F$$

$$s_5 + 2 \leq F$$

$$s_6 + 30 \leq F$$

$$s_3 + 3 \leq s_1$$

$$s_3 + 3 \leq s_5$$

$$s_3 + 3 \leq s_6$$

$$s_4 + 2 \leq s_1$$

$$s_4 + 2 \leq s_6$$

$$s_5 + 2 \leq s_1$$

$$s_5 + 2 \leq s_2$$

$$s_5 + 2 \leq s_6$$

$$s_6 + 30 \leq s_1$$

$$s_1 + 3 \leq s_2 + 1000(1 - y_{12}) + 1000(1 - x_1)$$

$$s_2 + 5 \leq s_1 + 1000 y_{12} + 1000(1 - x_1)$$

$$s_1 + 2 \leq s_3 + 1000(1 - y_{13}) + 1000 x_1$$

$$s_3 + 3 \leq s_1 + 1000 y_{13} + 1000 x_1$$

$$s_1 + 2 \leq s_4 + 1000(1 - y_{14}) + 1000 x_1$$

$$s_1 + 2 \leq s_3 + 1000(1 - y_{15}) + 1000 x_1$$

$$s_5 + 2 \leq s_1 + 1000 y_{15} + 1000 x_1$$

$$s_1 + 2 \leq s_6 + 1000(1 - y_{16}) + 1000 x_1$$

$$s_6 + 30 \leq s_1 + 1000 y_{15} + 1000 x_1$$

$$s_3 + 3 \leq s_4 + 1000(1 - y_{34})$$

$$s_4 + 2 \leq s_3 + 1000 y_{34}$$

$$s_3 + 3 \leq s_3 + 1000(1 - y_{35})$$

$$s_5 + 2 \leq s_3 + 1000 y_{35}$$

$$s_3 + 3 \leq s_6 + 1000(1 - y_{36})$$

$$s_6 + 30 \leq s_3 + 1000 y_{36}$$

$$s_4 + 2 \leq s_5 + 1000(1 - y_{45})$$

$$s_5 + 2 \leq s_4 + 1000 y_{45}$$

$$s_4 + 2 \leq s_6 + 1000(1 - y_{46})$$

$$s_6 + 30 \leq s_4 + 1000 y_{46}$$

$$s_5 + 2 \leq s_6 + 1000(1 - y_{56})$$

$$s_6 + 30 \leq s_5 + 1000 y_{56}$$

$$s_i \geq 0 \quad i \in 0$$

$$x_i \in \{0,1\}$$

$$y_{ij} \in \{0,1\} \quad i = 1,3,4,5, \quad j = 2,3,4,5,6 .$$

This problem was run on a Univac 1108 using a linear programming based branch-and-bound mixed-integer code. The solution time was approximately three seconds. The problem solution is 39 (\*10) time units, and can be depicted by the Gantt Chart in Figure 1.

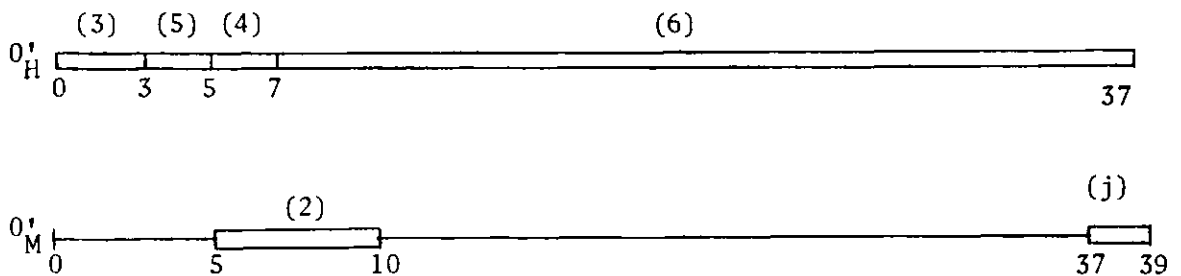


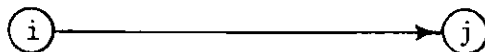
Figure 1. Gantt Chart Depiction of the Generator Assembly Solution

It is obvious that the growth in the number of variables and constraints is rapid enough to make realistically sized problems computationally prohibitive. Other formulations might be investigated, but they too exhibit the same growth. For example, the problem could be formulated as a modification of the Pritsker et al. model discussed in Chapter I. However, since all tasks are initially available, and due dates do not exist for any task, the planning horizon becomes large. (Note that the planning horizon must be at least as large as the optimal solution.) Thus the number of variables and constraints

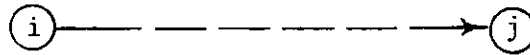
again become excessive. In the sample problem discussed above, there would be more than 240 zero-one variables and 90 constraints. For really large problems (e.g.  $|0_{MH}^i| = 50$ ,  $|0_M| = 50$ ,  $|0_H| = 50$ ) there may well be over a half million variables and thousands of constraints if the Pritsker et al. model is used. (It is true that the number of variables and constraints can be reduced by computing the earliest start and finish times for each task, but in the presence of different processing times for elements of  $0_{MH}$  and the conditional constraints, this may not significantly reduce the dimensions.) Even using decomposition, problems of this size are not easily solved. Therefore, other approaches are necessary in order to solve large scale assignment/sequencing problems.

#### Graph Theoretic Implications

It is of interest to note the graph theoretic construction for the problem discussed thus far. Consider a graph  $G(N;A)$  such that every  $i \in 0$  defines a member of  $N$ . In addition, consider the set  $A$  to consist of all ordered pairs  $(i,j)$  which represent either conditional or unconditional precedence as well as potential sequencing relationships. The ordered pairs or edges  $(i,j) \in A$  are of two forms: conjunctive or disjunctive. Relationships reflecting unconditional precedence are given by conjunctive or deterministic arcs (edges) as depicted below.



All arcs of the conjunctive variety are collected in set  $C$ . Relationships of conditional precedence are given by the set  $D'$  and are depicted as follows:



where  $(i,j) \in D'$  specifies that  $i$  precedes  $j$  relative to some assignment. Potential sequencing relationships are denoted by disjunctive pairs of arcs and are given by the set  $D''$ . A typical disjunctive pair is depicted below.



where  $(i,j) \in D''$ . It is clear that the general problem can be given by a graph  $G(N;C,D';D'')$ .

The solution of the problem specifies that a feasible assignment over  $O_{MH}$  be made and the resulting tasks be sequenced over the updated sets  $O'_M$  and  $O'_H$  such that the completion time of the entire set of tasks in  $O$  be made minimum or near minimum. Such an objective involves the synthesis of the initial graph such that all disjunctive relationships be either made conjunctive or deleted. Of course the maximal length path through the resulting graph specifies the completion time solution, while the elements of  $A^*$  yield the sequential properties of all operations. Note that  $A^*$  is the final set of edges after the above synthesis is completed.

Observe in Figure 2 the initial graphic representation for the sample problem. The numbers on each arc reflect the processing times

of the tasks (nodes) from which the arc emanates. The graph in Figure 3 depicts the final solution determined from the integer program. The numbers below each node reflect the completion time of the respective task. Obviously node (1) is unsucceeded, and its completion time defines the completion time of the entire set of tasks or operations. Note that several arcs in Figure 3 are redundant.

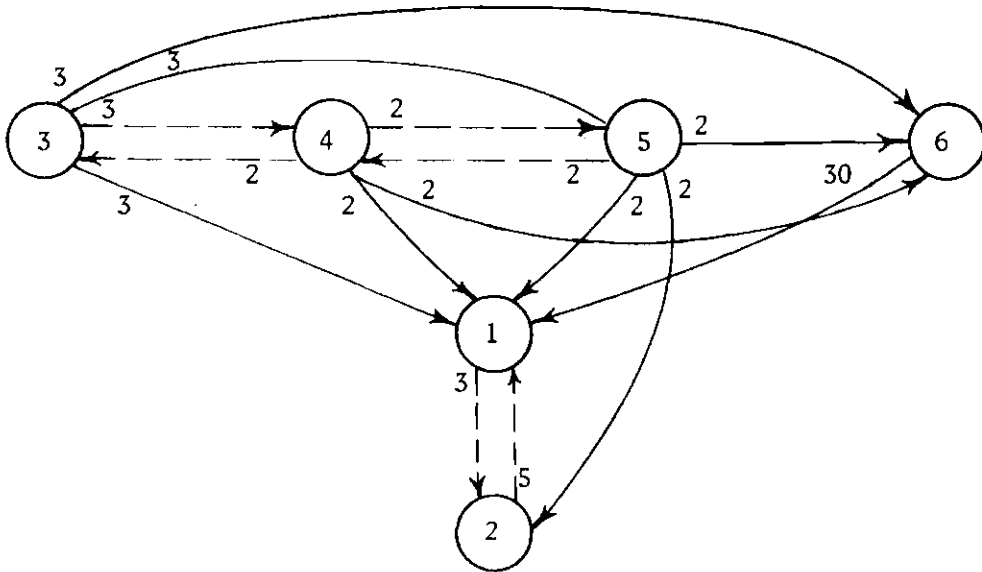


Figure 2. Graph Theoretic Depiction of the Generator Assembly Problem

### Unconstrained Problem

A simplification of the general problem can be given such that all precedence constraints are dropped. The effect of this simplification is to reduce the problem to one of assignment only, since the processing times considered thus far are considered to be sequence independent. The mixed-integer programming model of the unconstrained problem is:

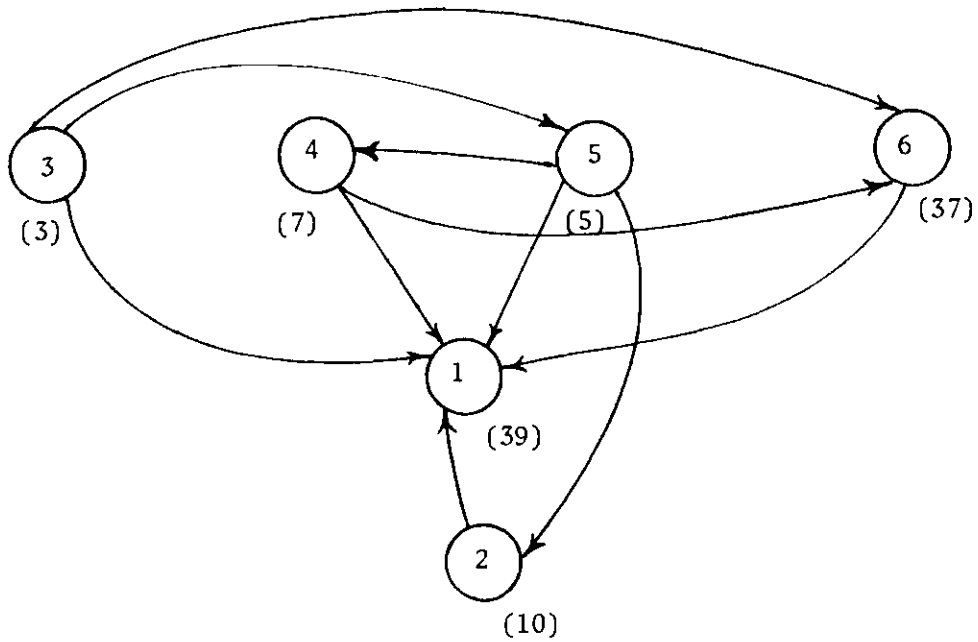


Figure 3. Graph Theoretic Depiction of the Generator Assembly Problem Solution

Minimize  $z$   
 Subject to:

$$z \geq \sum_{i \in 0_M} t_i^M + \sum_{i \in 0_{MH}} t_i^M x_i$$

$$z \geq \sum_{i \in 0_H \cup 0_{MH}} t_i^H - \sum_{i \in 0_{MH}} t_i^H x_i$$

$$x_i \in \{0,1\} \quad i \in 0_{MH}$$

where all variables are as previously defined. This problem has more interest from a theoretical viewpoint than from a practitioner's viewpoint, since few practical problems will fit this model. Again,

several problems were run using the algorithm mentioned above. Solution times for 30-variable problems were under one second.

## CHAPTER III

## AN ALGORITHM FOR THE GENERAL PROBLEM

The mixed integer programming formulation of the general problem introduced in the previous section holds little promise for the solution of realistically sized problems. The rapid increase in the number of constraints causes even moderately sized problems to become computationally untractable. Consequently, an alternative approach which exploits the special structure of the problem will be explored currently.

The principal factors affecting the difficulty of solution of the general problem are, of course, the assignment/sequencing decisions, in conjunction with the conditional precedence constraints. Since the problem is of a combinatorial nature, a decomposition scheme, whereby smaller, or less difficult, problems are solved repeatedly in lieu of solving the entire problem, would seem to hold promise.

If the problem is decomposed into one of assignment and then sequencing care must be taken to insure that a poor assignment is not the only one considered. The particular structure of the conditional precedence constraints causes a sequence to be highly dependent on an assignment. However, if sequences are constructed for all assignments, and the best one chosen, such a problem is overcome. However, this approach is computationally infeasible, since the enumeration of all assignments is, in itself, an untenable task. The approach to be pursued herein will be to generate only those assignments which could possibly yield better sequencing solutions than some incumbent

assignment, which, hopefully, will drastically reduce the number of assignments explored.

This approach has several advantages over the mixed-integer model. First, there is no need to make assignment decisions concurrently with sequencing decisions. Second, and of more importance, it is clear that for a given assignment, conditional precedence constraints either become unconditional, or drop out of the problem entirely.

### Basic Concepts

The above approach can be implemented in one of two ways: either by branch-and-bound, or by implicit enumeration. These two approaches are conceptually similar, but differ in their implementation. Branch-and-bound requires considerably more storage than implicit enumeration. However, the additional storage allows the branch-and-bound scheme to be very flexible in the order that candidate problems are explored, which is not true of implicit enumeration procedures. Since computer storage does not appear to be a problem, the greater flexibility of the branch-and-bound approach makes it more attractive than implicit enumeration.

Branch-and-bound schemes are based on a divide and conquer strategy. The two important facets of the procedure are separation and bounding. Since separation depends upon the bounds, bounding will be discussed first.

Recall that two factors are involved in the determination of schedule time. One is the actual assignment, while the other is the precedence relationships. If there are several assignments of equal

schedule times ignoring the precedence constraints, it is likely that for one of these assignments, there exists a sequence which satisfies the precedence constraints, without appreciably increasing the schedule time. If this is true, ignoring the precedence constraints and obtaining bounds only on the assignment might yield good results.

Ignoring the precedence constraints leaves the unconstrained problem (UP) discussed in Chapter II. For (UP), a sequence for both the machine and the human can be constructed such that no starting time of any task is delayed. Thus, given an assignment, the schedule time is given by the maximum of the completion time of the machine and the human. Inclusion of precedence constraints cannot give better completion times, and may actually introduce idle time and give a greater schedule time. Therefore the solution of (UP) gives a lower bound on the solution of (GP). Since (UP) is an integer programming problem, which is also difficult to solve, a relaxation is used. This relaxation is to drop the integrality requirements on the  $x_i$ 's, and replace them by upper bounds of one. The mathematical formulation of this relaxed problem is

$$\begin{array}{ll} \text{Min} & z \\ \text{(RUP) S.T.} & z \geq \sum_{i \in 0_M} t_i^M + \sum_{i \in 0_{MH}} t_i^M x_i \end{array} \quad (C1)$$

$$z \geq \sum_{i \in 0_H \cup 0_{MH}} t_i^H - \sum_{i \in 0_{MH}} t_i^H x_i \quad (C2)$$

$$0 \leq x_i \leq 1 \quad i \in 0_{MH}$$

At optimality, either (C1), (C2) or both must hold as an equality. Assume that (C1) is satisfied as an equality. Then the problem can be restated as follows:

$$\begin{aligned} \text{Min } z_1 &= \sum_{i \in O_M} t_i^M + \sum_{i \in O_{MH}} t_i^M x_i \\ \text{(RUP1) S.T. } & \sum_{i \in O_M} t_i^M + \sum_{i \in O_{MH}} t_i^M x_i \geq \sum_{i \in O_{H \cup O_{MH}}} t_i^H - \sum_{i \in O_{MH}} t_i^H x_i \\ & 0 \leq x_i \leq 1 \quad i \in O_{MH} \end{aligned}$$

Likewise, problem (RUP2) occurs if (C2) holds as an equality.

$$\begin{aligned} \text{Min } z_2 &= \sum_{i \in O_{H \cup O_{MH}}} t_i^H - \sum_{i \in O_{MH}} t_i^H x_i \\ \text{(RUP2) S.T. } & \sum_{i \in O_{H \cup O_{MH}}} t_i^H - \sum_{i \in O_{MH}} t_i^H x_i \geq \sum_{i \in O_M} t_i^M + \sum_{i \in O_{MH}} t_i^M x_i \\ & 0 \leq x_i \leq 1 \quad i \in O_{MH} \end{aligned}$$

Therefore, if  $z_1$  and  $z_2$  are the solutions to (RUP1) and (RUP2), then  $z = \min\{z_1, z_2\}$  must be the solution of (RUP). (Note that if either (RUP1) or (RUP2) is infeasible, its solution value is assumed infinite.)

Now note that by combining terms, each problem has the form

$$\text{Min} \quad \sum_{i \in 0_{MH}} c_i x_i$$

$$\text{S.T.} \quad \sum_{i \in 0_{MH}} a_i x_i \geq b$$

$$0 \leq x_i \leq 1 \quad i \in 0_{MH},$$

which is the well-known knapsack problem. To solve this problem, one simply ranks the ratios,  $c_i/a_i$ , in ascending order. Then, selecting the variable with the smallest ratio, allocate the maximum of the variable's upper bound (in this case one), or the fraction which causes the constraint to be satisfied as an equality. Thus, only one  $x_i$  will take on a fractional value. This variable also defines all values for the other variables, since all variables with smaller ratios will have value one, and all with larger ratios will have value zero.

The problems (RUP1) and (RUP2) can be rewritten as follows:

$$\text{Min} \quad \sum_{i \in 0_{MH}} t_i^M x_i$$

(RUP1)

$$\text{S.T.} \quad \sum_{i \in 0_{MH}} a_i x_i \geq b$$

$$0 \leq x_i \leq 1, \quad i \in 0_{MH}$$

and

$$\text{Max} \quad \sum_{i \in 0_{MH}} t_i^H x_i$$

$$\begin{aligned}
 \text{(RUP2)} \quad & \text{S.T.} \quad \sum_{i \in O_{MH}} a_i x_i \leq b \\
 & 0 \leq x_i \leq 1, \quad i \in O_{MH}
 \end{aligned}$$

where,

$$\begin{aligned}
 a_i &= t_i^M + t_i^H \\
 b &= \sum_{i \in O_H \cup O_{MH}} t_i^H - \sum_{i \in O_M} t_i^M
 \end{aligned}$$

To solve a maximization knapsack problem such as (RUP2), simply rank the ratios in descending order, and allocate as before, except that now the variable with the largest ratio is taken first, the second largest ratio next, and so forth. Tied ratios may be chosen arbitrarily.

The coefficients of (RUP1) and (RUP2) are related in such a way that the two rankings are the same, since  $t_i^M / (t_i^M + t_i^H) = 1 - [t_i^H / (t_i^M + t_i^H)]$ . Therefore, the decision variables for the two problems will have the same values, since  $b$  and all  $a_i$ 's are the same for each problem. Denote the value of the optimal solution variables by  $x_i^*$ . Thus the solution value is given by

$$z = \min \left\{ \sum_{i \in O_M} t_i^M + \sum_{i \in O_{MH}} t_i^M x_i^*, \sum_{i \in O_H \cup O_{MH}} t_i^H - \sum_{i \in O_{MH}} t_i^H x_i^* \right\},$$

Since all processing times are integer valued,  $z$  must be integer valued, and hence any  $z$  which is non-integer may be rounded up. Clearly,  $z$  is a lower bound on the schedule time.

Notation used in the branch-and-bound procedure will now be discussed. If a variable is restricted to take on a particular value, it is said to be fixed. If a variable can take on either of the values zero or one, it is said to be free. A subset of variables that is composed of fixed variables is called a partial solution. If  $x_i^*$  is the solution to (RUP) with some variables fixed, (i.e., a partial solution), then  $x_i^C = 1$  if  $x_i^* = 1$  and  $x_i^C = 0$  otherwise is called a completion of the partial solution. A candidate problem is specified by its partial solution, its completion, and the index of its fractional variable. All candidate problems will be stored in a candidate list, which will be denoted by CL.

For a particular candidate problem, denoted by  $CP_\ell$ , where  $\ell \in CL$ , the fixed variables will be denoted by an indicator  $F_{\ell,i}$ ,  $i \in 0_{MH}$ . If  $F_{\ell,i} = 1$ ; then variable  $i$  is fixed in candidate problem  $\ell$ , otherwise, if  $F_{\ell,i} = 0$ , variable  $i$  is free in candidate problem  $\ell$ . The partial solution and its completion will be denoted by

$$x_{\ell,i} = \begin{cases} 1 & \text{if } x_i^* = 1 \\ 0 & \text{otherwise} \end{cases}$$

The fractional variable's index is denoted by  $IFR_\ell$ , and the incumbent solution is denoted by  $z^+$ .

Denote the procedure for finding a solution for (RUP) using (RUP1) and (RUP2) by  $(KP_\emptyset)$ . Then for a given partial solution, say the one determined by  $CP_\ell$ , the solution procedure will be denoted by  $(KP_\ell)$ . This could be done by setting fixed variables at their appropriate values, and solving the reduced problem. However, this is unnecessary.

Consider a problem where only one variable is to be fixed which was not previously fixed in another candidate problem  $CP_c$ . If  $x_k = 1$  and is free in the solution of  $(KP_c)$ , then fixing  $x_k = 1$  does not affect the solution of  $(KP_c)$ . Fixing  $x_k = 0$ , the new solution is readily found, since the constraint of  $(KP_c)$  is now undersatisfied by an amount  $a_k$ . In such a case, simply begin with the fractional valued variable, and increase the variables in ranked order until the constraint is again satisfied. If a variable is free at zero value, and is fixed at value one, the constraint is then oversatisfied, and a similar process is used, but in this case the variables are reduced until the constraint is satisfied as an equality. In this case the fractional valued variable is again the starting variable, and the process continues in reverse ranked order. Thus, starting with  $(KP_\emptyset)$ , each time a variable is fixed, the bound for that candidate problem is readily obtained.

The aim of the branch-and-bound procedure will be to start with a particular candidate problem, and drive to a feasible assignment. A feasible assignment is defined as a candidate problem which has  $F_{\ell,i} = 1 \forall i \in O_{MH}$ . The process of fixing variables is accomplished by choosing a separating variable, fixing it at its free value in the current candidate problem, and then storing a newly created candidate problem with the variable fixed at the complemented free value, in a candidate list for later exploration. A bound on each candidate problem is computed and if the bound is no better than the incumbent solution, the candidate problem need not be explored further. When a candidate problem becomes feasible, a sequence for that assignment is determined. If the new sequence is better than the incumbent, a

replacement is specified. When a new incumbent is obtained, all candidate problems in the candidate list which have bounds that are no better than the new incumbent are deleted. In any case, a new candidate problem is chosen, if one exists, such that the new problem is always the one with the lowest bound. The procedure terminates when the candidate list is empty.

It should be clear that the determination of bounds is critical. If good bounds are obtained, many of the possible assignments will not be explored, since the bounds on such candidate problems will be higher than the incumbent solution.

These bounds are also used to choose separating variables. Separating variables will be chosen such that the candidate problems created will have large bounds. This strategy is suggested by Tomlin [18], and empirical studies have shown that it is usually a superior strategy. Intuitively, if the ratio  $t_i^M / (t_i^M + t_i^H)$  is small, the value of  $t_i^H$  must dominate that of  $t_i^M$ , hence the best policy would be to assign  $i$  to the machine. A similar argument for assigning variables with large ratios to the human can be made. Also, the size of the ratio is an indication of how appealing the assignment to either the machine or the human is. Thus, taking the smallest ratio and assigning that task to the human should result in a large bound for the candidate problem created, which, hopefully, will exclude it from further consideration. Therefore candidates for separation will be the free variables with the largest and smallest ratios. Bounds are computed (by solving  $(KP_\ell)$ ) with each of these variables fixed at their complemented value. The variable which gives the higher bound is the chosen

as the separating variable.

The nature of the branch-and-bound procedure discussed above gives rise to a simplification. Since the chosen candidate problem is kept until it can give no better solution than the incumbent, (or until it becomes feasible), each separating variable is derived from the same parent problem. Hence, if the fractional valued variable is the last variable fixed, only one of the separation candidates needs to be computed, since the one that was not chosen at the previous stage is still a valid candidate with the same bound as was computed before. Of course, when a new candidate problem is chosen from the candidate list, both separating variables must be used to compute bounds.

Finally, the subject of precedence constraints and sequencing must be explored. For a given assignment, the set of precedence constraints,  $P$ , which must hold can be defined as:

$$P = G_1^M \cup G_1^H \cup G_1^{MH} \cup G_3^M \cup G_3^H \cup G_3^{MH} \cup \overline{G}_3^M \cup \overline{G}_3^H$$

where

$$G_3^M = \{(i,j) \mid (i,j) \in G_2^M, x_j = 1\}$$

$$G_3^H = \{(i,j) \mid (i,j) \in G_2^H, x_j = 0\}$$

$$G_3^{MH} = \{(i,j) \mid (i,j) \in G_2^{MH}, x_i = x_j\}$$

$$\overline{G}_3^M = \{(i,j) \mid (i,j) \in \overline{G}_2^M, x_i = 1\}$$

$$\overline{G}_3^H = \{(i,j) \mid (i,j) \in \overline{G}_2^H, x_i = 0\}.$$

Given the precedence set  $P$ , a sequence for the assignment can be determined. Let  $S_M$  be the set of all tasks which are available for sequencing on the machine.  $S_M$  is called the candidate set for the machine, and a task is included in this set if and only if all tasks which precede it have been scheduled.  $S_H$  is defined in a similar manner for the human. Define  $CM$  and  $CH$  as the completion times of the last tasks scheduled on the machine and human respectively. Also, let  $\bar{O}'_M$  and  $\bar{O}'_H$  be the sets of unscheduled tasks for the machine and human respectively.

For each candidate task  $i$ , compute the earliest start time,  $ES_i$ , where  $ES_i$  is the maximum of the completion times of all predecessors of  $i$ , and  $CM$  if  $i \in \bar{O}'_M$  or  $CH$  if  $i \in \bar{O}'_H$ . Note that  $ES_i = CM$  if  $i \in \bar{O}'_M$  or  $ES_i = CH$  if  $i \in \bar{O}'_H$ , unless there exists some  $(k,i) \in P$  such that  $k \in \bar{O}'_H$  if  $i \in \bar{O}'_M$ , or  $k \in \bar{O}'_M$  if  $i \in \bar{O}'_H$ .

For each candidate task, bounds on the schedule time are computed, and the minimum bound task on the machine and human respectively are scheduled. The procedure continues until all tasks are scheduled.

Bounds are obtained as follows:

Let  $T = |O| + 1$ . Construct a quantified precedence graph with  $T$  nodes. Let node  $i$  correspond to task  $i$  for all  $i \in O$ , and node  $T$  correspond to a terminal node. If  $(i,j) \in P$ , include an arc from node  $i$  to node  $j$  with a time of  $t_i = t_i^M x_i + t_i^H (1 - x_i)$  units. Also include an arc from node  $i$ , for all  $i \in O$ , to node  $T$ , with a time of  $t_i^M x_i + t_i^H (1 - x_i)$ . Let  $D_{kT}$  be the length of the longest path from node  $k$  to node  $T$ . Also define  $LPM = \max_k [D_{kT}]$  and  $LPH = \max_k [D_{kT}]$ . Then, for  $i \in S_M$ , the lower bound on schedule time arises as the maximum of the following

four bounds.

- 1)  $CH + \sum_{i \in \bar{0}'_H} t_i^H$
- 2)  $CH + LPH$
- 3)  $ES_i + t_i^M + \max_{\substack{k \in \bar{0}'_M \\ k \neq i}} D_{kT}$
- 4)  $ES_i + \sum_{i \in \bar{0}'_M} t_i^M$

Similar bounds are computed for elements of  $S_H$ .

It should be noted that the procedure for solving the sequencing problem (as outlined above) is considered in a first pass mode. No backtracking over active schedules [3] is considered. Hence the procedure does not guarantee optimal sequences. A formal statement of the computational algorithm can now be given.

#### Computational Algorithm

Consider the following step-by-step statement of the procedure discussed above.

##### Step 0: Initialization

- 0.1 Compute  $a_i = t_i^M + t_i^H$ , and let  $r_i = t_i^M/a_i$ ,  $\forall i \in \bar{0}_{MH}$ .
- 0.2 Rank the assignment variables in ascending order of the  $r_i$ 's and solve  $(KP_\emptyset)$ . Denote the solution by  $x_i^*$ ,  $i \in \bar{0}_{MH}$ .
- 0.3 Set  $z^+ = \infty$  and let  $BND_1 = z_{(KP_\emptyset)}$ , and let  $IFR_1$  be the index of the fractional assignment. Let

$$x_{1,i} = \begin{cases} 1 & \text{if } x_i^* = 1 \\ 0 & \text{otherwise} \end{cases} \quad \forall i \in 0_{MH}$$

$$F_{1,i} = 0 \quad \forall i \in 0_{MH}$$

Step 1: Select a candidate problem

- 1.1 If  $CL = \phi$ , the procedure terminates.
- 1.2 Let  $BND_C = \min_{\ell \in CL} BND_\ell$ .  $CP_C$  is the candidate problem to be explored.
- 1.3 Let  $0'_M = 0_M \cup \{i | i \in 0_{MH}, x_{c,i} = 1, F_{c,i} = 1\}$ ,  
 $0'_H = 0_H \cup \{i | i \in 0_{MH}, x_{c,i} = 0, F_{c,i} = 1\}$ .

Step 2: Select a separating variable

- 2.1 Let  $Q = \{i | i \in 0_{MH}, F_{c,i} = 0, i \neq IFR_C\}$ . If  $Q \neq \phi$  go to 2.3.
- 2.3. If  $F_{c,IFR_C} = 1$ , go to step 4.

- 2.2 Let  $IS = IFR_C$  and set

$$Z_{IS} = \max \left\{ \sum_{i \in 0_M} t_i^M + \sum_{i \in 0_{MH}} t_i^M x_i, \sum_{i \in 0_H} t_i^H + \sum_{i \in 0_{MH}} t_i^H (1 - x_i) \right\}$$

Go to step 3.

- 2.3 Let  $r_{IF} = \min_{i \in Q} r_i$  and  $r_{IB} = \max_{i \in Q} r_i$ .
- 2.4 Solve  $(KP_C)$  with variable IF complemented. Denote the solution by  $Z_{IF}$  and let  $\bar{F}_{IF}$  be the index of the fractional variable. Similarly, find  $Z_{IB}$  and  $\bar{F}_{IB}$ .
- 2.5 Let  $Z_{IS} = \max \{Z_{IF}, Z_{IB}\}$ .

Step 3: Update the candidate list if necessary

3.1 If  $Z_{IS} \geq Z^+$  go to 3.3.

3.2 Let  $\ell$  be an element of the candidate list. Set

$$IFR_{\ell} = \bar{F}_{IS}$$

$$BND_{\ell} = Z_{IS}$$

$$x_{\ell,i} = \begin{cases} x_{c,i} & \text{if } F_{c,i} = 1 \\ 1 - x_{c,i} & \text{, if } i = IS \\ 1 & \text{, if } F_{c,i} = 0 \text{ and } x_i^* = 1 \\ 0 & \text{, if } F_{c,i} = 0 \text{ and } x_i^* < 1 . \end{cases}$$

Set  $F_{c,IS} = 1$  and add IS to  $O_M'$  if  $x_{c,IS} = 1$ . Otherwise, add IS to  $O_H'$ . Let  $F_{\ell,i} = F_{c,i}$ ,  $\forall i \in O_{MH}$ .

3.3 If  $F_{c,i} = 0$  for some  $i \in O_{MH}$ , go to step 2.

Step 4: Solve the sequencing problem

4.1 Construct P, the set of precedence relationships for the particular assignment and set

$$ES_i = NP_i = 0, \forall i \in O$$

$$CM = CH = 0$$

$$S_M = S_H = \phi$$

$$\bar{O}_M' = O_M'$$

$$\bar{O}_H' = O_H'$$

4.2 Let  $NP_i = NP_i + 1, \forall k \exists (k,i) \in P$ . Set  $t_i = t_i^M x_i + t_i^H(1 - x_i), \forall i \in O$ . Construct the quantified precedence graph and determine  $D_{k,T} \forall k \in O$ .

4.3 Determine the candidate tasks for sequencing:

$$S_M = \{i | NP_i = 0, i \in \bar{O}'_M\}$$

$$S_H = \{j | NP_j = 0, j \in \bar{O}'_H\}$$

If  $S_M = S_H = \phi$ , go to 4.8.

4.4 Compute the earliest start times such that

$$ES_i = \max\{CM, ES_k + t_k, \forall k \in (k,i) \in P\} \forall i \in S_M$$

$$ES_j = \max\{CH, ES_k + t_k, \forall k \in (k,j) \in P\} \forall j \in S_H$$

4.5 Compute bounds for each candidate task:

$$B_i = \max\{CH + LPH, CH + \sum_{k \in \bar{O}'_H} t_k, ES_i + \sum_{k \in \bar{O}'_M} t_k, \max_{k \in \bar{O}'_M} (ES_i + t_i + D_{k,T})\} \forall i \in S_M$$

$$B_j = \max\{CM + LPM, CM + \sum_{k \in \bar{O}'_M} t_k, ES_j + \sum_{k \in \bar{O}'_H} t_k, \max_{k \in \bar{O}'_H} (ES_j + t_j + D_{k,T})\} \forall j \in S_H$$

4.6 Select tasks  $i^*$  and  $j^*$  for sequencing such that

$$B_{i^*} = \min_{i \in S_M} B_i$$

$$B_{j^*} = \min_{j \in S_H} B_j$$

If  $B_{i^*} \geq Z^+$  or  $B_{j^*} \geq Z^+$ , delete  $CP_c$  from CL and return

to step 1.

4.7 Schedule  $i^*$  and  $j^*$ . Set

$$CM = ES_{i^*} + t_{i^*}$$

$$CH = ES_{j^*} + t_{j^*}$$

$$\bar{O}_M^i = \bar{O}_M^i - \{i^*\}$$

$$\bar{O}_H^i = \bar{O}_H^i - \{j^*\}$$

$$NP_k = NP_k - 1, \forall k \exists (i^*, k) \in P \text{ or } (j^*, k) \in P.$$

Go to 4.3.

4.8 Set  $Z = \max[CM, CH]$ .

Step 5: Replace the incumbent solution, if necessary, and prune the candidate list

5.1 If  $Z \geq Z^+$ , delete  $CP_c$  from CL and go to step 1.

5.2 Let  $Z^+ = Z$ ;  $x_i^+ = x_{c,i}$ ,  $\forall i \in O_{MH}$  and  $ES_i^+ = ES_i$ ,  $\forall i \in O$ .

5.3 Delete from the candidate list all problems  $\ell$  such that

$$BND_\ell \geq Z^+.$$

5.4 Go to step 1.

The computational algorithm can be demonstrated by considering a small sample problem.

#### Sample Problem

Consider the problem specified by Table 2. Note that the problem involves nine precedence constraints, and that the time to

Table 2. Data for Sample Problem 1

| Set      | Task Index $i$ | $t_i^M$  | $t_i^H$  | Precedence                                       |
|----------|----------------|----------|----------|--|
| $O_{MH}$ | 1              | 3        | 9        | $(1,2) \in G_2^{MH}; (1,7) \in \overline{G}_2^M$ |
|          | 2              | 5        | 3        | $(2,10) \in \overline{G}_2^H$                    |
|          | 3              | 9        | 9        | --   |
|          | 4              | 5        | 10       | $(4,6) \in G_2^{MH}$                             |
|          | 5              | 10       | 8        | $(5,10) \in G_1^{MH}$                            |
|          | 6              | 1        | 3        | --   |
| $O_M$    | 7              | 15       | $\infty$ | $(7,3) \in G_2^M$                                |
|          | 8              | 10       | $\infty$ | --   |
| $O_H$    | 9              | $\infty$ | 12       | $(9,10) \in G_1^H$                               |
|          | 10             | $\infty$ | 9        | $(10,1) \in G_1^H$                               |
|          | 11             | $\infty$ | 8        | $(11,3) \in G_2^H$                               |

process some task  $i$ ,  $i \in O_M$ , on the human is considered to be infinite.

Likewise, the time to process some task  $i$ ,  $i \in O_H$ , on the machine is also infinite.

The algorithm discussed previously will now be applied to the sample problem.

Step 0: Initialize the problem

| 0.1 | Variable | $a_i$ | $t_i^M/a_i$ | Rank |
|-----|----------|-------|-------------|------|
|     | 1        | 12    | .250        | 1    |
|     | 2        | 8     | .625        | 6    |
|     | 3        | 18    | .500        | 4    |
|     | 4        | 15    | .333        | 3    |
|     | 5        | 18    | .555        | 5    |
|     | 6        | 4     | .250        | 2    |

0.2 Now  $(KP_\phi)$  is given by

min  $z$

$$\text{S.T. } Z \geq 25 + 3x_1 + 5x_2 + 9x_3 + 5x_4 + 10x_5 + x_6$$

$$Z \geq 71 - 9x_1 - 3x_2 - 9x_3 - 10x_4 - 8x_5 - 3x_6$$

$$0 \leq x_i \leq 1 \quad i \in 0_{MH}.$$

For which the two knapsack problems have solution

$$x_1^* = x_4^* = x_6^* = 1; x_3^* = 15/18; x_2^* = x_5^* = 0.$$

$$\text{Thus } \text{BND}_1 = \min\{25+3+5+1+9(15/18);$$

$$71-9-10-3-9(15/18)\}$$

$$= 42.$$

Also let  $\text{IFR}_1 = 3$ .

0.3 Also let  $z^+ = \infty$ . This gives an initial problem for the candidate list:

| Candidate Problem | Variable Values |       |       |       |       |       | Fixed Variables |       |       |       |       |       | Problem Bound |
|-------------------|-----------------|-------|-------|-------|-------|-------|-----------------|-------|-------|-------|-------|-------|---------------|
|                   | $x_1$           | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $F_1$           | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ |               |
| 1                 | 1               | 0     | 0     | 1     | 0     | 1     | 0               | 0     | 0     | 0     | 0     | 0     | 42            |

Step 1: Choose a candidate problem.

1.1  $\text{CL} \neq \emptyset$

1.2  $\text{BND}_1 = \min_{\ell \in \text{CL}} \text{BND}_\ell$

1.3  $0_M^* = \{7, 8\}$

$$0_H^* = \{9, 10, 11\}.$$

Step 2: Choose a separating variable.

2.1  $Q = \{1, 2, 4, 5, 6\}$

$$2.3 \quad r_{IF} = r_1$$

$$r_{IB} = r_2$$

$$2.4 \quad z_{IF} = 45$$

$$z_{IB} = 43$$

$$2.5 \quad z_{IS} = z_{IF} = z_1$$

Step 3: Add a new candidate problem to the candidate list if necessary.

$$3.1 \quad z_1 = 45 \geq z^+ = \infty$$

$$3.2 \quad \ell = 2$$

$$IFR_2 = 5$$

#### Candidate List

| Candidate Problem | Variable Values |       |       |       |       |       | Fixed Variables |       |       |       |       |       | Problem Bound |
|-------------------|-----------------|-------|-------|-------|-------|-------|-----------------|-------|-------|-------|-------|-------|---------------|
|                   | $x_1$           | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $F_1$           | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ |               |
| 1                 | 1               | 0     | 0     | 1     | 0     | 1     | 1               | 0     | 0     | 0     | 0     | 0     | 42            |
| 2                 | 0               | 0     | 1     | 1     | 0     | 1     | 1               | 0     | 0     | 0     | 0     | 0     | 45            |

$$3.3 \quad O'_M = \{1,7,8\}$$

3.4 Go to step 2.

This sequence of steps is repeated until the candidate list consists of the following problems.

#### Candidate List

| Candidate Problem | Variable Values |       |       |       |       |       | Fixed Variables |       |       |       |       |       | Problem Bound |
|-------------------|-----------------|-------|-------|-------|-------|-------|-----------------|-------|-------|-------|-------|-------|---------------|
|                   | $x_1$           | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $F_1$           | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ |               |
| 1                 | 1               | 0     | 0     | 1     | 0     | 1     | 1               | 1     | 1     | 1     | 1     | 1     | 49            |
| 2                 | 0               | 0     | 1     | 1     | 0     | 1     | 1               | 0     | 0     | 0     | 0     | 0     | 45            |
| 3                 | 1               | 0     | 1     | 1     | 0     | 0     | 1               | 0     | 0     | 0     | 0     | 1     | 43            |

| Candidate Problem | Variable Values |       |       |       |       |       | Fixed Variables |       |       |       |       |       | Problem Bound |
|-------------------|-----------------|-------|-------|-------|-------|-------|-----------------|-------|-------|-------|-------|-------|---------------|
|                   | $x_1$           | $x_2$ | $x_3$ | $x_4$ | $x_5$ | $x_6$ | $F_1$           | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ |               |
| 4                 | 1               | 0     | 1     | 0     | 0     | 1     | 1               | 0     | 0     | 1     | 0     | 1     | 45            |
| 5                 | 1               | 0     | 0     | 1     | 1     | 1     | 1               | 0     | 0     | 1     | 1     | 1     | 44            |
| 6                 | 1               | 1     | 0     | 1     | 0     | 1     | 1               | 1     | 0     | 1     | 1     | 1     | 43            |
| 7                 | 1               | 0     | 1     | 1     | 0     | 1     | 1               | 1     | 1     | 1     | 1     | 1     | 43            |

$$O'_M = \{1,4,6,7,8\}$$

$$O'_H = \{2,3,5,9,10,11\}$$

The algorithm will now proceed again in a step-by-step manner.

Step 4: Solve the sequencing problem.

$$4.1 \quad ES_i = NP_i = 0 \quad \forall i \in 0$$

$$CM = CH = 0$$

$$S_M = S_H = \emptyset$$

$$\bar{O}'_M = \{1,4,6,7,8\}$$

$$\bar{O}'_H = \{2,3,5,9,10,11\}$$

$$P = (1,7), (2,10), (4,6), (5,10), (9,10), (10,1), (11,3)$$

| 4.2 | Task | $t_i$ | $NP_i$ | $D_{iT}$ |
|-----|------|-------|--------|----------|
|     | 1    | 3     | 1      | 18       |
|     | 2    | 3     | 0      | 30       |
|     | 3    | 9     | 1      | 9        |
|     | 4    | 5     | 0      | 6        |
|     | 5    | 8     | 0      | 35       |
|     | 6    | 1     | 1      | 1        |
|     | 7    | 15    | 1      | 15       |
|     | 8    | 10    | 0      | 10       |
|     | 9    | 12    | 0      | 39       |
|     | 10   | 9     | 3      | 27       |
|     | 11   | 8     | 0      | 17       |

$$4.3 \quad S_M = \{4, 8\}$$

$$S_H = \{2, 5, 9, 11\}$$

$$4.4 \quad ES_i = 0 \text{ for every } i \in S_M \cup S_H$$

$$4.5 \quad B_4 = \max\{39, 49, 34, 23\} = 49$$

$$B_8 = \max\{39, 49, 34, 28\} = 49$$

$$B_2 = \max\{18, 34, 49, 42\} = 49$$

$$B_5 = \max\{18, 34, 49, 47\} = 49$$

$$B_9 = \max\{18, 34, 49, 47\} = 49$$

$$B_{11} = \max\{18, 34, 49, 47\} = 49$$

$$4.6 \quad B_4 = \min_{i \in S_M} B_i = 49$$

$$B_2 = \min_{j \in S_H} B_j = 49$$

$$4.7 \quad CM = 0 + 5 = 5$$

$$CH = 0 + 3 = 3$$

$$\bar{O}_M = \{1, 6, 7, 8\}$$

$$\bar{O}_H = \{3, 5, 9, 10, 11\}$$

$$NP_6 = 0$$

$$NP_{10} = 2$$

$$4.8 \quad S_M = \{6, 8\}$$

$$S_H = \{5, 9, 11\}$$

The steps 4.3-4.7 are repeated until the following earliest start times are obtained:

$$\begin{array}{ll}
 ES_1 = 32 & \\
 ES_2 = 0 & \\
 ES_3 = 40 & \\
 ES_4 = 0 & CM = 50 \\
 ES_5 = 3 & CH = 49 \\
 ES_6 = 5 & \\
 ES_7 = 35 & \\
 ES_8 = 6 & \\
 ES_9 = 11 & \\
 ES_{10} = 23 & \\
 ES_{11} = 32 &
 \end{array}$$

and the procedure continues.

$$4.3 \quad S_M = S_H = \emptyset, \text{ go to 4.8}$$

$$4.8 \quad z = \max\{50, 49\} = 50$$

Step 5: Replace the incumbent solution if necessary and prune the candidate list.

5.1 Delete  $CP_1$  from CL.

5.2 Let

$$z^+ = 50$$

$$x_1^+ = x_4^+ = x_6^+ = 1, \quad x_2^+ = x_3^+ = x_5^+ = 0$$

$$ES_i^+ = ES_i \quad \text{for every } i \in 0.$$

5.3 No other CP's deleted

5.4 Go to step 1.

Step 1: Choose a new candidate problem.

$$1.1 \quad CL \neq \emptyset$$

$$1.2 \quad \text{Let } BND_7 = \min_{\ell \in CL} BND_\ell$$

$$1.3 \quad O'_M = \{1, 3, 4, 6, 7, 8\}$$

$$O'_H = \{2, 5, 9, 10, 11\}$$

Step 2: Choose a separating variable.

$$2.1 \quad Q = \emptyset; F_{C, IFR_c} = 1, \text{ go to step 4.}$$

The procedure continues in a similar fashion. The results can easily be described in the form of a branch-and-bound tree given by Figure 4.

As shown, the final solution is  $z^+ = 44$ ,  $x_1^+ = x_4^+ + x_5^+ = 1$ ,  $x_2^+ = x_3^+ = x_6^+ = 0$  and

$$\begin{array}{lll} ES_1^+ = 25 & ES_5^+ = 5 & ES_9^+ = 3 \\ ES_2^+ = 0 & ES_6^+ = 24 & ES_{10}^+ = 15 \\ ES_3^+ = 35 & ES_7^+ = 28 & ES_{11}^+ = 27 \\ ES_4^+ = 0 & ES_8^+ = 15 & \end{array}$$

It is of interest to note that this solution is the optimal solution to the problem.

#### Computational Results

The above algorithm was coded in Fortran and several test problems were solved on a Univac 1108 computer.

The problems were generated as follows:

All processing times for the machine are random integers in the interval

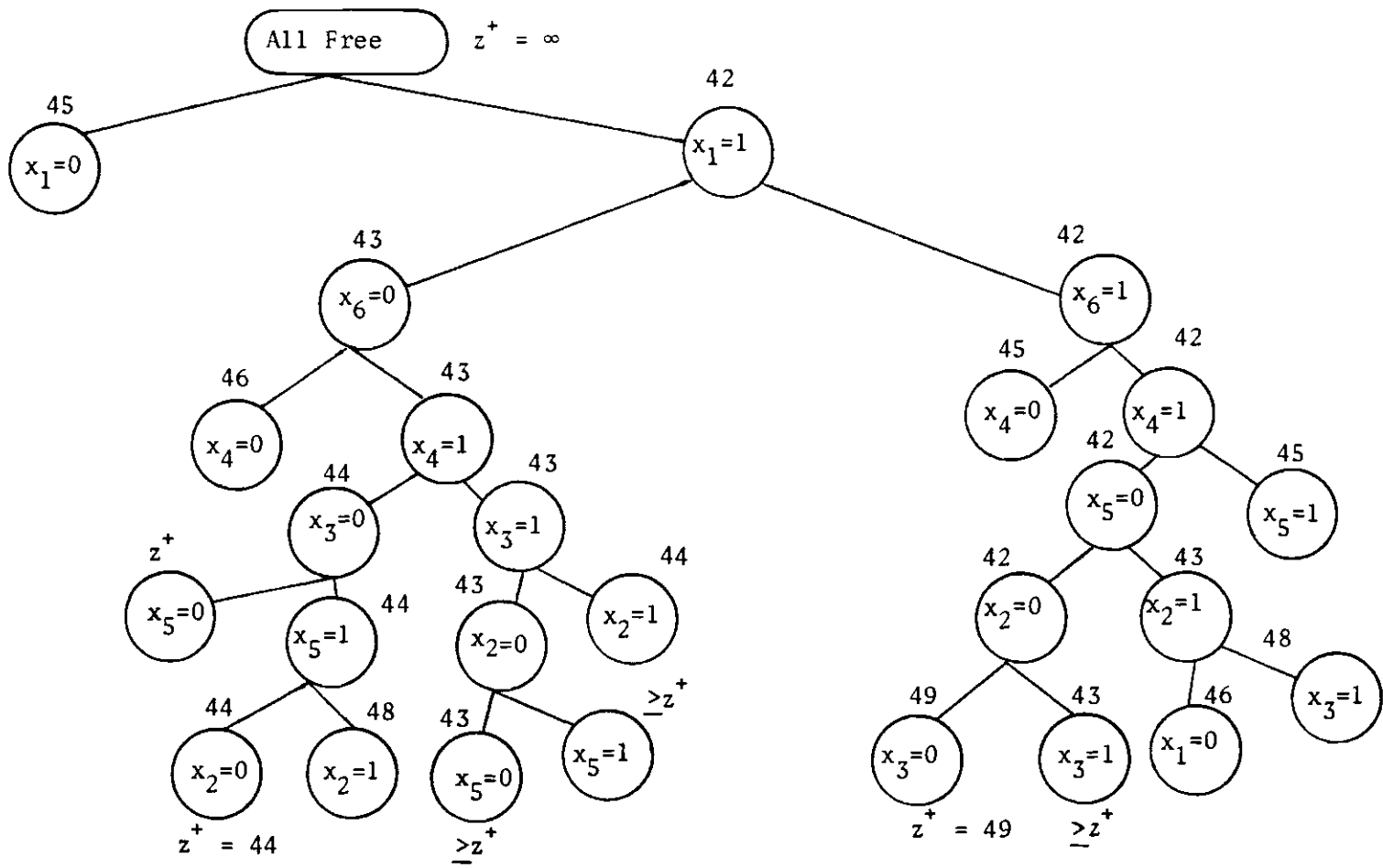


Figure 4. Solution Tree for Sample Problem 1

[1,50], while times for the human were in the interval [1,99]. Constraints were also generated randomly, but in such a manner that precedence between  $O'_M$  and  $O'_H$  occurred about 30 percent of the time.

Several inferences can be drawn from the summary of results given in Table 3. First, the solution times indicate that problems of a realistic size can be solved in a reasonable amount of computer time. The dominate factor in the solution time appears to be the time taken in step four of the algorithm, which is the solution of the sequencing problem. It increases with the total number of tasks, regardless of the number of assignment variables. There are two other factors affecting the time required to sequence tasks. One is the number of tasks which are candidates for sequencing at a particular time. The increase in the number of bounds that must be computed is directly related to the number of these tasks. More important, however, is the number of precedence constraints across the machine and the human. If no constraints exist across the two sets, the sequencing problem is trivial, since bounds are no longer needed. An improvement in the algorithm would be to break any ties in bounds by sequencing the task involved in the most precedence constraints across the two sets.

A second inference that can be made from Table 3 is that storage requirements, which are traditionally high for branch-and-bound procedures, are not prohibitive. By packing the information needed for each candidate problem, only five words of memory were required for each candidate problem. Since there were at most only 53 candidate problems in storage at any one time, this might not be necessary. For large problems, an alternative to bit-packing might be to use off-line

Table 3. Computational Experience for Branch-and-Bound Algorithm

| Problem Number | Problem Size<br>$O_{M1}/O_{M2}/O_H/$<br># Const. | First Solution | Time for First Solution | Last Solution | Time for Last Solution | Time for Termination | Total Number Nodes Explored | Maximum Nodes Stored | Terminal Nodes Explored | Scheduling Problems Started | Scheduling Problems Completed | Time for Scheduling Problems | Ratio of First and Last Solutions |
|----------------|--|----------------|-------------------------|---------------|------------------------|----------------------|-----------------------------|----------------------|-------------------------|-----------------------------|-------------------------------|------------------------------|-----------------------------------|
| 1              | 30/10/10/1                                       | 637            | .076                    | 637           | .076                   | .089                 | 33                          | 29                   | 4                       | 1                           | 1                             | .024                         | 1.00                              |
| 2              | 30.10/10/20                                      | 752            | .468                    | 752           | .468                   | .482                 | 31                          | 29                   | 2                       | 1                           | 1                             | .418                         | 1.00                              |
| 3              | 30/10/10/40                                      | 722            | .438                    | 709           | 1.978                  | 1.989                | 61                          | 29                   | 26                      | 5                           | 3                             | 1.807                        | 0.98                              |
| 4              | 30/20/20/1                                       | 1228           | .097                    | 1228          | .097                   | .109                 | 33                          | 29                   | 4                       | 1                           | 1                             | .046                         | 1.00                              |
| 5              | 30/20/20/20                                      | 1191           | 1.430                   | 1191          | 1.430                  | 1.430                | 31                          | 29                   | 2                       | 1                           | 1                             | 1.560                        | 1.00                              |
| 6              | 30/20/20/40                                      | 1047           | 1.059                   | 1047          | 1.059                  | 1.079                | 37                          | 29                   | 8                       | 1                           | 1                             | 1.004                        | 1.00                              |
| 7              | 40/10/10/1                                       | 825            | .104                    | 819           | .154                   | .167                 | 43                          | 39                   | 4                       | 2                           | 2                             | .072                         | 0.99                              |
| 8              | 40/10/10/30                                      | 866            | .754                    | 844           | 3.917                  | 3.934                | 51                          | 39                   | 12                      | 5                           | 5                             | 3.760                        | 0.97                              |
| 9              | 40/10/10/50                                      | 842            | .761                    | 835           | 1.457                  | 1.483                | 50                          | 39                   | 10                      | 2                           | 2                             | 1.372                        | 0.99                              |
| 10             | 40/20/20/1                                       | 1247           | .133                    | 1236          | .287                   | .300                 | 48                          | 39                   | 8                       | 3                           | 3                             | .182                         | 0.99                              |
| 11             | 40/20/20/30                                      | 1201           | 2.127                   | 1189          | 3.839                  | 3.853                | 43                          | 39                   | 4                       | 2                           | 2                             | 3.716                        | 0.99                              |
| 12             | 40/20/20/50                                      | 1245           | 1.549                   | 1237          | 4.717                  | 4.729                | 45                          | 39                   | 6                       | 3                           | 3                             | 4.600                        | 0.99                              |
| 13             | 50/10/10/1                                       | 964            | 1.410                   | 964           | 1.410                  | 1.424                | 51                          | 49                   | 2                       | 1                           | 1                             | 1.293                        | 1.00                              |
| 14             | 50/10/10/30                                      | 1049           | 1.340                   | 1046          | 2.602                  | 2.618                | 58                          | 49                   | 8                       | 2                           | 2                             | 2.474                        | 0.99                              |
| 15             | 50/10/10/50                                      | 1014           | .958                    | 994           | 4.588                  | 4.602                | 129                         | 53                   | 52                      | 5                           | 4                             | 4.505                        | 0.98                              |
| 16             | 50/20/20/1                                       | 1387           | .172                    | 1377          | .296                   | .315                 | 55                          | 49                   | 6                       | 2                           | 2                             | .192                         | 0.99                              |
| 17             | 50/20/20/30                                      | 1281           | 2.198                   | 1279          | 4.315                  | 4.329                | 53                          | 49                   | 4                       | 2                           | 2                             | 4.197                        | 0.99                              |
| 18             | 50/20/20/50                                      | 1341           | 2.583                   | 1330          | 5.049                  | 5.067                | 57                          | 49                   | 6                       | 2                           | 2                             | 4.860                        | 0.99                              |
| 19             | 50/25/25/1                                       | 1440           | .208                    | 1440          | .208                   | .225                 | 51                          | 49                   | 2                       | 1                           | 1                             | .109                         | 1.00                              |
| 20             | 50/25/25/50                                      | 1551           | 2.584                   | 1537          | 7.615                  | 7.713                | 56                          | 49                   | 6                       | 3                           | 3                             | 7.337                        | 0.99                              |
| 21             | 50/25/25/100                                     | 1509           | 2.599                   | 1498          | 10.039                 | 10.070               | 69                          | 49                   | 14                      | 4                           | 4                             | 9.869                        | 0.99                              |
| 22             | 50/25/25/200                                     | 1533           | 1.657                   | 1533          | 1.657                  | 2.963                | 53                          | 49                   | 4                       | 2                           | 1                             | 2.850                        | 1.00                              |
| 23             | 50/50/50/100                                     | 2539           | 8.491                   | 2539          | 8.491                  | 8.505                | 51                          | 49                   | 2                       | 1                           | 1                             | 8.522                        | 1.00                              |
| 24             | 50/25/5/30                                       | 1200           | 1.791                   | 1181          | 5.256                  | 5.283                | 61                          | 49                   | 10                      | 3                           | 3                             | 5.126                        | 0.98                              |
| 25             | 50/25/5/50                                       | 1104           | 1.553                   | 1103          | 3.002                  | 3.024                | 67                          | 49                   | 14                      | 2                           | 2                             | 2.864                        | 0.99                              |
| 26             | 50/5/25/30                                       | 1272           | 1.750                   | 1272          | 1.750                  | 1.765                | 51                          | 49                   | 2                       | 1                           | 1                             | 1.663                        | 1.00                              |
| 27             | 50/5/25/50                                       | 1276           | 1.606                   | 1276          | 1.626                  | 1.622                | 51                          | 49                   | 2                       | 1                           | 1                             | 1.515                        | 1.00                              |
| 28             | 50/50/25/50                                      | 1792           | 5.831                   | 1784          | 11.407                 | 11.421               | 55                          | 49                   | 6                       | 2                           | 2                             | 11.281                       | 0.99                              |

storage.

The last inference concerns the appeal of backtracking. Since the solution procedure used to solve the sequencing problem cannot guarantee optimality (without some form of backtracking), it may be unwise to backtrack over the assignments. It was originally felt that the possibility of vastly differing constraint sets for different assignments would make a backtrack necessary, but computational experience does not bear this out. Usually the first solution was very close to the final solution, and therefore the time spent to obtain a better solution may not be well spent.

An improvement to the algorithm would be to determine an initial incumbent solution which would, hopefully, make it unnecessary to store some of the candidate problems created. A good initial solution can be obtained by considering the solution to  $(KP_\phi)$ , and computing bounds for each assignment created by setting the fractional valued variable to zero and one. A sequence is then determined for the assignment with the smaller bound, and this sequence is then used as the initial incumbent solution. At this time, the algorithmic procedure described previously can be applied to the problem and candidate problems with bounds no better than this incumbent solution need not be stored, resulting in a savings of computer storage.

A second, and more powerful improvement would be to use penalties, such as those proposed by Tomlin [18], to strengthen the bounds on each candidate problem. This procedure uses the linear programming solution to an integer programming problem (i.e., the solution of the integer programming problem with the integrality restrictions relaxed) as a

bound, and then determines a bound on the change in the objective function brought about by enforcing the integrality restrictions. This bound on the change of the objective function is called a penalty. Penalties are computed from the elements of the optimal simplex tableau.

Since (RUP1) and (RUP2) are relaxations of integer problems, it would be natural to apply the penalty approach to them. However, they are not solved by the simplex method, and thus their tableaux are not readily available. Due to the special structure of these problems, i.e., a single constraint and upper bounds of one on each variable, the optimal tableaux are easily identified. Then, penalties for both (RUP1) and (RUP2) can be computed, their objective function values adjusted accordingly, and the bound for the candidate problem becomes the minimum of these two bounds. The details will be carried out for (RUP2).

Let  $q = |0_{MH}| + 1$ . Now, recall that (RUP2) has the form

$$\begin{aligned}
 z_2 = D - \max \quad & \sum_{i=1}^q c_i x_i \\
 \text{(RUP2)} \quad & \text{S.T.} \quad \sum_{i=1}^q a_i x_i = b \\
 & 0 \leq x_i \leq 1 \quad i = 1, 2, \dots, q-1 \\
 & x_q \geq 0
 \end{aligned}$$

where  $x_q$  is a slack variable, with  $c_q = 0$  and  $a_q = 1$ . Further, let  $x_i^*$  be the optimal solution to (RUP2) found as previously discussed, and let  $x_p^*$  be the fractional valued variable. Using the upper bounded

variable simplex procedure, it is easily seen that the optimal tableau is given as

$$\bar{a}_i = \begin{cases} -a_i/a_p & x_i^* = 1 \\ a_i/a_p & \text{otherwise} \end{cases}$$

$$\bar{c}_i = \begin{cases} c_i + c_p^* \bar{a}_i & x_i^* = 1 \\ -c_i + c_p^* a_i & \text{otherwise} \end{cases}$$

and

$$\bar{b} = x_p^* .$$

Tomlin gives a penalty derived from a Gomory cut as follows:

$$P_G = \min_{i \neq p} \begin{cases} \bar{b} \bar{c}_i / f_i & f_i \leq \bar{b} \\ (1-\bar{b}) \bar{c}_i / (1-f_i) & f_i \geq \bar{b} \end{cases}$$

where  $\bar{a}_i = n_i + f_i$ ,  $f_i \geq 0$  and  $n_i$  is an integer.

Consider the example problem discussed previously. (RUP2) is given as

$$z_2 = 71 - \max 9x_1 + 3x_2 + 9x_3 + 10x_4 + 8x_5 + 3x_6 + 0x_7$$

(RUP2) S.T.

$$12x_1 + 8x_2 + 18x_3 + 15x_4 + 18x_5 + 4x_6 + x_7 = 46$$

$$0 \leq x_i \leq 1 \quad i = 1, 2, \dots, 6$$

with solution  $x_1^* = x_4^* = x_6^* = 1$ ,  $x_3^* = 15/18$ ,  $x_2^* = x_5^* = x_7^* = 0$ , and  $z_2 = 41 \frac{1}{2}$ . The optimal tableau for this problem is

| $x_1$            | $x_2$          | $x_3$ | $x_4$            | $x_5$ | $x_6$           | $x_7$          | b                |
|------------------|----------------|-------|------------------|-------|-----------------|----------------|------------------|
| $-\frac{12}{18}$ | $\frac{8}{18}$ | 1     | $-\frac{15}{18}$ | 1     | $-\frac{4}{18}$ | $\frac{1}{18}$ | $\frac{15}{18}$  |
| 3                | 1              | 0     | $\frac{5}{2}$    | 1     | 1               | $\frac{1}{2}$  | 29 $\frac{1}{2}$ |

which gives

$$f_1 = \frac{6}{18}, f_2 = \frac{8}{18}, f_3 = 0, f_4 = \frac{3}{18}, f_5 = 0, f_6 = \frac{14}{18}, f_7 = \frac{1}{18}$$

and the penalty is given by

$$P_G = \min \left\{ \frac{15}{2}, \frac{15}{8}, \frac{25}{2}, \infty, \frac{15}{14}, \frac{15}{2} \right\} = \frac{15}{14}$$

and

$$z_2' = 71 - \left( 29 \frac{1}{2} - \frac{15}{14} \right) = 42 \frac{6}{14} = 43 .$$

Likewise, an optimal tableau for (RUP1) can be constructed and  $z_1'$  computed. For this example,  $z_1' = 43$  and hence  $z' = 43$  which is greater than  $z = 42$ , the previous bound, and thus a better lower bound.

CHAPTER IV  
 THE UNCONSTRAINED SEQUENCE-DEPENDENT  
 PROCESSING TIME PROBLEM

Recall the unconstrained variation of the general problem which was mentioned at the conclusion of Chapter II. It was clear that the problem was inherently simple since a problem of assignment only, arose. The reason for the disappearance of the sequencing problem in such a relaxation is clear since all processing times until now have been assumed to be sequence independent. However, in the current chapter such processing times are considered to be, in fact, dependent on sequence. Hence, the problem takes on complexity in that attainment of an optimal solution once again involves assignment and sequencing considerations. The measure of performance remains as minimization of the completion time of all operations.

The processing times can best be represented by a matrix,  $\mathcal{J}$ , where each row and column represents a task, and the element  $\tau_{ij}$  represents the processing time of job  $j$  if it follows job  $i$  in the sequence. In addition, there is an extra row added to the matrix which represents an initial set-up time, and a final tear-down time is represented by an added column. These may be considered as dummy tasks which must start and finish the sequence. Furthermore, let  $\tau_{ii} = \infty$ , since a job cannot precede itself, and  $\tau_{ij} = \infty$  if  $i \in 0_M$ ,  $j \in 0_H$ , or  $i \in 0_H$  and  $j \in 0_M$ .

It is of interest to note that the current problem leads to an

enormous solution space. In itself such a characteristic is not surprising since the problem, as is the case throughout the entire research, is combinatorial. Nevertheless, one can compute the total number of solutions to the problem say  $\theta$ , such that

$$\theta = \sum_{i=0}^{|O_{MH}|} (|O_M| + i)! (|O_H| + |O_{MH}| - i)!$$

Of course, if constraints are imposed on the problem, relative to sequencing, the number of feasible solutions reduces to something less than  $\theta$ . The fact remains, that for even modest increases in problem size, there results an overwhelming number of solutions.

Prior to examining a potential algorithmic development to solve the problem at hand, consider a subtlety involving a similar well-known problem.

The unconstrained sequence-dependent processing time problem is very similar to the parallel processor problem with two machines. However, there are at least three differences. First, the parallel processor problem traditionally assumes identical processors, which is not the case here. Secondly, the parallel processor problem usually maintains that processors are initially unloaded, which again differs from the problem discussed here. The major difference, however, is in the measure of performance used. As in the previous chapter, this treatment seeks to minimize maximum completion time over operations in  $O'_M$  and  $O'_H$ , while the parallel processor problem utilizes maximization of total processing time of all activities, or an analogous function such as total set-up time. The two measures are not

necessarily equivalent, as can be demonstrated by considering the following proposition.

Proposition 1: Minimizing project duration is not equivalent to minimizing total processing time.

Note that project duration is defined as the maximum completion time of all tasks and total processing time is the sum of  $\tau_{ij}$  for  $(i,j)$ 's in the sequence.

The proposition will be demonstrated by a counterexample. Let  $O_{MH} = \{1\}$ ,  $O_M = \emptyset$ , and  $O_H = \{2\}$ . Also let S and T represent the initial and final operations of any sequence. The processing times are  $\tau_{11} = \infty$ ,  $\tau_{12} = 2$ ,  $\tau_{IT} = 2$ ,  $\tau_{21} = 5$ ,  $\tau_{22} = \infty$ ,  $\tau_{2T} = 3$ ,  $\tau_{S1} = 4$ ,  $\tau_{S2} = 2$ ,  $\tau_{ST} = \infty$ . Total enumeration gives the following assignments and sequences:

| Machine | Human   | Project Completion Time | Total Processing Time |
|---------|---------|-------------------------|-----------------------|
| S-1-T   | S-2-T   | 6                       | 11                    |
| -       | S-1-2-T | 9                       | 9                     |
| -       | S-2-1-T | 8                       | 8                     |

Thus, the solution which minimizes project completion time is not the solution which minimizes total processing time.

### Basic Concepts

Due to the inconsistency of the measures of performance considered in both problems, it would seem that traditional algorithms for the two processor problem would not be useful in the solution of the problem of the current chapter. Hence, the composition of the

remainder of this chapter will center around the development of a suitable procedure.

Prior to discussing the development of an algorithm, however, it is of interest to note that the graph-theoretic implication of the current problem differs from that of the so-called general problem given earlier. Whereas the general problem dictated a mixed type graph,  $G(N;C \cup D)$ , the problem presently under consideration is completely disjunctive initially and is denoted by  $G(N;D)$ . The disjunctive nature of the problem arises from the lack of constraints either among operations in  $O_M$  and  $O_H$  (initially as well as in the final solution) as between such operations. Of course, a final solution will have all operations fixed in sequence which reflects a synthesis from a disjunctive state to one totally conjunctive, given by some  $G(N;A)$ .

As before, one approach to solving the problem would be a decomposition scheme such that assignment and sequencing decisions are made in a nearly independent manner. Specifically, the method used is to enumerate (either explicitly or implicitly) each assignment and determine the optimal sequence for each such assignment. Inherent in such a procedure is the requirement that bad assignments be recognized quickly, so that they not be explicitly enumerated.

Bounds on the solution may be obtained by summing the minimum processing times for each task to be sequenced. If one has a solution that is as good as the computed bounds, for a given assignment, there is no need to explore that assignment further. Also, if in building a sequence, it can be shown that the sequence can produce no better solution than the best solution found thus far, there is no need to

further pursue this assignment.

The enumeration scheme used herein is a variation of one proposed by Geoffrion [5]. The major difference is that only assignments that have all tasks in  $O_{MH}$  assigned to either the human or the machine need be investigated. The mechanism for enumerating these assignments is as follows. Let  $S$  be an ordered vector composed of either positive or negative, and underlined or non-underlined indices of  $O_{MH}$ . If  $+i$  is in  $S$ , then task  $i$  is assigned to the machine, and if  $-i$  is in  $S$ , then  $J$  is assigned to the human. If an element of  $S$ , say  $i_0$  is underlined, then all feasible assignments, with the elements of  $S$  to the left of  $i_0$  taking on their present values, have been enumerated.

The scheme for generating the assignments, referred to as a backtrack procedure, can now be given: start with any arbitrary assignment, placing the indices in  $S$  with the proper signs. Initially, let  $S$  be void of underlined elements. The first assignment is then explored, and  $S$  is changed by complementing the last element of  $S$  and underlining it. In general, to change  $S$ , complement the right-most non-underlined element of  $S$ , and delete all underlines to its right. The procedure terminates when all elements of  $S$  are underlined.

To show that this procedure is finite, it is both necessary and sufficient to show that the sequence of  $S$  vectors generated is non-redundant, since there are exactly  $2^n$  such  $S$  vectors. Consider the following proposition.

**Proposition 2:** The backtrack procedure presented above generates a non-redundant sequence of assignments.

Proof: To show that the sequence  $S_j$  is non-redundant, note that  $S_1$  cannot be redundant, and assume that  $S_1, S_2, \dots, S_k$  is non-redundant.  $S_{k+1}$  is formed by complementing the right-most non-underlined element of  $S_k$ , say  $i_0$ , and deleting all underlines to its right. Therefore,  $S_{k+1}$  differs from  $S_k$  since  $i_0$  is different. Hence,  $S_1, S_2, \dots, S_k, S_{k+1}$  is non-redundant, and therefore by induction the sequence  $S_j$  is non-redundant.

In order to obtain the optimal solution, no assignment may be overlooked which could give a better solution than the incumbent. Since every assignment generated by the backtrack procedure is checked for a possible better solution, and not explored only if it cannot produce a better solution, it remains only to show that the backtrack procedure generates all  $2^n$  possible assignments. The following proposition does this.

Proposition 3: The backtrack procedure generates all possible assignments.

Proof: Since the procedure terminates only when all elements of  $S$  are underlined, it will suffice to show that all possible assignments to the right of the underlined variable have been generated. Since the first element underlined is the last element of  $S$ , it is obvious that, for all elements to the left fixed, all possible assignments have been generated, since there are only two possible assignments. Now assume that this is true for  $S_1, S_2, \dots, S_k$ . Suppose  $i$  is the left-most underlined element of  $S_k$ . Then all elements to its right are underlined, and hence all possible assignments for the variables to its right, and the fixed assignment of variables to its left have

been explored. Thus by induction after all elements of S have been underlined, all possible assignments have been generated.

The sequencing problems can be formulated as a variation of the well-known traveling salesman problem. The technique used to solve these problems is similar to that of Little et al. [10] and of Ashour et al. [2]. Although not presented here, the procedure is given in Appendix A. If the sequencing problems are solved exactly, the proposed algorithm will provide exact solutions for the complete problem. However, in the current work, only first pass solutions to the sequencing problems are obtained and hence, exact solutions cannot be specified. It is worth noting that the method employed in the sequencing solution procedure has been shown to be efficient in the first pass mode [2] and is, in addition, easily convertible to an exact method should optimality be more critical than computational effort.

Bounds on the solution to the operation sequencing problems can be computed in a rather traditional manner using the reduction coefficients. Consider, specifically, the set of operations to be performed by the human and define the following computations.

The reduction coefficients used to compute bounds are found as follows:

For the human, set

$$R_i^H = \min_{j \in O_H \cup O_{MH}} \tau_{ij} \quad \forall i \in O_H \cup O_{MH}$$

$$\tau'_{ij} = \tau_{ij} - R_i^H \quad \forall i, j \in O_H \cup O_{MH}$$

$$C_i^H = \min_{j \in O_H \cup O_{MH}} \tau_{ij}^H \quad \forall i \in O_H \cup O_{MH}$$

and

$$HB_i = R_i^H + C_i^H \quad \forall i \in O_H \cup O_{MH},$$

where  $R_i^H$  represents the minimum processing time for task  $i$  preceding any other task, while  $C_i^H$  represents the minimum processing time for task  $i$  following any other task. Similar reduction coefficients, denoted by  $MB_i$ , can be found for tasks to be done by the machine.

The concepts given above can now be formalized by the following algorithmic statement.

#### Computational Algorithm

Step 0: Initialize the procedure.

0.1 Compute  $MB_i$  and  $HB_i \quad \forall i \in O$ .

0.2 Determine an initial assignment which defines  $S$ ,  $O_M'$  and  $O_H'$ .

0.3 Set  $z^+ = \infty$ .

Step 1: Compute bounds for the sequencing problems for the machine

$(P_M')$  and the human  $(P_H')$ .

1.1 Let

$$z_M = \sum_{i \in O_M'} MB_i$$

$$z_H = \sum_{i \in O_H'} HB_i$$

1.2 If  $z_M \geq z^+$  or  $z_H \geq z^+$  go to step 4.

Step 2: Solve  $P'_M$  and  $P'_H$ .

2.1 Let  $V'_M$  and  $V'_H$  be the solutions to  $P'_M$  and  $P'_H$  respectively.

2.2 If  $V'_M \geq z^+$  or  $V'_H \geq z^+$  go to step 4.

Step 3: Replace the incumbent solution.

3.1 Let  $z^+ = \max\{V'_M, V'_H\}$

3.2 Let

$$x_i^+ = \begin{cases} 1 & \text{if } i \in O'_M \\ 0 & \text{otherwise} \end{cases}$$

Step 4: Perform a backbrack operation.

4.1 Find the right-most non-underlined element of  $S$ , say  $i_0$ .

If no such element exists, terminate, the incumbent is optimal.

4.2 Replace  $i_0$  by its complement, underline it, and delete all underlines to its right. Update  $O'_M$  and  $O'_H$ .

4.3 Go to step 1.

The following section will demonstrate the algorithm by applying it to a small sample problem.

#### Sample Problem

Consider a problem consisting of eight tasks. These tasks are given such that  $O_{MH} = \{1,2,3,4\}$ ,  $O_M = \{5,6,9\}$  and  $O_H = \{7,8,9\}$ . Note that task nine is a dummy task which represents both the beginning (set-up) and final (tear-down) operations. The times for sequencing

these tasks are given in Table 4.

Table 4. Data for Sample Problem 2

| Task | 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        |
|------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 1    | $\infty$ | 5        | 3        | 4        | 4        | 6        | 5        | 8        | 4        |
| 2    | 5        | $\infty$ | 4        | 5        | 5        | 2        | 8        | 7        | 2        |
| 3    | 1        | 5        | $\infty$ | 6        | 1        | 6        | 8        | 5        | 6        |
| 4    | 6        | 6        | 8        | $\infty$ | 3        | 2        | 1        | 4        | 4        |
| 5    | 4        | 3        | 4        | 2        | $\infty$ | 3        | $\infty$ | $\infty$ | 7        |
| 6    | 6        | 1        | 6        | 6        | 5        | $\infty$ | $\infty$ | $\infty$ | 4        |
| 7    | 5        | 8        | 9        | 4        | $\infty$ | $\infty$ | $\infty$ | 6        | 5        |
| 8    | 8        | 6        | 9        | 1        | $\infty$ | $\infty$ | 2        | $\infty$ | 3        |
| 9    | 4        | 6        | 8        | 3        | 2        | 3        | 3        | 4        | $\infty$ |

The step by step procedure is:

Step 0: Initialize the procedure.

$$0.1 \quad MB_1 = 3, MB_2 = 2, MB_3 = 1, MB_4 = 2$$

$$MB_5 = 2, MB_6 = 1, MB_9 = 2$$

$$HB_1 = 3, HB_2 = 4, HB_3 = 1, HB_4 = 1$$

$$HB_7 = 4, HB_8 = 2, HB_9 = 3$$

$$0.2 \quad S = \{1, -2, 3, -4\}$$

$$O_M^1 = \{1, 3, 5, 6, 9\}$$

$$O_H^1 = \{2, 4, 7, 8, 9\}$$

$$0.3 \quad z^+ = \infty$$

Step 1: Compute bounds for  $P'_M$  and  $P'_H$ .

$$1.1 \quad z_M = 3+1+2+1+2 = 9$$

$$z_H = 4+1+4+2+3 = 14$$

$$1.2 \quad z_M \not\leq z^+ \text{ and } z_H \not\leq z^+$$

Step 2: Solve  $P'_M$  and  $P'_H$ .

$$2.1 \quad V'_M = 15$$

$$V'_H = 16$$

$$2.2 \quad V'_M \not\leq z^+ \text{ and } V'_H \not\leq z^+$$

Step 3: Replace the incumbent solution.

$$3.1 \quad z^+ = \max\{15, 16\} = 16$$

$$3.2 \quad x_1^+ = x_3^+ = 1; \quad x_2^+ = x_4^+ = 0$$

Step 4: Perform a backtrack operation.

$$4.1 \quad i_0 = -4$$

$$4.2 \quad S = \{1, -2, 3, 4\}$$

$$O'_M = \{1, 3, 4, 5, 6, 9\}$$

$$O'_H = \{2, 7, 8, 9\}$$

4.3 Go to step 1.

Step 1: Compute bounds for  $P'_M$  and  $P'_H$ .

$$1.1 \quad z_M = 3+1+2+2+1+2 = 11$$

$$z_H = 4+4+2+3 = 13$$

$$1.2 \quad z_M \not\leq z^+ \text{ and } z_H \not\leq z^+$$

Step 2: Solve  $P'_M$  and  $P'_H$ .

$$2.1 \quad V'_M = 18$$

$$V_H' = 16$$

2.2  $V_M' \geq z^+$ , go to step 4.

Step 4: Perform a backtrack operation.

$$4.1 \quad i_0 = 3$$

$$4.2 \quad S = 1, -2, -\underline{3}, 4$$

$$O_M' = \{1, 4, 5, 6, 9\}$$

$$O_H' = \{2, 3, 7, 8, 9\}$$

4.3 Go to step 1.

The procedure continues in this manner, with the results given in Table 5. The optimal solution is

$$z^+ = 14$$

$$x_1^+ + x_2^+ = x_3^+ = 1, \quad x_4^+ = 0$$

and the sequence for the human is 9-8-4-7-9 with completion time 11 and for the machine 9-1-3-5-6-2-9 with completion time 14. This solution is depicted graphically in Figure 5.

#### Computational Results

The above algorithm was coded in Fortran and several problems run on a Univac 1108 computer. The results have been summarized, and are presented in Table 6. All test problems were randomly generated; processing times were uniformly distributed on the interval [3,10] for tasks belonging to the sets  $O_{MH}$  and  $O_M$ , and on the interval [5,10] for tasks in  $O_H$ .

Table 5. Summary of Solution Procedure Applied to Sample Problem 2

| Iteration | S  | Outcome                     |
|-----------|--|-----------------------------|
| 3         | {1, -2, - <u>3</u> , 4}                            | $V_M^i$ or $V_H^i \geq z^+$ |
| 4         | {1, -2, - <u>3</u> , - <u>4</u> }                  | $V_M^i$ or $V_H^i \geq z^+$ |
| 5         | {1, - <u>2</u> , -3, -4}                           | $V_M^i$ or $V_H^i \geq z^+$ |
| 6         | {1, <u>2</u> , -3, <u>4</u> }                      | $V_M^i$ or $V_H^i \geq z^+$ |
| 7         | {1, <u>2</u> , 3, 4}                               | $z^+ = 15$                  |
| 8         | {1, <u>2</u> , <u>3</u> , - <u>4</u> }             | $z^+ = 14$                  |
| 9         | {- <u>1</u> , 2, 3, -4}                            | $V_M^i$ or $V_H^i \geq z^+$ |
| 10        | {- <u>1</u> , 2, 3, <u>4</u> }                     | $V_M^i$ or $V_H^i \geq z^+$ |
| 11        | {- <u>1</u> , 2, - <u>3</u> , 4}                   | $V_M^i$ or $V_H^i \geq z^+$ |
| 12        | {- <u>1</u> , 2, - <u>3</u> , - <u>4</u> }         | $z_M$ or $z_H \geq z^+$     |
| 13        | {- <u>1</u> , - <u>2</u> , -3, -4}                 | $z_M$ or $z_H \geq z^+$     |
| 14        | {- <u>1</u> , - <u>2</u> , - <u>3</u> , <u>4</u> } | $z_M$ or $z_H \geq z^+$     |
| 15        | {- <u>1</u> , - <u>2</u> , <u>3</u> , 4}           | $z_M$ or $z_H \geq z^+$     |
| 16        | {- <u>1</u> , - <u>2</u> , <u>3</u> , - <u>4</u> } | $z_M$ or $z_H \geq z^+$     |

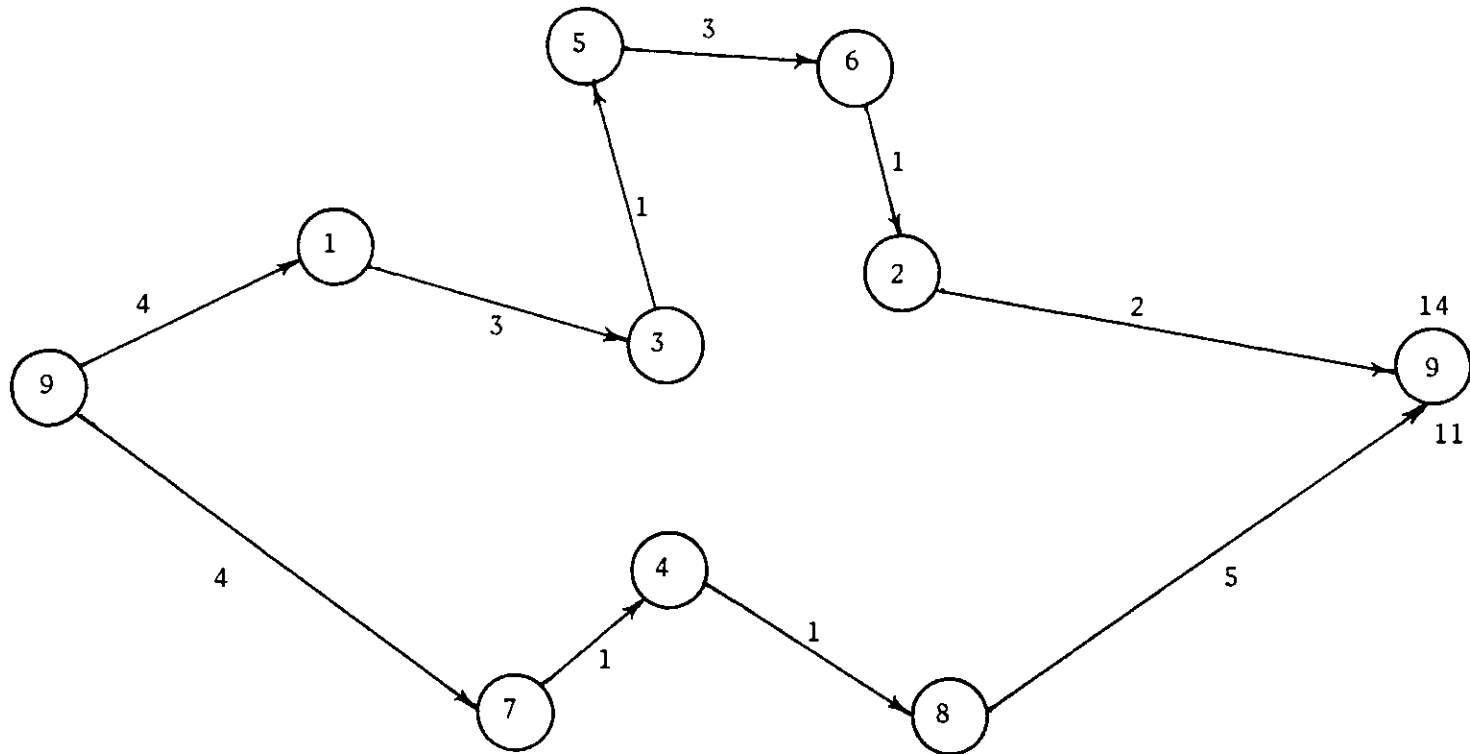


Figure 5. Graph Theoretic Depiction of the Solution to Sample Problem 2

Table 6. Computational Results for the Sequence-Dependent Set-up Time Problem

| Problem Number | Problem Size<br>0 <sub>MH</sub> /0 <sub>M</sub> /0 <sub>H</sub> | Sequencing Problems |           | Solution Value |       | Ratio of Initial & Final Solutions | Solution Time<br>CPU Seconds |
|----------------|---|---------------------|-----------|----------------|-------|------------------------------------|------------------------------|
|                |   | Started             | Completed | Initial        | Final |                                    |                              |
| 1              | 2/2/2   | 6                   | 4         | 24             | 22    | .916                               | .137                         |
| 2              | 2/2/2   | 6                   | 2         | 12             | 12    | 1.000                              | .073                         |
| 3              | 4/2/2   | -                   | -         | 16             | 13    | .812                               | .156                         |
| 4              | 4/2/14  | -                   | -         | 43             | 43    | 1.000                              | 3.747                        |
| 5              | 5/5/5   | -                   | -         | 35             | 28    | .800                               | .945                         |
| 6              | 6/4/18  | 64                  | 2         | 35             | 35    | 1.000                              | 31.308                       |
| 7              | 6/6/6   | 66                  | 14        | 41             | 28    | .683                               | 2.014                        |
| 8              | 6/18/4  | 14                  | 6         | 22             | 21    | .954                               | 33.200                       |
| 9              | 7/14/8  | 131                 | 8         | 39             | 36    | .923                               | 41.501                       |
| 10             | 7/7/7   | 72                  | 11        | 31             | 25    | .806                               | 3.091                        |
| 11             | 8/4/10  | 263                 | 14        | 45             | 35    | .778                               | 17.757                       |
| 12             | 8/4/18  | 234                 | 6         | 44             | 40    | .909                               | 61.274                       |
| 13             | 8/8/8   | 166                 | 12        | 32             | 24    | .750                               | 6.727                        |
| 14             | 8/10/4  | 453                 | 200       | 24             | 22    | .916                               | 103.958                      |
| 15             | 8/18/4  | 79                  | 13        | 25             | 22    | .880                               | 44.768                       |
| 16             | 9/1/1   | --                  | --        | 22             | 11    | .500                               | 1.296                        |
| 17             | 9/4/18  | 454                 | 10        | 41             | 35    | .854                               | 133.411                      |
| 18             | 9/9/9   | 300                 | 10        | 35             | 27    | .771                               | 8.314                        |
| 19             | 9/9/9   | 292                 | 44        | 28             | 33    | .820                               | 45.520                       |
| 20             | 9/9/9   | --                  | --        | 31             | 28    | .903                               | 62.756                       |
| 21             | 11/11/11  | 1,791               | 25        | 41             | 32    | .780                               | 146.045                      |
| 22*            | 11/11/11  | --                  | --        | 33             | 26    | .788                               | > 180.000                    |
| 23             | 15/1/1  | 10,819              | 104       | 20             | 15    | .750                               | 113.242                      |
| 24*            | 19/1/1  | --                  | --        | 18             | 15    | .833                               | > 180.000                    |

From Table 6, it can be seen that the solution times for the test problems reflect a high variance. For the most part, this is due to the problem dependency of the algorithm. Some problems have many solutions of approximately the same project duration, which causes the algorithm to investigate many assignments. Problem number 14 in Table 6 is of this type. Others, such as problem number 13, reflect a large difference between the final and other solutions, and, as such, are solved rather quickly.

The first solution obtained was found by balancing the reduction coefficients. That is, assignments were made so that the sum of reduction coefficients for tasks in  $O_M^1$  was close to the sum of reduction coefficients or tasks in  $O_H^1$ . The ratio of the final solution obtained to the initial solution gives an indication of the desirability of backtracking. For the 22 problems of Table 6 which terminated, the average quality was 84 percent. Again, it should be noted that the final solutions of Table 6 are not necessarily optimal, since the sequencing problems are not solved exactly.

A second piece of information contained in Table 6 is the number of sequencing problems attempted, along with the number of sequencing problems which were completed. A pair of sequencing problems were considered only if the bounds computed in Step 1 of the algorithm were better than the incumbent solution. However, once these problems were considered, the computations need only be continued as long as there is a possibility that a solution better than the incumbent can be obtained. Thus, many sequencing problems that are considered only need a few iterations of the sequencing algorithm

before it is obvious that the particular assignment can yield no better solution than the incumbent.

All solution times are in CPU seconds. Problems were arbitrarily terminated at the end of 180 CPU seconds. Those problems which were terminated before completion are marked by an asterisk in Table 6.

## CHAPTER V

## TASK ASSIGNMENT AND SEQUENCING OF CREWS

A generalization of the problem presented in the previous chapter (as well as an extension of the general problem) would be to consider more than a single man and a single machine in the system. In this case, there may be several men and/or machines to be considered. This problem will be discussed in the context of a crew of men who must complete a project. As before, the project is comprised of a set of tasks  $O$ . Certain of these tasks must be done by a particular crew member, while other tasks form a pool of tasks which could conceivably be done by any crew member. Let  $O_k$  be the set of tasks which must be done by crew member  $k$ , and  $O_{MH}$  be the pool of tasks which are to be assigned to the crew members. The number of crew members will be denoted by  $\ell$ .

If no precedence constraints are allowed, and processing times are assumed to be sequence independent, a formulation similar to the formulation at the end of Chapter II can be stated. The problem is

$$\begin{aligned}
 \text{min.} \quad & z \\
 \text{s.t.} \quad & z \geq \sum_{i \in O_k} \tau_{ik} + \sum_{i \in O_{MH}} \tau_{ik} x_{ik} \quad k = 1, 2, \dots, \ell \\
 & \sum_{i \in O_{MH}} x_{ik} = 1 \quad k = 1, 2, \dots, \ell \\
 & x_{ik} \in \{0, 1\} \quad i \in O_{MH}, k = 1, 2, \dots, \ell
 \end{aligned}$$

where

$$x_{ik} = \begin{cases} 1 & \text{if task } i \text{ is assigned to crew member } k \\ 0 & \text{otherwise} \end{cases}$$

If, as in the previous chapter, no precedence constraints are allowed, and processing times are considered to be sequence dependent, this formulation is no longer valid. The measure of performance will be to minimize project duration, which is the maximum completion time of all tasks.

Analogous to the definitions of Chapter III, define  $O'_k$  to be the union of  $O_k$  and that sub-set of tasks of  $O_{MH}$  which are assigned to crew member  $k$ . Let  $P'_k$  be the sequencing problem for crew member  $k$ , with tasks  $O'_k$ . Also, let  $\mathcal{T}'_k$  be the processing times of  $O'_k$ .

For  $\ell = 2$ , the above problem is identical to the problem discussed in Chapter IV. However, the addition of more men and/or machines makes the problem increasingly complex. Previously, there were  $2^{|O_{MH}|}$  possible assignments to consider; for the crew served problem with  $\ell$  crew members, there are  $\ell^{|O_{MH}|}$  possible assignments. For each possible assignment, there are  $\ell$  sequencing problems to consider, each of which has  $|O'_k|!$  solutions.

### Basic Concepts

As in the previous chapters, the basic approach will be one of decomposing the assignment and sequencing decisions. An algorithm similar to the algorithm of Chapter III is developed, since it is felt that the branch-and-bound approach will give more flexibility than the enumeration scheme of Chapter IV. Unassigned, or free variables will

be denoted by the set  $O'_F$ . If  $O'_F = \emptyset$ , then all tasks are assigned, and the assignment is said to be feasible. A relaxation of the problem will be to allow tasks to be unassigned or free, that is, to consider an infeasible assignment.

Let  $z'_k$  denote the solution to  $P'_k$ . Also let  $L'_k$  denote the largest  $\tau_{ij}$ , for all  $(i,j)$  in the sequence determined for  $P'_k$ . Then a bound on the completion time of crew member  $k$ , if an additional task  $i$  is assigned to crew member  $k$ , is given by

$$B_{ik} = z'_k - L'_k + \min_{j \in O'_k \cup O'_k} \tau_{ij} + \min_{j \in O'_F \cup O'_k} \tau_{ji} .$$

These bounds are used to determine an assignment to fix so that two new candidate problems are created (and added to the candidate list if necessary) and are also used as lower bounds on the candidate problems. If task  $i$  is assigned to crew member  $k$  (denoted by  $(i,k)$ ) in one of the candidate problems, then the bound on that candidate problem is  $B_{ik}$ . The bound on the candidate problem which prohibits this assignment (denoted by  $(\overline{i,k})$ ) is given by

$$\text{BND} = \min_{\substack{j=1,2,\dots,\ell \\ j \neq k}} B_{ij}$$

The separating assignment is chosen such that the bound on  $(\overline{i,k})$  is as large as possible.

The mechanics of the procedure are identical to that of Chapter III, with one exception. Since a solution could be obtained quickly using the procedure of Chapter III, no effort was made to find a

starting solution. In this chapter, the difficulty of getting a feasible assignment makes it desirable to obtain a starting solution. This is done by starting the procedure in such a manner that the next candidate problem chosen is the one which, at the previous iteration, made an assignment, rather than the one which prohibited an assignment. After an initial solution is found in this manner, candidate problems are chosen in the lowest bound first order.

As in Chapter IV, the sequencing problems are formulated in a manner similar to traveling salesman problems. If these are solved exactly, the procedure used to solve the total problem gives the optimal solution, which can be shown by the following proposition.

Proposition 4: The branch-and-bound scheme outlined above is finite and terminates with the optimal solution.

Proof: Each task has  $\ell$  possible assignments, and thus, there are only  $\ell^{I_{MH}}$  feasible assignments. Each of the feasible assignments will appear as one of the  $\sum_{i=1}^{I_{MH}} \ell^i$  candidate problems. Since each candidate problem is compared with the incumbent solution, as soon as the optimal solution is explored, it becomes the incumbent. After this, no candidate problem will change the incumbent. Candidate problems not explicitly explored cannot have solutions better than the incumbent, since the lower bounds on the solution was no better than the incumbent. Hence, the incumbent at termination is optimal.

A formal statement of the algorithm can now be given.

Computational Algorithm

Step 0: Initialize the procedure.

- 0.1 Set  $0'_k = 0_k$   $k = 1, 2, \dots, \ell$ .
- 0.2 Solve  $P'_k$ ,  $k = 1, 2, \dots, \ell$ . Denote the solution values by  $z'_k$ , and the maximum link in each sequence by  $L'_k$ . Set  $z^+ = \infty$ .
- 0.3 Go to step 5.

Step 1: Choose a candidate problem.

- 1.1 If  $CL = \emptyset$ , stop, the incumbent is optimal.
- 1.2 Let CP be the element of CL with the lowest bound.
- 1.3 If CP prohibits an assignment, go to step 5.
- 1.4 Update  $0'_F$ .

Step 2: Solve a sequencing problem.

- 2.1 Let task  $i$  be assigned to processor  $k$  by candidate problem CP.
- 2.2 Add  $i$  to  $0'_k$ .
- 2.3 Solve  $P'_k$ . Denote its solution by  $z'_k$  and the maximum link by  $L'_k$ .

Step 3: Attempt to fathom CP.

- 3.1 If  $z'_k < z^+$ , go to step 4.
- 3.2 Delete CP from CL and go to step 1.

Step 4: Check for feasibility.

- 4.1 If  $0'_F = \emptyset$ , go to step 6.

Step 5: Separate the candidate problem.

- 5.1 Compute bounds  $B_{ik}$  for each free assignment.
- 5.2 Choose a separating assignment  $(i,k)$ .
- 5.3 Determine bounds for both new candidate problems and add them to the candidate list if necessary.
- 5.4 Go to step 1.

Step 6: Replace the incumbent solution.

- 6.1 Let  $z^+ = \max_{k=1,2,\dots,\ell} z'_k$   
 $0_k^+ = 0'_k$  ,  $k = 1,2,\dots,\ell$ .
- 6.2 Delete all candidate problems from the candidate list with bounds no better than  $z^+$ .
- 6.3 Go to step 1.

The algorithm will now be applied to a small sample problem.

#### Sample Problem

Consider a problem involving three crew members and eight tasks such that  $0_{MH} = \{1,2,3\}$ ,  $0_1^1 = \{4,5,9\}$ ,  $0_2^1 = \{6,9\}$ , and  $0_3^1 = \{7,8,9\}$ , where task nine represents both start-up and tear-down operations. The processing times for this problem are given in Table 7. A step by step application of the algorithm to this problem follows.

Step 0: Initialize the procedure.

- 0.1  $0_1^1 = \{4,5,9\}$ ,  $0_2^1 = \{6,9\}$ ,  $0_3^1 = \{7,8,9\}$ .
- 0.2  $z_1^1 = 11$              $L_1^1 = 5$   
 $z_2^1 = 13$              $L_2^1 = 9$

Table 7. Data for Sample Problem 3

| Task | 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        |
|------|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| 1    | $\infty$ | 9        | 5        | 9        | 8        | 6        | 8        | 3        | 5        |
| 2    | 3        | $\infty$ | 4        | 3        | 6        | 8        | 3        | 6        | 4        |
| 3    | 5        | 4        | $\infty$ | 6        | 6        | 7        | 4        | 5        | 2        |
| 4    | 4        | 5        | 5        | $\infty$ | 3        | $\infty$ | $\infty$ | $\infty$ | 4        |
| 5    | 3        | 2        | 4        | 5        | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 3        |
| 6    | 3        | 3        | 5        | $\infty$ | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 4        |
| 7    | 3        | 5        | 3        | $\infty$ | $\infty$ | $\infty$ | $\infty$ | 6        | 6        |
| 8    | 3        | 6        | 2        | $\infty$ | $\infty$ | $\infty$ | 5        | $\infty$ | 3        |
| 9    | 5        | 5        | 2        | 5        | 8        | 9        | 4        | 7        | $\infty$ |

$$z_3^1 = 13 \quad L_3^1 = 6$$

$$z^+ = \infty$$

0.3 Go to step 5.

Step 5: Separate the candidate problem.

$$5.1 \quad B_{11} = 11 - 5 + 5 + 3 = 14$$

$$B_{12} = 12$$

$$B_{13} = 13$$

$$B_{21} = 11$$

$$B_{31} = 10$$

$$B_{22} = 10$$

$$B_{32} = 8$$

$$B_{23} = 14$$

$$B_{33} = 11$$

5.2 (1,2)

5.3  $CL_1(1,2)$        $BND_1 = 13$

$CL_2(\overline{1,2})$        $BND_2 = 13$

5.4 Go to step 1.

Step 1: Choose a candidate problem.

1.1  $CL \neq \emptyset$

1.2  $CP = CL_1$

1.3 Does not prohibit an assignment

1.4  $O'_F = \{2,3\}$

Step 2: Solve a sequencing problem.

2.1 (1,2)

2.2  $O'_2 = \{1,6,9\}$

2.3  $z'_2 = 15, L'_2 = 6$

Step 3: Attempt to fathom CP.

3.1  $z'_2 < z^+ = \infty$ , go to step 4.

Step 4: Check for feasibility.

4.1  $O'_F = \{2,3\} \neq \emptyset$

Step 5: Separate the candidate problem.

5.1  $B_{21} = 11$                        $B_{31} = 10$

$B_{22} = 15$                        $B_{32} = 13$

$B_{23} = 14$                        $B_{33} = 11$

5.2 (2,1)

5.3  $CL_1$  (1,2); (2,1)      BND = 15

$CL_3$  (1,2);  $\overline{(2,1)}$       BND = 15

5.4 Go to step 1.

Step 1: Choose a candidate problem.

- 1.1  $CL \neq \emptyset$
- 1.2  $CP = CL_2$
- 1.3 Go to step 5.

Step 5: Separate the candidate problem.

- 5.1  $B_{11} = 14$        $B_{21} = 11$        $B_{31} = 10$   
 $B_{13} = 13$        $B_{22} = 10$        $B_{32} = 8$   
 $B_{23} = 14$        $B_{33} = 11$
- 5.2 (1,3)
- 5.3  $CL_2 = (1,3)$       BND = 13  
 $CL_4 = (1,1)$       BND = 14
- 5.4 Go to step 1.

Step 1: Choose a candidate problem.

- 1.1  $CL \neq \emptyset$
- 1.2  $CP = CL_2$
- 1.3 Does not prohibit an assignment.
- 1.4  $O_F^1 = \{2,3\}$

Step 2: Solve a sequencing problem

- 2.1 (1,3)
- 2.2  $O_3^1 = \{1,7,8,9\}$
- 2.3  $z_3^1 = 13$        $L_3^1 = 4$

Step 3: Attempt to fathom CP.

- 3.1  $z_3^1 < z^+ = \infty$ , go to step 4.

Step 4: Check for feasibility.

$$4.1 \quad 0'_F \neq \emptyset.$$

Step 5: Separate the candidate problem.

$$5.1 \quad B_{21} = 12 \quad B_{31} = 10$$

$$B_{22} = 11 \quad B_{32} = 8$$

$$B_{23} = 16 \quad B_{33} = 13$$

$$5.2 \quad (2,2)$$

$$5.3 \quad CL_2 = (1,3); (2,2) \quad BND = 13$$

$$CL_5 = (1,3); (\overline{2,2}) \quad BND = 13$$

5.4 Go to step 1.

The procedure continues in this manner until the final solution is obtained. The steps of the procedure are summarized in Figure 6. The final solution found is  $z^+ = 14$ ,  $0_1^+ = \{2,4,5,9\}$ ,  $0_2^+ = \{3,6,9\}$ , and  $0_3^+ = \{1,7,8,9\}$ .

#### Computational Results

The above algorithm was coded in Fortran and several problems run on a Univac 1108. These problems were randomly generated, but the processing times were generated such that a triangular property held. That is, for every  $i, k \in 0$ ,  $\tau_{ik} \leq \tau_{ij} + \tau_{jk}$  for all  $j \in 0$ . This was necessary to insure that the addition of a task to a sequencing problem would not result in a better solution. A rectangle is  $R^2$ , defined by XMIN, XMAX, YMIN, and YMAX was formed, and for each task in  $0$ , a point,  $(x_i, y_i)$  in this rectangle was randomly generated. The processing times are then given by

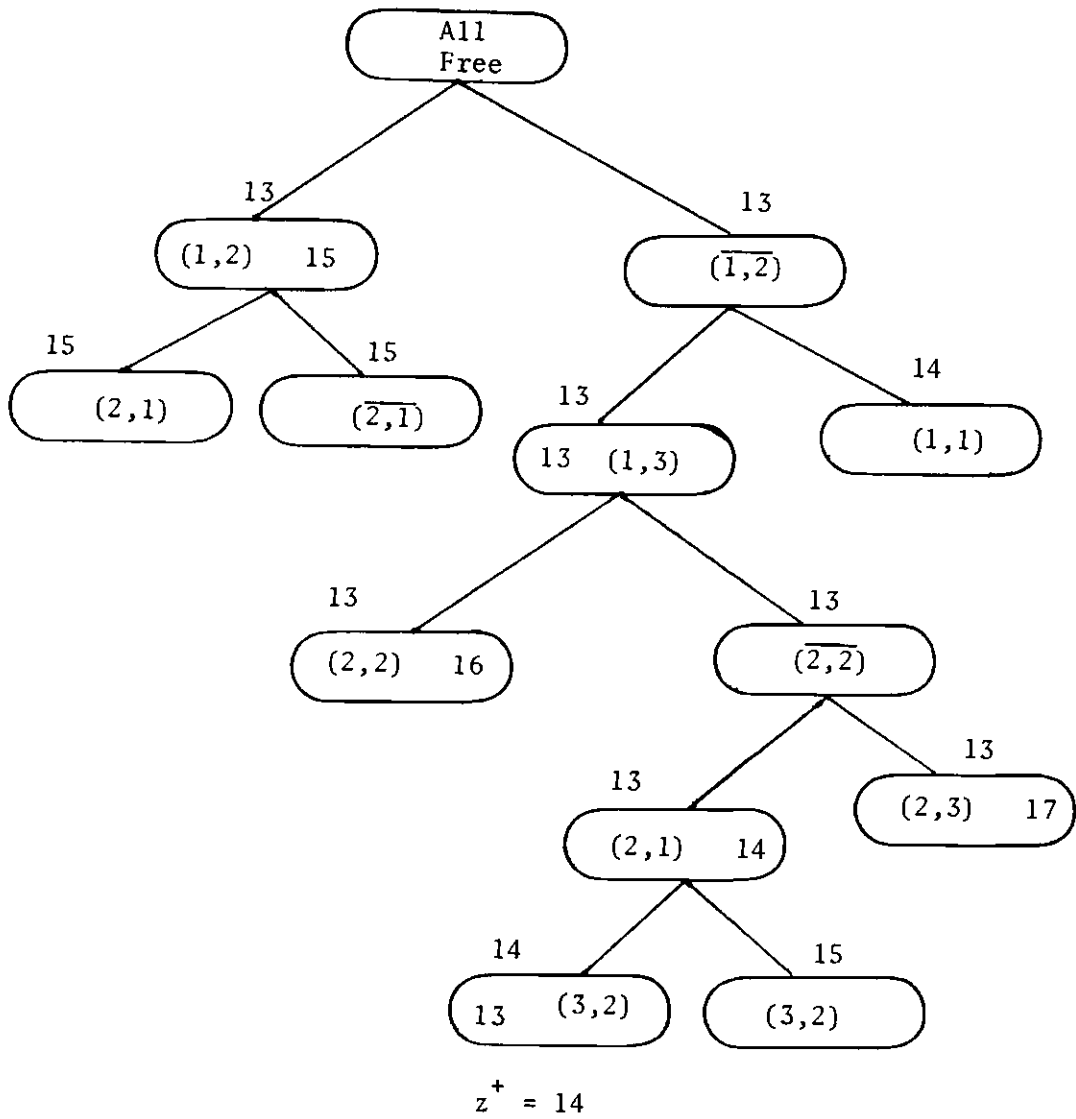


Figure 6. Solution Tree for Sample Problem 3

$$\tau_{ij} = \begin{cases} (y_i - YMIN) + (y_j - YMIN) + (x_j - x_i) & \text{if } x_i < x_j \\ (YMAX - y_i) + (YMAX - y_j) + (x_i - x_j) & \text{if } x_i \geq x_j \end{cases}$$

This method will generate processing times which have the triangular property, but are not necessarily symmetric.

A summary of the computational experience is given in Table 8. The primary drawback of this algorithm appears to be the number of nodes explored. Even though the ratio of the number of nodes explored to the total number of nodes may be small, the large number of possible nodes for even moderate sized problems causes the algorithm to exceed computer memory. Provisions were made to store 5,000 nodes in the program, and, as exemplified by problems 12, 29, 32, 37, and 39 in Table 8, such an allotment may not be sufficient. Increasing core available, or packing words, might alleviate the problem, but a better solution would be to develop tighter bounds, which would exclude many nodes from being stored.

Solution times again have high variance. It is clear that one crew member can dominate the solution if he must do many tasks. A problem of this type, such as problems 5, 20, 23, and 39, will solve very quickly. Problems in which the crew members all have about the same initial loading are much more difficult to solve.

The first solution was obtained quickly, usually in less than one second of CPU time. The average ratio of the first solution to the last solution was 96 percent. This may be misleading in that some problems (marked by an asterisk) were not run to termination, either

Table 8. Computational Results for Task Assignment and Sequencing of Crews

| Problem Number | Problem Size<br>$0_{M1}/0_1/0_2/\dots/0_g$ | Solution<br>Initial/Final | Ratio | Sequencing                   | Candidate                 | Time                        |
|----------------|--|---------------------------|-------|------------------------------|---------------------------|-----------------------------|
|                |  |                           |       | Problems<br>Started/Finished | Problems<br>Maximum/Total | (CPU secs)<br>Initial/Final |
| 1              | 2/2/2                                      | 72/70                     | .972  | 6/6                          | 4/7                       | .174/.201                   |
| 2              | 2/6/2                                      | 118/118                   | 1.000 | 4/4                          | 4/4                       | .148/.154                   |
| 3              | 2/10/10                                    | 188/184                   | .978  | 8/7                          | 7/9                       | .369/.738                   |
| 4              | 5/2/2                                      | 84/82                     | .976  | 26/17                        | 16/32                     | .145/.238                   |
| 5              | 5/6/2                                      | 142/114                   | .803  | 13/13                        | 16/17                     | .208/.315                   |
| 6              | 5/10/10                                    | 160/158                   | .988  | 40/25                        | 32/55                     | .714/2.468                  |
| 7              | 10/2/2                                     | 134/132                   | .985  | 321/180                      | 241/473                   | .314/5.190                  |
| 8              | 10/6/2                                     | 148/148                   | 1.000 | 470/245                      | 278/675                   | .437/13.397                 |
| 9              | 10/10/2                                    | 158/158                   | 1.000 | 20/20                        | 20/20                     | .645/.650                   |
| 10             | 10/10/10                                   | 194/186                   | .959  | 497/352                      | 642/920                   | 1.682/6.072                 |
| 11             | 15/2/2                                     | 142/114                   | .804  | 440/438                      | 1028/1031                 | .583/13.662                 |
| 12*            | 15/6/2                                     | 166/162                   | --    | --/--                        | > 5000                    | .873/160.568                |
| 13             | 2/2/2/2                                    | 88/88                     | 1.000 | 4/4                          | 4/4                       | .147/.153                   |
| 14             | 2/10/10/10                                 | 178/174                   | .978  | 10/6                         | 9/14                      | .549/.963                   |
| 15             | 2/6/2/2                                    | 136/136                   | 1.000 | 4/4                          | 4/4                       | .139/.145                   |
| 16             | 5/2/2/2                                    | 104/104                   | 1.000 | 40/19                        | 15/59                     | .149/.366                   |
| 17             | 5/6/2/2                                    | 110/110                   | 1.000 | 10/10                        | 10/10                     | .168/.173                   |
| 18             | 5/10/10/10                                 | 192/184                   | .959  | 15/15                        | 19/20                     | .870/1.370                  |
| 19             | 10/2/2/2                                   | 112/94                    | .839  | 178/93                       | 110/302                   | .362/2.183                  |
| 20             | 10/10/2/2                                  | 164/164                   | 1.000 | 20/20                        | 20.20                     | .380/.385                   |
| 21             | 10/10/10/10                                | 204/196                   | .961  | 634/454                      | 1211/1508                 | 1.622/71.035                |
| 22             | 15/2/2/2                                   | 118/98                    | .831  | 2193/1562                    | 3971/5086                 | .577/58.845                 |
| 23             | 15/10/2/2                                  | 148/140                   | .946  | 61/51                        | 75/101                    | .653/1.980                  |
| 24             | 2/2/2/2                                    | 90/90                     | 1.000 | 6/6                          | 4/4                       | .132/.138                   |
| 25             | 2/10/10/10                                 | 172/172                   | 1.000 | 6/6                          | 4/4                       | .472/.477                   |
| 26             | 5/2/2/2/2                                  | 94/90                     | .959  | 14/14                        | 19/20                     | .211/.290                   |
| 27             | 5/5/2/2/2                                  | 104/98                    | .942  | 191/81                       | 138/366                   | .272/1.916                  |
| 28             | 5/10/10/10/10                              | 176/176                   | 1.000 | 9/9                          | 10/10                     | .873/.878                   |
| 29*            | 10/2/2/2/2                                 | 104/--                    | --    | --                           | > 5000                    | .279/31.848                 |
| 30             | 10/5/2/2/2                                 | 108/106                   | .982  | 1554/531                     | 758/2722                  | .293/18.560                 |
| 31             | 10/10/10/10/10                             | 190/178                   | .936  | 404/179                      | 270/808                   | 1.291/31.281                |
| 32*            | 15/2/2/2/2                                 | 116/--                    | --    | --                           | > 5000                    | .470/54.916                 |
| 33             | 15/7/2/2/2                                 | 128/124                   | .969  | 1536/440                     | 263/2.694                 | .643/27.565                 |
| 34             | 2/2/2/2/2/2                                | 88/88                     | 1.000 | 8/6                          | 5/11                      | .137/.158                   |
| 35             | 5/2/2/2/2/2                                | 94/94                     | 1.000 | 10/10                        | 10/10                     | .173/.178                   |
| 36             | 5/10/10/10/10/10                           | 160/160                   | 1.000 | 10/10                        | 10/10                     | .870/.874                   |
| 37*            | 10/2/2/2/2/2                               | 110/--                    | --    | --                           | > 5000                    | .282/46.046                 |
| 38             | 10/10/10/10/10/10                          | 168/158                   | .941  | 263/148                      | 317/612                   | 1.455/23.768                |
| 39             | 15/2/2/2/2/2                               | 114/--                    | --    | --                           | > 5000                    | .492/63.591                 |
| 40             | 15/2/2/2/2/2                               | 112/112                   | 1.000 | 30/30                        | 30/30                     | .680/686                    |

because of core limitations or time limitations. Also there is a large number of smaller problems which have a ratio of 100 percent.

Problems 1-12 are the same type problems that were solved in Chapter IV. By comparing the two sets of problems, one can determine that the time to solution is about the same for similar size problems using the two algorithms.

## CHAPTER VI

### CONCLUSIONS AND RECOMMENDATIONS

#### Conclusions

A new problem, the man-machine task assignment and sequencing problem, has been defined in this research. A mathematical statement of this problem has been given and validated. A branch-and-bound algorithm has been developed to solve this problem, and the algorithm has been coded in FORTRAN. Computational experience was obtained which indicates that good solutions to large problems are attainable.

A variation on the above problem, formed by ignoring all precedence constraints and assuming that processing times are sequence dependent, was then explored. An implicit enumeration algorithm to solve this problem was developed and coded in FORTRAN. Computational results indicate that good initial solutions are obtained, but that for medium and large problems, attempts to improve this solution may require large amounts of computer time.

A branch-and-bound algorithm was then developed for a third problem, the crew served problem. This algorithm was also coded in FORTRAN and computational experience obtained. As in the previous case, the initial solution is good, and attempts to obtain a better solution required much computer time.

All of the algorithms mentioned above decompose the assignment and sequencing decisions. Each algorithm is exact if the sequencing sub-problems are solved optimally. However, the major effort expended

lies in solving the sequencing sub-problems, hence, the algorithms are used in a heuristic mode. Further, in many cases, the quality of the first solution is high enough to warrant the elimination of backtracking over assignments.

### Recommendations

Several extensions/improvements to this work have evolved from the present investigation.

Different Measures of Performance. Several alternative measures of performance could be implemented. This would necessitate changing the bounds computed in all three algorithms. However, most other measures of performance would probably result in bounds that were easier to compute, and also closer in value to the actual solution.

Some of the alternative measures might be to minimize total processing time, total cost, or to maximize operator disgression. Total processing time could be handled in much the same way as schedule time, except that bounds would be computed by adding the machine and human processing times, instead of taking the maximum. The same would hold true of costs, unless there was a cost associated with the assignment, in which case the problem becomes similar to a fixed charge problem. To maximize operator disgression, a measure called overlap [9] is introduced. Again this could be handled in a straight-forward manner.

Constraints. There are many constraints other than precedence constraints which could be considered in the formulations presented herein. For example, there may be environmental considerations for

the human that would limit the amount of time he could work without resting. Constraints of this type would be introduced into the sequencing procedures of the algorithm.

Bounds. The bounds developed in each algorithm are extremely important in determining the time required for solving the problem. Thus, good bounds are critical. Tighter bounds for the crew served problem may be obtained by somehow allocating unassigned tasks to crew members on a pro-rata basis, as is done in Chapter III. Other bounding techniques, such as solving linear programming problems, should also be explored.

Recursive Solutions to Sequencing Problems. In all algorithms discussed in this research, the decomposition approach dictates that several sequencing sub-problems be solved. Since the sequencing problems are often only slight perturbations of a previously solved sequencing problem, it would appear that being able to solve one sequencing problem using the solution to a similar problem would be helpful computationally.

APPENDIX A

SOLUTION OF THE SEQUENCING PROBLEMS

## APPENDIX A

## SOLUTION OF THE SEQUENCING PROBLEMS

The sequencing problems discussed in Chapters IV and V are solved by a variation of the algorithm of Ashour et al. [2]. If  $n$  is the number of tasks to be sequenced, and  $T(i,j)$  is the processing time associated with sequencing task  $i$  before task  $j$ , the algorithm is as follows:

Step 1. Initialize the sequence assignments.

1.1 Set the level index  $L = 1$

1.2 Set all the sequence assignments

$$A(i,j) = 0, \quad i,j = 1,2,\dots,n$$

Step 2: Compute the regrets.

2.1 Reduce the matrix such that for each row  $I$ ,

$$T(I,j) := T(I,J) - \min_j(T(I,j)), \quad I = 1,2,\dots,n$$

and then for each column  $J$ ,

$$T(I,J) := T(I,J) - \min_i(T(i,J)), \quad J = 1,2,\dots,n$$

2.2 Mark the cells which have zeros in the reduced matrix by

$(\hat{I}, \hat{J})$  and evaluate the associated regrets such that

$$R(\hat{I}, \hat{J}) = \min_{\substack{j \\ j \neq \hat{J}}} (T(\hat{I}, j)) + \min_{\substack{i \\ i \neq \hat{I}}} (T(i, \hat{J}))$$

Step 3. Select the link  $(I^*, J^*)$  for possible inclusion in the final sequence such that

$$R(I^*, J^*) = \max_{(\hat{I}, \hat{J})} (R(\hat{I}, \hat{J}))$$

3.1 If a tie does not exist, go to step 4.

3.2 If a tie exists and  $L < n - 1$ , for each tied link  $(I^*, J^*)$  set temporarily  $T(J^*, I^*) = \infty$  and compute the total reduction such that

$$D(I^*, J^*) = \sum_{\substack{i \\ i \neq I^*}} [\min_{\substack{j \\ j \neq J^*}} \{T(i, j)\}] + \sum_{\substack{j \\ j \neq J^*}} [\min_{\substack{i \\ i \neq I^*}} \{T(i, j)\}]$$

and select the link which has the minimum reduction.

3.3 If a tie exists and  $L = n - 1$ , break the tie by any particular rule.

Step 4. Check the level index.

4.1 If  $L < n$ , include the selected link  $(I^*, J^*)$  in the final sequence by setting

$$A(I^*, J^*) = 1$$

Exclude the link  $(k, \ell)$  which would join the ends of the longest connected path involving  $(I^*, J^*)$  and the previously selected links by setting

$$T(k, \ell) = \infty$$

Delete row  $I^*$  and column  $J^*$  by setting

$$T(I^*, j) = T(i, J^*) = \infty, \quad i, j = 1, 2, \dots, n$$

and set

$$T(J^*, I^*) = \infty$$

Set  $L = L + 1$  and go to step 2.

4.2 If  $L = n$ , include the selected link  $(I^*, J^*)$  in the final sequence by setting

$$A(I^*, J^*) = 1$$

and go to step 5.

Step 5. Evaluate the final sequence.

5.1 Set the sequence of links having

$$A(I^*, J^*) = 1$$

5.2 Find the total schedule time such that

$$z = \sum_{(I^*, J^*)} T(I^*, J^*)$$

The solution will result in a complete sequence which may or may not be optimal. The procedure can incorporate a backtracking scheme to insure optimality, but is not used in this research.

## APPENDIX B

COMPUTER CODE FOR THE GENERAL PROBLEM

```

COMPILER (FLD=ABS)
IMPLICIT INTEGER (A-Z)
DIMENSION NDI(4000,4),NBD(4000)
DEFINE NODINF(I,J)=FLD(0,36,NDI(I,J))
DEFINE NDBND(I)=FLD(0,36,NBD(I))
DEFINE XX(I,J)=FLD(J-36*(J/37)-1,1,NODINF(I,(J/37)+1))
DEFINE FX(I,J)=FLD(J-36*(J/37)-1,1,NODINF(I,(J/37)+3))
DEFINE NEXT(I)=FLD(18,18,NODINF(I,2))
DEFINE RHS(I)=FLD(18,18,NODINF(I,4))
DEFINE BND(I)=FLD(0,30,NDBND(I))
DEFINE NINV(I)=FLD(30,6,NDBND(I))
COMMON MH,MST,MEND,HST,HEND,T, TM,TH, TIME,X,NPR, FROM,TO, TYPE,
$ SUMM,SUMH,ST,LARGE,SCHDTM,ZPLUS,TIMSQ,NOSCD,NOFIN
DIMENSION TM(51),TH(51),TIME(101),X(101),FROM(200),TO(200),
$ TYPE(200),ST(101)
REAL AMIN,HREAL,MREAL,XREAL,RAT
DIMENSION A(51),RANK(51),F(51),RAT(51)
DIMENSION XPLUS(51)
1 FORMAT( )
  READ(5,1) NREAD
  READ(NREAD,1)MH,M,H
  MST=MH+1
  MEND=MH+M
  HST=MEND+1
  HEND=MEND+H
  T=HEND+1
  READ(NREAD,2)(TM(I),TH(I),I=1,MH)
  READ(NREAD,2)(TIME(I),I=MST,MEND)
  READ(NREAD,2)(TIME(I),I=HST,HEND)
2 FORMAT(20(1X,I2))
  DO 10 I=1,HEND
10 X(I)=0
  DO 20 I=1,MST,MEND
20 X(I)=1
  READ(NREAD,1)NPR
  DO 30 I=1,NPR
30 READ(NREAD,1) FROM(I),TO(I),TYPE(I)
  LARGE=9999999
  TIMSQ=0
  TERND=0
  MOSNOD=0
  TNOD=0
  NOFIN=0
  NOSCD=0
  TIMIN=ITIME(TM1, TM2)
C COMPUTE THE RATIO AND RANK OF THE KNAPSACK VARIABLES
  MH1=MH-1
  MH2=MH-2
  ZPLUS=LARGE
  ISTOP=0
  DO 110 I=1,MH
  A(I)=TM(I)+TH(I)
110 RAT(I)=FLOAT(TM(I))/FLOAT(A(I))
  DO 130 I=1,MH

```

```

    AMIN=2.0
    DO 120 J=1,MH
    IF(RAT(J).GE.AMIN) GO TO 120
    IMIN=J
    AMIN=RAT(J)
120  CONTINUE
    RANK(I)=IMIN
    RAT(IMIN)=2.1
130  CONTINUE
    MTOT=0
    DO 140 I=MST,MEND
140  MTOT=MTOT+TIME(I)
    HTOT=0
    DO 150 I=1,MH
150  HTOT=HTOT+TH(I)
    DO 160 I=MST,MEND
160  HTOT=HTOT+TIME(I)
    BBAR=HTOT-MTOT
    HLM=BBAR
    MREAL=MTOT
    HREAL=HTOT
    DO 190 I=1,MH
    J=RANK(I)
    IF(BBAR-A(J))170,180,180
170  XREAL=FLOAT(BBAR)/FLOAT(A(J))
    MREAL=MREAL+XREAL*FLOAT(TM(J))
    HREAL=HREAL-XREAL*FLOAT(TH(J))
    IFR=J
    GO TO 200
180  MREAL=MREAL+TM(J)
    HREAL=HREAL-TH(J)
    BBAR=BBAR-A(J)
190  CONTINUE
200  NINV(1)=IFR
    DO 210 I=1,MH
    J=RANK(I)
    IF(J.EQ.IFR) GO TO 215
    X(J)=1
    XX(1,J)=1
210  CONTINUE
215  BD=MREAL+.9999
    BD2=HREAL+.9999
    IF(BD2.LT.BD) BD=BD2
    BND(1)=BD
    RHS(1)=HTOT-MTOT
    NNOD=4000
    K=NNOD-1
    DO 220 I=2,K
220  NEXT(I)=I+1
    NXTAV=2
    NXTNOD=0
    NEXT(1)=0
    LSTNOD=1
    NXTBND=LARGE

```

```

LSTBND=BD
CURND=1
GO TO 1015
C CHOOSE A CANDIDATE PROBLEM AND EXPLORE IR
1000 IF(ISTOP.GT.0.OR.NXTNOD.EQ.0) GO TO 6000
    CURND=NXTNOD
    NXTNOD=NEXT(CURND)
    IF(NXTNOD.NE.0) NXTBND=BND(NXTNOD)
    NFIX=0
    DO 1010 I=1,MH
    X(I)=XX(CURND,I)
    F(I)=FX(CURND,I)
    IF(F(I).GT.0) NFIX=NFIX+1
1010 CONTINUE
1015 NRHS=RHS(CURND)
    IF(NFIX.EQ.MH1) GO TO 4000
    IFR=NINV(CURND)
    MCOST=MTOT
    MFIXD=MCOST
    H COST=HTOT
    DO 1020 I=1,MH
    IF(X(I).EQ.0) GO TO 1020
    MCOST=MCOST+TM(I)
    H COST=H COST-TH(I)
    IF(F(I).GT.0) MFIXD=MFIXD+TM(I)
1020 CONTINUE
    WIS=-1
    AIFR=HLM
    DO 1030 I=1,MH
1030 IF(X(I).GT.0) AIFR=AIFR-A(I)
    DO 1040 I=1,MH
    IF(IFR.NE.RANK(I)) GO TO 1040
    RIFR=I
    GO TO 1045
1040 CONTINUE
1045 HFIXD=HTOT
    DO 1050 I=1,MH
1050 IF(F(I).EQ.0.OR.X(I).GT.0) HFIXD=HFIXD-TH(I)
    CURBND=BND(CURND)
C CHOOSE A SEPARATING VARIABLE
2000 IF(MH2.NE.NFIX) GO TO 2009
    IF(WIS.GE.0) GO TO 2005
    WIS=1
    BBND=1
    GO TO 2009
2005 IF(WIS.EQ.0) BBND=-1
    IF(WIS.GT.0) FBND=-1
    GO TO 2600
2009 NS=0
    IF(WIS.GE.0) NS=1
    IF(WIS.EQ.0) GO TO 2025
    IFRONT=0
    DO 2010 I=1,MH
    J=RANK(I)

```

```

        IF(F(J).GT.0) GO TO 2010
        IF(J.EQ.IFR) GO TO 2010
        IFRONT=J
        GO TO 2020
2010 CONTINUE
2020 IF(WIS.GT.0) GO TO 2040
2025 IBACK=0
        DO 2030 I=MH,1,-1
            J=RANK(I)
            IF(F(J).GT.0) GO TO 2030
            IF(J.EQ.IFR) GO TO 2030
            IBACK=J
            GO TO 2040
2030 CONTINUE
2040 IS=IFRONT
        IF(WIS.EQ.0) IS=IBACK
2070 IF(X(IS).GT.0) GO TO 2400
C NOW SOLVE AN X FROM 0 TO 1 PROBLEM
        MREAL=MCOST+TM(IS)
        HREAL=HCOST-TH(IS)
        PBND=MFIXD+TM(IS)
        BBAR=A(IS)-AIFR
        IF(PBND.LT.HFIXD) PBND=HFIXD
        IF(BBAR.GE.0) GO TO 2100
        XREAL=-FLOAT(BBAR)/FLOAT(A(IFR))
        MREAL=MREAL+XREAL*FLOAT(TM(IFR))+.9999
        HREAL=HREAL-XREAL*FLOAT(TH(IFR))+.9999
        MINF=IFR
        GO TO 2140
2100 II=RIFR-1
        DO 2130 I=II,1,-1
            J=RANK(I)
            IF(F(J).GT.0) GO TO 2130
            IF(BBAR-A(J)) 2110,2120,2120
2110 XREAL=FLOAT(BBAR)/FLOAT(A(J))
            MREAL=MREAL-XREAL*FLOAT(TM(J))+.9999
            HREAL=HREAL+XREAL*FLOAT(TH(J))+.9999
            MINF=J
            GO TO 2140
2120 MREAL=MREAL-TM(J)
            HREAL=HREAL+TH(J)
            BBAR=BBAR-A(J)
2130 CONTINUE
2140 I=MREAL
            J=HREAL
            IF(I.GT.J) I=J
            IF(PBND.LT.I) PBND=I
            GO TO 2450
2400 MREAL=MCOST-TM(IS)
            PBND=MFIXD
            HREAL=HCOST+TH(IS)
            BBAR=A(IS)+AIFR
            ITMH=HFIXD+TH(IS)
            IF(PBND.LT.ITMH) PBND=ITMH

```

```

DO 2430 I=RIFR,MH
  J=RANK(I)
  IF(F(J).GT.0) GO TO 2430
  IF(BBAR-A(J)) 2410,2420,2420
2410 XREAL=FLOAT(BBAR)/FLOAT(A(J))
  MREAL=MREAL+XREAL*FLOAT(TM(J))+.9999
  HREAL=HREAL-XREAL*FLOAT(TH(J))+.9999
  MINF=J
  GO TO 2440
2420 MREAL=MREAL+TM(J)
  HREAL=HREAL-TH(J)
  BBAR=BBAR-A(J)
2430 CONTINUE
2440 I=MREAL
  J=HREAL
  IF(I.GT.J) I=J
  IF(I.GT.PBND) PBND=I
2450 IF(IS.EQ.IBACK) GO TO 2460
  FBND=PBND
  FFRC=MINF
  GO TO 2470
2460 BBND=PBND
  BFRC=MINF
2470 NS=NS+1
  IF(NS.GT.1) GO TO 2600
  IS=IBACK
  GO TO 2070
2600 IF(BBND.GT.FBND) GO TO 2610
  ISEP=IFRONT
  NEWBND=FBND
  NEWFRC=FFRC
  WIS=1
  GO TO 2620
2610 ISEP=IBACK
  NEWBND=BBND
  NEWFRC=BFRC
  WIS=0
2620 NEWRHS=NRHS
  IF(X(ISEP).EQ.0) NEWRHS=NRHS-A(ISEP)
C ADD A NODE TO THE CANDIDATE LIST IF NECESSARY
3000 F(ISEP)=1
  FX(CURND,ISEP)=1
  NFIX=1+NFIX
  IF(X(ISEP).EQ.0) HFIXD=HFIXD+TH(ISEP)
  IF(X(ISEP).EQ.0) GO TO 3005
  NRHS=NRHS-A(ISEP)
  RHS(CURND)=NRHS
  MFIXD=MFIXD+TM(ISEP)
3005 IF(NEWBND.LT.ZPLUS) GO TO 3010
  IF(NFIX.EQ.MH1) GO TO 4000
  GO TO 2000
3010 NEWNOD=NXTAV
  TNOD=TNOD+1
  IF(NEWNOD.GT.MOSNOD) MOSNOD=NEWNOD

```

```

NXTAV=NEXT(NEWNOD)
DO 3020 I=1,4
3020 NODINF(NEWNOD,I)=NODINF(CURND,I)
IF(IFR.EQ.NEWFRC) GO TO 3029
IF(X(ISEP).EQ.0) GO TO 3025
II=NEWFRC-1
DO 3022 I=IFR,II
IF(F(I).GT.0) GO TO 3022
XX(NEWNOD,I)=1
3022 CONTINUE
GO TO 3029
3025 II=NEWFRC+J
DO 3027 I=II,IFR
IF(F(I).GT.0) GO TO 3027
XX(NEWNOD,I)=0
3027 CONTINUE
3029 IF(X(ISEP).GT.0) XX(NEWNOD,ISEP)=0
IF(X(ISEP).EQ.0) XX(NEWNOD,ISEP)=1
BND(NEWNOD)=NEWBND
NINV(NEWNOD)=NEWFRC
RHS(NEWNOD)=NEWRHS
C NOW FIND THE POSITION IN THE NEXT BOUND LIST
IF(NEWBND.LT.LSTBND) GO TO 3030
C END OF LIST
II=LSTNOD
NEXT(LSTNOD)=NEWNOD
NEXT(NEWNOD)=0
LSTNOD=NEWNOD
LSTBND=NEWBND
IF(II.NE.CURND) GO TO 3070
NXTBND=NEWBND
NXTNOD=NEWNOD
GO TO 3070
3030 IF(NEWBND.GT.NXTBND) GO TO 3040
C TOP OF THE LIST
NEXT(NEWNOD)=NXTNOD
NXTBND=NEWBND
NXTNOD=NEWNOD
GO TO 3070
C IN THE MIDDLE OF THE LIST
3040 J=NXTNOD
3050 I=NEXT(J)
IF(NEWBND.LE.BND(I)) GO TO 3060
J=I
GO TO 3050
3060 NEXT(NEWNOD)=I
NEXT(J)=NEWNOD
3070 IF(NFIX.NE.MH1) GO TO 2000
C FIX THE ONLY FREE VARIABLE
4000 MCOST=MTOT
TERND=TERND+1
HCOST=HTOT
DO 4010 I=1,MH
IF(F(I).EQ.0) IFIX=I

```

```

IF(X(I).EQ.0) GO TO 4010
MCCOST=MCCOST+TM(I)
HCCOST=HCCOST-TM(I)
4010 CONTINUE
NEXT(CURND)=NXTAV
NXTAV=CURND
BZERO=MCCOST
IF(BZERO.LT.HCCOST) BZERO=HCCOST
BONE=MCCOST+TM(IFIX)
HCCOST=HCCOST-TM(IFIX)
IF(BONE.LT.HCCOST) BONE=HCCOST
IF(BONE.GE.ZPLUS.AND.BZERO.GE.ZPLUS) GO TO 1000
NS=0
IPROB=1
IF(BONE.GT.BZERO) IPROB=0
IF(IPROB.GT.0) GO TO 4030
4020 SUMM=MCCOST
SUMH=HCCOST+TM(IFIX)
X(IFIX)=0
GO TO 4040
4030 SUMM=MCCOST+TM(IFIX)
SUMH=HCCOST
X(IFIX)=1
4040 CALL SCEDLE
NS=NS+1
IF(SCHDTM.LT.ZPLUS) GO TO 5000
4500 IF(NS.GT.1) GO TO 1000
IF(IPROB.EQ.0) GO TO 4510
IF(BZERO.GE.ZPLUS) GO TO 1000
GO TO 4020
4510 IF(BONE.GE.ZPLUS) GO TO 1000
GO TO 4030
C REPLACE THE INCUMBENT SOLUTION AND PRUNE THE TREE
5000 ZPLUS=SCHDTM
TIMSL=ABS(ITIME(TM1, TM2)-TIMIN)/5
WRITE(6,5001) TIMSL
5001 FORMAT(' SOLUTION FOUND AT',I9,' MILSEC')
DO 5010 I=1,MH
5010 XPLUS(I)=X(I)
WRITE(6,5020) ZPLUS
5020 FORMAT(' THE NEW SOLUTION IS', I10)
WRITE(6,5025)(XPLUS(I),I=1,MH)
5025 FORMAT(1X,30I2)
IF(ISTOP.GT.0) GO TO 6000
IF(LSTBND.LT.ZPLUS) GO TO 4500
IF(NXTBND.GE.ZPLUS) GO TO 5050
J=NXTNOD
5030 I=NEXT(J)
IF(BND(I).GE.ZPLUS) GO TO 5040
J=I
GO TO 5030
5040 NEXT(I)=NXTAV
NXTAV=I
LSTNOD=J

```

```

LSTBND=BNB(J)
NEXT(J)=0
GO TO 4500
5050 IF(NS.GT.1) GO TO 6000
      ISTOP=1
      GO TO 4500
6000 WRITE(6,6001) ZPLUS
6001 FORMAT(' THE PROBLEM SOLUTION IS',I10,/)
      TIMSL=ABS(ETIME(TM1, TM2)-TIMIN)/K
      WRITE(6,6002) TIMSL
6002 FORMAT(' OPTIMAL VERIFIED AT',I9,' MLSEC')
      ZPLUS=LARGE
      DO 6010 I=1, MH
6010 X(I)=XPLUS(I)
      WRITE(6,6011) (X(I), I=1, MH)
6011 FORMAT(' RHE ASSIGNMENT IS ',/,35I2)
      WRITE(6,6012) TIMSQ
      WRITE(6,6013) TNOD, MOSNOD, TERND, NOSCD, NOFIN
      CALL SCEDLE
6012 FORMAT(' SCHEDULING TOOK',I9,' MLSEC')
6013 FORMAT(1X,2I5)
      WRITE(6,6015)
6015 FORMAT(' TASK',2X,'START')
      DO 6020 I=1, HEND
6020 WRITE(6,6021) I, ST(I)
6021 FORMAT(1X,I3,2X,I6)
      END

```

```

COMPILER (FLD=ABS)
SUBROUTINE SCEDLE
IMPLICIT INTEGER (A-Z)
COMMON MH, MST, MEND, HST, HEND, T, TM, TH, TIME, X, NPR, FROM, TO, TYPE,
$ SUMM, SUMH, ST, LARGE, SCHDTM, ZPLUS, TIMSQ, NOSCD, NOFIN
DIMENSION TM(51), TH(51), TIME(101), X(101), FROM(200), TO(200),
$ TYPE(200), ST(101)
DIMENSION PNT(101), NP(101), CC(101), ES(101), B(101), IPT(101)
DIMENSION IBN(200), IEN(200), IND(200), INTO(200), OUT(200) ,
$ D(101,101)
C SET UP THE PROPER TIMES FOR ASSIGNED TASKS
NOSCD=NOSCD+1
TIMIN=ETIME(TM1, TM2)
SCHDTM=LARGE+1
SMALL=-LARGE
DO 20 I=1, MH
IF(X(I).EQ.0) GO TO 10
TIME(I)=TM(I)
GO TO 20
10 TIME(I)=TH(I)
20 CONTINUE

```

```

DO 30 I=1,NEND
PNT(I)=0
CC(I)=0
30 NP(I)=0
ICT=0
NCC=0
DO 40 I=1,T
DO 40 J=1,T
40 D(I,J)=-LARGE
C SET UP THE PRECEDENCE RELATIONSHIPS FOR THIS ASSIGNMENT
DO 100 IAR=1,NPR
TY=TYPE(IAR)
I=FROM(IAR)
J=TO(IAR)
C TY=0 IS UNCONDITIONAL PRECEDENCE
IF(TY.EQ.0) GO TO 90
IF(TY.LT.T) GO TO 50
C TY = LARGE IS I,J ASSIGNED THE SAME
IF(X(I).NE.X(J)) GO TO 100
GO TO 90
50 ATY=ABS(TY)
IF(I.EQ.ATY) GO TO 70
C J IS THE CONDITIONAL VARIABLE
IF(TY.LT.0) GO TO 60
C TYPE = +J
IF(X(J).EQ.0) GO TO 100
GO TO 90
C TYPE = -J
60 IF (X(J).GT.0) GO TO 100
GO TO 90
C I IS THE CONDITIONAL VARIABLE
70 IF(TY.LT.0) GO TO 80
C TYPE = +I
IF(X(I).EQ.0) GO TO 100
GO TO 90
C TYPE = -I
80 IF(X(I).GT.0) GO TO 100
C ADD THE PRECEDENCE RELATION
90 ICT=ICT+1
IBN(ICT)=I
IEN(ICT)=J
IND(ICT)=1
NP(J)=NP(J)+1
D(I,J)=TIME(I)
100 CONTINUE
NARCS=ICT
C DETERMINE POINTERS FOR LIST
I=0
GO 110 IAR=1,NARCS
J=IBN(IAR)
IF(I.EQ.J) GO TO 110
PNT(J)=IAR
I=J
110 CONTINUE

```

```

C   COMPLETE THE LONGEST PATH MATRIX
DO 120 I=1,HEND
120 D(I,T)=TIME(I)
C   DETERMINE THE ORDERED TO-FROM LIST
II=0
DO 140 J=1,HEND
IP=0
DO 130 I=1,HEND
IF(D(I,J).LT.0) GO TO 130
II=II+1
IF(IP.EQ.0) IP=II
INTO(II)=J
OUT(II)=I
IF(X(I).EQ.X(J)) GO TO 130
CC(J)=1
130 CONTINUE
IPT(J)=IP
140 CONTINUE
CM=0
SUMMR=SUMM
SUMHR=SUMH
CH=0
DO 150 I=1,HEND
150 IF(CC(I).GT.0) NCC=NCC+1
IF(NCC.EQ.0) GO TO 9000
C   DETERMINE THE SCHEDULABLE TASKS
1000 NSM=0
NSH=0
ISTAR=0
JSTAR=0
IF(NCC.EQ.0) GO TO 9000
DO 1020 I=1,HEND
IF(NP(I).NE.0) GO TO 1020
IF(X(I).EQ.0) GO TO 1010
NSM=NSM+1
IF(NSM.GT.1) GO TO 1030
ISTAR=I
GO TO 1020
1010 NSH=NSH+1
IF(NSH.GT.1) GO TO 1030
JSTAR=I
1020 CONTINUE
IF(ISTAR.EQ.0) GO TO 1025
ES(ISTAR)=CM
IF(CC(ISTAR).EQ.0) GO TO 1025
FIN=0
J=IPT(ISTAR)
1023 K=OUT(J)
FN=ST(K)+TIME(K)
IF(FIN.LT.FN) FIN=FN
J=J+1
IF(INTO(J).EQ.ISTAR) GO TO 1023
IF(FIN.GT.ES(ISTAR)) ES(ISTAR)=FIN
1025 IF(JSTAR.EQ.0) GO TO 2000

```

```

ES(JSTAR)=CH
IF(CC(JSTAR).EQ.0) GO TO 2000
FIN=0
J=IPT(JSTAR)
1027 K=OUT(J)
FN=ST(K)+TIME(K)
IF(FIN.LT.FN) FIN=FN
J=J+1
IF(INTO(J).EQ.JSTAR) GO TO 1027
IF(FIN.GT.ES(JSTAR)) ES(JSTAR)=FIN
IF(NSM.EQ.0.AND.NSH.EQ.0) GO TO 9000
GO TO 2000
C SOLVE THE LONGEST PATH PROBLEM
1030 DO 1040 I=1,T
DO 1040 J=1,T
IF(D(J,I).LE.SMALL) GO TO 1040
DO 1035 K=1,T
IF(D(I,K).LE.SMALL) GO TO 1035
IWK=D(J,I)+D(I,K)
IF(IWK.GT.D(J,K)) D(J,K)=IWK
1035 CONTINUE
1040 CONTINUE
C FIND THE LONGEST PATH FOR M AND H
LPM=0
LPH=0
DO 1060 I=1,MH
IF(NP(I).LT.0) GO TO 1060
IF(X(I).EQ.0) GO TO 1050
IF(D(I,T).GT.LPM) LPM=D(I,T)
GO TO 1060
1050 IF(D(I,T).GT.LPH) LPH=D(I,T)
1060 CONTINUE
DO 1070 I=MST,MEND
IF(NP(I).LT.0) GO TO 1070
IF(D(I,T).GT.LPM) LPM=D(I,T)
1070 CONTINUE
DO 1080 I=HST,HEND
IF(NP(I).LT.0) GO TO 1080
IF(D(I,T).GT.LPH) LPH=D(I,T)
1080 CONTINUE
II=LPH+CH
JJ=LPM+CM
IF(II.GE.ZPLUS.OR.JJ.GE.ZPLUS) GO TO 9990
C COMPUTE THE EARLIEST START TIMES FOR EACH SCHEDULEABLE TASK
DO 1130 I=1,HEND
IF(NP(I).NE.0) GO TO 1130
IF(CC(I).GT.0) GO TO 1110
IF(X(I).EQ.0) ES(I)=CH
IF(X(I).EQ.0) ES(I)=CH
GO TO 1130
1110 FIN=0
J=IPT(I)
1120 K=OUT(J)
FN=ST(K)+TIME(K)

```

```

IF(FIN.LT.FN) FIN=FN
J=J+1
IF(INTO(J),EQ.I) GO TO 1120
IF(X(I).GT.0.AND.FIN.LT.CM) FIM=CM
IF(X(I).EQ.0.AND.FIN.LT.CH)FIN=CH
ES(I)=FIN
1130 CONTINUE
C DETERMINE THE BOUNDS FOR EACH CANDIDATE TASK
BLPM=CM+LPM
BLPH=CH+LPH
BSM=CM+SUMMR
BSH=CH+SUMHR
IF(BSM.LT.BLPM) BSM=BLPM
IF(BSH.LT.BLPH) BSH=BLPH
DO 1240 I=1,HEXD
IF(NP(I).NE.0) GO TO 1240
IF(X(I).GT.0) GO TO 1210
BND=ES(I)+SUMMR
IF(BND.LT.BSM) BND=BSM
GO TO 1220
1210 BND=ES(I)+SUMMR
IF(BND.LT.BSH) BND=BSH
1220 IF(BND.LT.D(I,T)) BND=D(I,T)
XI=X(I)
LPI=ES(I)+TIME(I)
MAX=-1
DO 1230 J=1,HEXD
IF(NP(J).LT.0) GO TO 1230
IF(X(J).NE.XI) GO TO 1230
IF(J.EQ.I) GO TO 1230
LPJ=LPI+D(J,T)
IF(MAX.LT.LPJ) MAX=LPJ
1230 CONTINUE
IF(BND.LT.MAX) BND=MAX
B(I)=BND
1240 CONTINUE
ISTAR=0
JSTAR=0
C FIND THE SMALLEST BOUND
MINM=LARGE
MINH=LARGE
DO 1320 I=1,HEXD
IF(NP(I).NE.0) GO TO 1320
IF(X(I).EQ.0) GO TO 1310
IF(B(I).GE.MINM) GO TO 1320
MINM=B(I)
ISTAR=I
GO TO 1320
1310 IF(B(I).GE.MINH) GO TO 1320
MINH=B(I)
JSTAR=I
1320 CONTINUE
IF(MINM.GE.ZPLUS.AND,ISTAR.GT.0) GO TO 9990
IF(MINH.GE.ZPLUS.AND,JSTAR.GT.0) GO TO 9990

```

```

2000 IF(ISTAR,EQ.0) GO TO 2010
      ST(ISTAR)=ES(ISTAR)
      CM=ES(ISTAR)+TIME(ISTAR)
      NP(ISTAR)=-1
      SUMMR=SUMMR-TIME(ISTAR)
      IF(CC(ISTAR).GT.0) NCC=NCC-1
2010 IF(JSTAR,EQ.0) GO TO 2020
      ST(JSTAR)=ES(JSTAR)
      CH=ES(JSTAR)+TIME(JSTAR)
      NP(JSTAR)=-1
      SUMHR=SUMHR-TIME(JSTAR)
      IF(CC(JSTAR).GT.0) NCC=NCC-1
2020 NODE=ISTAR
      IF(ISTAR,EQ.0) NODE=JSTAR
2030 J=PNT(NODE)
      IF(J,EQ.0) GO TO 2050
2040 I=IEN(J)
      NP(I)=NP(I)-1
      IND(J)=0
      J=J+1
      IF(J.GT.NARCS) GO TO 2050
      IF(IBN(J),EQ.NODE) GO TO 2040
2050 IF(NODE,EQ.JSTAR) GO TO 3000
      IF(JSTAR,EQ.0) GO TO 3000
      NODE=JSTAR
      GO TO 2030
C     SET UP NEW LONGEST PATH PROBLEM
3000 DO 3010 I=1,T
      DO 3010 J=1,T
3010 D(I,J)=SMALL
3015 DO 3020 IAR=1,NARCS
      IF(IND(IAR),EQ.0) GO TO 3020
      I=IBN(IAR)
      J=IEN(IAR)
      D(I,J)=TIME(I)
3020 CONTINUE
      DO 3030 I=1,HEND
      IF(NP(I),LT.0) GO TO 3025
      D(I,T)=TIME(I)
      GO TO 3030
3025 D(I,T)=SMALL
3030 CONTINUE
      GO TO 1000
9000 CONTINUE
9010 IPROB=0
      DO 9050 I=1,HEND
      IF(NP(I),NE.0) GO TO 9050
      IF(X(I),GT.0) GO TO 9020
      ST(I)=CH
      CH=CH+TIME(I)
      GO TO 9030
9020 ST(I)=CM
      CM=CM+TIME(I)
9030 IPROB=1

```

```
NP(I)=-1
J=PNT(I)
IF(J.EQ.0) GO TO 9050
9040 K=IEN(J)
NP(K)=NK(K)-1
J=J+1
IF(IBN(J).EQ.1) GO TO 9040
9050 CONTINUE
IF(IPROB.GT.0) GO TO 9010
CH=0
CM=0
DO 9070 I=1,HEND
II=ST(I)+TIME(I)
IF(X(I).GT.0) GO TO 9060
IF(II.GT.CH) CH=II
GO TO 9070
9060 IF(CM.LT.II) CM=II
9070 CONTINUE
SCHDTM=CM
IF(SCHDTM.LT.CH) SCHDTM=CH
NOFIN=NOFIN+1
TIMSQ=TIMSQ+ABS(ITIME(TM1, TM2)-TIMIN)/5
RETURN
9990 SCHDTM=LARGE
TIMSQ=TIMSQ+ABS(ITIME(TM1, TM2)-TIMIN)/5
RETURN
END
```

## APPENDIX C

COMPUTER CODE FOR THE UNCONSTRAINED SEQUENCE  
DEPENDENT PROCESSING TIMES PROBLEM

```

      IMPLICIT INTEGER (A-Z)
      LOGICAL BKTRK, THRU
      COMMON BKTRK, H, HEND, HST, IEQ, IMAX, INDIC, TSTAR, JMAX, JSTAR, L,
*   LARGE, M, MAX, MEND, MH, MH1, MHP, N, N1, SUMH, SUMM, T, TCOST, THRU, MST,
*   YES, C(62,62), COST(62,62), R(62,62), S(20), UNDER(20), X(20),
*   XPLUS(20), XX(62), B(62), BND(62), BNDH(62), BNDM(62), ICIN(62),
*   IRIN(62), LCO(62), LRO(62), MRH(62), MRM(62), ZT(62), ZCOST, ZPLUS
      THRU=.FALSE.
      BKTRK=.FALSE.
      READ(11) MH, M, H
      MST=MH+1
      MEND=MH+M
      HST=MEND+1
      HEND=MEND+H
      T=MEND+1
      LARGE=999999
      ZPLUS=LARGE
      READ(11) COST
      CALL FNDMIN
      CALL ROUNDS
      CALL STARTX
200  IZ=0
      PMORH=1
      IF(SUMH.GT.SUMM) PMORH=0
210  IF(PMORH.EQ.1) GO TO 215
      CALL SETUPH
      NHS=NHS+1
      GO TO 218
215  CALL SETUPM
      NMS=NMS+1
218  IF(BKTRK) GO TO 400
      CALL SEQNCE
      IF(BKTRK) GO TO 400
      NSLV=NSLV+1
      IF(PMORH.EQ.1) MCOST=ZCOST
      IF(PMORH.EQ.0) HCOST=ZCOST
      IF(IZ.EQ.1) GO TO 300
      IZ=1
      PMORH=1-PMORH
      GO TO 210
300  IF(MCOST.LT.HCOST) ZPLUS=HCOST
      IF(HCOST.GE.HCOST) ZPLUS=MCOST
      WRITE(6,8) ZPLUS
      8  FORMAT(' THE NEW INCUMBENT IS, ,I9)
      DO 310 I=1, MH
310  XPLUS(I)=X(I)
400  CALL BAKTRK
      IF(THRU) GO TO 410
      GO TO 200
410  CALL OUTPUT
      END

```

```

SUBROUTINE BOUNDS
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK, THRU
DO 10 I=1, T
  BND(I)=0
  DO 10 J=1, T
10  C(I, J)=COST(I, J)
    DO 40 I=1, MEND
      MIN=LARGE
      DO 20 J=1, MEND
20  IF (MIN.GT.C(I, J)) MIN=C(I, J)
        IF (MIN.GT.C(I, T)) MIN=C(I, T)
        BND(I)=MIN
      DO 30 J=1, MEND
30  C(I, J)=C(I, J)-MIN
40  C(I, T)=C(I, T)-MIN
      MIN=LARGE
      DO 50 J=1, MEND
50  IF (MIN.GT.C(T, J)) MIN=C(T, J)
        BND(T)=MIN
      DO 60 J=1, MEND
60  C(T, J)=C(T, J)-MIN
      DO 80 J=1, MEND
      MIN=LARGE
      DO 70 I=1, MEND
70  IF (MIN.GT.C(I, J)) MIN=C(I, J)
        IF (MIN.GT.C(T, J)) MIN=C(T, J)
        BND(J)=BND(J)+MIN
80  CONTINUE
      MIN=LARGE
      DO 90 I=1, MEND
90  IF (MIN.GT.C(I, T)) MIN=C(I, T)
        BND(T)=BND(T)+MIN
        SUMM=BND(T)
      DO 100 I=MST, MEND
100  SUMM=SUMM+BND(I)
      DO 110 I=1, MH
110  BNDM(I)=BND(I)
      DO 120 I=1, T
        BND(I)=0
      DO 120 J=1, T
120  C(I, J)=COST(I, J)
      DO 150 I=1, MH
        MIN=LARGE
      DO 130 J=1, MH
130  IF (MIN.GT.C(I, J)) MIN=C(I, J)
      DO 140 J=HST, T
140  IF (MIN.GT.C(I, J)) MIN=C(I, J)
        BND(I)=MIN
      DO 150 J=1, T
        C(I, J)=C(I, J)-MIN
150  CONTINUE
      DO 190 I=HST, T
        MIN=LARGE
      DO 160 J=1, MH
160  IF (MIN.GT.C(I, J)) MIN=C(I, J)
      DO 170 J=HST, T
170  IF (MIN.GT.C(I, J)) MIN=C(I, J)

```

```

      BND(I)=MIN
      DO 180 J=1,T
180  C(I,J)=C(I,J)-MIN
190  CONTINUE
      DO 220 J=1,MH
      MIN=LARGE
      DO 200 I=1,MH
200  IF(MIN.GT.C(I,J)) MIN=C(I,J)
      DO 210 I=HST,T
210  IF(MIN.GT.C(I,J)) MIN=C(I,J)
      BND(J)=BND(J)+MIN
220  CONTINUE
      DO 250 J=HST,T
      MIN=LARGE
      DO 230 I=1,MH
230  IF(MIN.GT.C(I,J)) MIN=C(I,J)
      DO 240 I=HST,T
240  IF(MIN.GT.C(I,J)) MIN=C(I,J)
      BND(J)=BND(J)+MIN
250  CONTINUE
      DO 260 I=1,MH
260  BNDH(I)=BND(I)
      SUMH=0
      DO 270 I=HST,T
270  SUMH=SUMH+BND(I)
      RETURN
      END

```

```

SUBROUTINE BKTRK
  IMPLICIT INTEGER (A-Z)
  LOGICAL BKTRK,THRU
  BKTRK=.FALSE.
  DO 20 I=MH,1,-1
    K=IABS(S(I))
    IF(UNDER(K).NE.0) GO TO 20
    S(I)=-S(I)
    UNDER(K)=1
    II=I+1
    DO 10 J=II,MH
      K=IABS(S(J))
10    UNDER(K)=0
      GO TO 30
20    CONTINUE
      THRU=.TRUE.
      RETURN
30    IF(S(I).LT.0) GO TO 40
      K=S(I)
      SUMM=SUMM+BNDM(K)
      SUMH=SUMH-BNDH(K)
      X(K)=1
      GO TO 50
40    K=-S(I)
      SUMM=SUMM-BNDM(K)
      SUMH=SUMH+BNDH(K)
      X(K)=0
50    IF(SUMM.GE.ZPLUS) GO TO 5
      IF(SUMH.GE.ZPLUS) GO TO 5
      RETURN
      END

```

```

SUBROUTINE RETURN
  C=0
  IC=ST=0
  II=
  DO 20 I=1,MH
  IF(X(I).EQ.0) GO TO 20
  II=II+1
  XX(II)=I
  JJ=
  DO 10 J=1,MH
  IF(Y(J).EQ.0) GO TO 10
  JJ=JJ+1
  C(II,JJ)=COST(I,J)
10 CONTINUE
  DO 15 J=MSI,MENI
  JJ=JJ+1
15 C(II,JJ)=COST(I,J)
20 CONTINUE
  NI=I+1
  NI=NI+1
  DO 25 I=1,NI
  IRM(I)=1
  I(IRM(I))=1
  LCO(I)=0
25 LRM(I)=0
  MH1=II+1
  MH=II
  II=MH
  DO 40 I=MH1,N1
  II=II+1
  XX(I)=II
  JJ=
  DO 26 J=1,MH
  IF(Y(J).EQ.0) GO TO 26
  JJ=JJ+1
  C(II,JJ)=COST(II,J)
26 CONTINUE
  JJ=MH
  DO 30 J=MH1,N1
  JJ=JJ+1
30 C(II,J)=COST(II,JJ)
  C(I,II)=COST(II,I)
40 C(N,I)=COST(I,II)
  II=
  DO 50 I=1,MH
  IF(X(I).EQ.0) GO TO 50
  II=II+1
  C(II,N)=COST(I,T)
  C(N,II)=COST(T,I)
50 CONTINUE
  C(N,N)=LARGE
  XX(N)=T
  K=0
  DO 60 I=1,MH
  IF(X(I).EQ.0) GO TO 60
  K=K+1
  R(K)=MRM(I)
60 CONTINUE
  K=MSI-1
  DO 70 I=MH1,N1
  K=K+1
70 R(I)=MRM(K)

```

```

      B(N)=MRM(T)
      DO 100 I=1,N
      -- MIN=B(I)
      DO 80 J=1,MHP
      80 IF(MIN.GT.C(I,J)) MIN=C(I,J)
      DO 90 J=1,N
      90 C(I,J)=C(I,J)-MIN
      TCOST=TCOST+MIN
      100 CONTINUE
      IF(TCOST.GE.ZPLUS) GO TO 140
      DO 130 J=1,N
      MIN=LARGE
      DO 110 I=1,N
      110 IF(MIN.GT.C(I,J)) MIN=C(I,J)
      IF(MIN.EQ.C) GO TO 130
      TCOST=TCOST+MIN
      DO 120 I=1,N
      120 C(I,J)=C(I,J)-MIN
      130 CONTINUE
      IF(TCOST.GE.ZPLUS) GO TO 140
      RETURN
      140 BKTRK=.TRUE.
      RETURN
      END

```

```

SUBROUTINE FNDMIN
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,THRU
DO 30 I=1,MH
MIN=LARGE
DO 10 J=MST,MEND
10 IF(MIN.GT.COST(I,J)) MIN=COST(I,J)
IF(MIN.GT.COST(I,T)) MIN=COST(I,T)
MRM(I)=MIN
MIN=LARGE
DO 20 J=HST,T
20 IF(MIN.GT.COST(I,J)) MIN=COST(I,J)
30 MRH(I)=MIN
DO 50 I=MST,MEND
MIN=LARGE
DO 40 J=MST,MEND
40 IF(MIN.GT.COST(I,J)) MIN=COST(I,J)
IF(MIN.GT.COST(I,T)) MIN=COST(I,T)
50 MRM(I)=MIN
MIN=LARGE
DO 60 J=MST,MEND
60 IF(MIN.GT.COST(T,J)) MIN=COST(T,J)
MRM(T)=MIN
DO 80 I=HST,T
MIN=LARGE
DO 70 J=HST,T
70 IF(MIN.GT.COST(I,J)) MIN=COST(I,J)
80 MRH(I)=MIN
RETURN
END

```

```

SUBROUTINE SETUPH
IMPLICIT INTEGER (A-Z)
LOGICAL B&TRK,THRU
L=0
II=0
TCOST=0
DO 20 I=1,MH
IF(X(I).GT.0) GO TO 20
II=II+1
XX(II)=I
JJ=0
DO 10 J=1,MH
IF(X(J).GT.0) GO TO 10
JJ=JJ+1
C(II,JJ)=COST(I,J)
10 CONTINUE
DO 15 J=HST,T
JJ=JJ+1
15 C(II,JJ)=COST(I,J)
20 CONTINUE
N1=II+H
N=N1+1
MH1=1+II
MHP=II
II=HST-1
DO 25 I=1,N
IRIN(I)=1
ICIN(I)=1
LR0(I)=0
25 LCO(I)=0
DO 40 I=MH1,N
II=II+1
XX(II)=II
JJ=HST-1
DO 30 J=MH1,N
JJ=JJ+1
30 C(I,J)=COST(II,JJ)
JJ=0
DO 35 J=1,MH
IF(X(J).GT.0) GO TO 35
JJ=JJ+1
C(I,JJ)=COST(II,J)
35 CONTINUE
40 CONTINUE
K=0
DO 50 I=1,MH
IF(X(I).GT.0) GO TO 50
K=K+1
B(K)=MRH(I)
50 CONTINUE
K=HST-1
DO 60 I=MH1,N
K=K+1
60 B(I)=MRH(K)
DO 90 I=1,N
MIN=B(I)
DO 70 J=1,MHP
70 IF(MIN.GT.C(I,J)) MIN=C(I,J)
DO 80 J=1,N
80 C(I,J)=C(I,J)-MIN

```

```

TCOST=TCOST+MIN
90 CONTINUE
IF(TCOST.GE.ZPLUS) GO TO 130
DO 120 J=1,M
MIN=LARGE
DO 100 I=1,N
100 IF(MIN.GT.C(I,J)) MIN=C(I,J)
IF(MIN.EQ.0) GO TO 120
DO 110 I=1,N
110 C(I,J)=C(I,J)-MIN
TCOST=TCOST+MIN
120 CONTINUE
IF(TCOST.GE.ZPLUS) GO TO 130
RETURN
130 BKTRK=.TRUE.
RETURN
END

```

```

SUBROUTINE STARTX
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,THRU
DO 20 I=1,MH
IF(SUMM.LE.SUMH) GO TO 10
X(I)=0
SUMH=SUMH+BNDH(I)
S(I)=-I
GO TO 20
10 X(I)=1
SUMM=SUMM+BNDM(I)
S(I)=I
20 CONTINUE
RETURN
END

```

```

SUBROUTINE OUTPUT
1 FORMAT(' THE PROBLEM SOLUTION IS',/)
2 FORMAT(' THE ASSIGNMENT IS',/,15(1X,I2),/)
3 FORMAT(' M COST IS',I8)
4 FORMAT(' THE SEQUENCE IS',/,1X,0',15(1=,I2))
5 FORMAT(' H COST IS',I8)
WRITE(6,1)
WRITE(6,2) (XPLUS(I),I=1,MH)
DO 10 I=1,MH
10 X(I)=XPLUS(I)
ZPLUS=LARGE
PMORH=1
20 CALL SETUPM
CALL SEQNCE
IF(PMORH.EQ.0) GO TO 30
WRITE(6,3) ZCOST
WRITE(6,4) (ZT(I),I=1,N)
PMORH=0
CALL SETUPH
GO TO 20
30 WRITE(6,5) ZCOST
WRITE(6,4) (ZT(I),I=1,N)
RETURN
END

```

APPENDIX D

COMPUTER CODE FOR TASK ASSIGNMENT AND  
SEQUENCING OF CREWS

```

COMMON COST(75,75)
COMMON C(60,60),R(60,60),ICIN(60),IRIN(60),LCO(60),LRO(60),ZT(60)
COMMON BOUND(50,5),P(50,5)
COMMON XPLUS(50),XX(50),X(50)
COMMON PREJ(5000),END(5000),VN(5000),AN(5000),BND(5000),ZN(5000),
* LN(5000),NXTBND(5000),ZP(5),LP(5),ZERO(5),LERO(5),IST(5),IEN(5),
* ZPL(5),IEQ,IMAX,INDIC,ISTAR,JMAX,JUSTAR,L,LARGE,M,MH,N,N1,T,TCOST,
* ZCOST,ZPLUS,LMAX,NPR,NXTAV,NN1,NXTCST,IFX,KFX,ZBND,K2,NN2,DIJ,
* LSTBND,SEPIT,BKTRK,FEAS,NOEND,OUTRM
COMMON IH(5),MINRED,MAXRED,TNOD,MNOD
COMMON NOLIFO,LEVEL
LOGICAL NOLIFO
TM=ITIME(TM1, TM2)
NOLIFO=.FALSE.
LEVEL=0
CALL READIT
CALL INITAL
GO TO 200
100 IF(NXTCST.EQ.1) GO TO 600
CALL CHOOSE
IF(SEPIT.EQ.2) GO TO 100
IF(SEPIT) 110,200,300
110 NPR=AN(NXTCST)
NTS=NTS+1
CALL SEQNCE
IF(BKTRK) GO TO 400
PCT=(LEVEL*1000)/MH
IF(PCT.GE.750) NOLIFO=.FALSE.
NFIN=NFIN+1
ZN(NXTCST)=ZCOST
LN(NXTCST)=LMAX
ZP(NPR)=ZCOST
LP(NPR)=LMAX
IF(ZBND.LT.ZCOST) ZBND=ZCOST
CALL FEASBL
IF(FEAS) GO TO 500
200 CALL ROUNDS
CALL PICKIT
300 CALL ADDEM
IF(OUTRM) GO TO 700
GO TO 100
400 CALL DELND
IF(NOEND) GO TO 100
GO TO 600
500 CALL NEWINC
TME=ABS(ITIME(TM1, TM2)-TM)/5
WRITE(6,501) TME
501 FORMAT(' NEW INCUMBENT FOUND AT',I9,' MLSEC')
CALL DELND
NOLIFO=.TRUE.
IF(.NOT.NOEND) GO TO 600
CALL PRUNIT
IF(NOEND) GO TO 100
600 WRITE(6,601) NTS,NFIN,' MNOD',TNOD
601 FORMAT(' NTS',I4,' NFIN',I5,' MNOD',I5,' TNOD',I5)
TME=ABS(ITIME(TM1, TM2)-TM)/5
WRITE(6,602) TME
CALL OUTPUT
602 FORMAT(' OPTIMAL VERIFIED AT',I9,' MLSEC')
STOP
700 WRITE(6,701)
701 FORMAT(' NO ROOM')
END

```

```

SUBROUTINE CHOOSE
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
LOGICAL NOLIFO
SEPII=-1
IF(NXTCST.NE.NN1) GO TO 100
I=VN(NXTCST)
K=AN(NXTCST)
X(I)=K
LEVEL=LEVEL+1
RETIJPN
100 DO 110 I=1,MH
X(I)=0
DO 110 K=1,M
110 P(I,K)=0
DO 120 K=1,M
120 ZP(K)=0
LEVEL=0
I=PREO(NXTCST)
IF(I.NE.1) GO TO 130
ZBND=-1
DO 125 I=1,M
ZP(I)=ZERO(I)
IF(ZBND.LT.ZP(I)) ZBND=ZP(I)
125 LP(I)=LERO(I)
I=NXTCST
II=VN(I)
IF(II.LT.0) GO TO 126
KK=AN(I)
X(II)=KK
LEVEL=LEVEL+1
RETURN
126 II=-II
KK=AN(I)
P(II,KK)=1
I=II
J=0
GO TO 175
130 II=VN(I)
IF(II.LT.0) GO TO 150
KK=AN(I)
X(II)=KK
IF(ZP(KK).GT.0) GO TO 140
ZP(KK)=ZN(I)
LP(KK)=LN(I)
140 I=PREO(I)
IF(I.EQ.1) GO TO 160
GO TO 130
150 II=-II
KK=AN(I)
P(II,KK)=1
I=PREO(I)
IF(I.EQ.1) GO TO 160
GO TO 130
160 ZBND=-1
DO 170 K=1,M
IF(ZP(K).GT.0) GO TO 170
ZP(K)=ZERO(K)
LP(K)=LERO(K)
170 IF(ZBND.LT.ZP(K)) ZBND=ZP(K)

```

```

IF(VN(NXTCST).LT.0) GO TO 172
I=VN(NXTCST)
K=AN(NXTCST)
X(I)=K
GO TO 210
172 I=-VN(NXTCST)
J=0
K=AN(NXTCST)
P(I,K)=1
175 DO 180 K=1,M
IF(P(I,K).GT.0) GO TO 180
J=J+1
IF(J.GT.1) GO TO 200
KK=K
180 CONTINUE
IFX=I
KFX=KK
K2=1
IF(KK.EQ.1) K2=2
BOUND(IFX,K2)=LARGE
MINR=COST(I,T)
MINC=COST(T,I)
DO 185 J=1,MH
IF(X(J).NE.KK.AND.X(J).NE.0) GO TO 185
IF(P(J,KK).GT.0) GO TO 185
IF(MINR.GT.COST(I,J)) MINR=COST(I,J)
IF(MINC.GT.COST(J,I)) MINC=COST(J,I)
185 CONTINUE
JJ=IST(KK)
JJJ=IEN(KK)
DO 190 J=JJ,JJJ
IF(MINR.GT.COST(T,J)) MINR=COST(I,J)
190 IF(MINC.GT.COST(J,I)) MINC=COST(J,I)
BOUND(IFX,KFX)=ZP(KK)-LP(KK)+MINR+MINC
IF(BOUND(IFX,KFX).LT.ZP(KK)) BOUND(IFX,KFX)=ZP(KK)
SEPIT=1
IF(BOUND(IFX,KFX).LT.ZPLUS) RETURN
CALL DELND
SEPIT=2
RETURN
200 SEPIT=0
210 RETURN
END

```

```

SUBROUTINE NEWINC
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
ZPLUS=-1
DO 10 K=1,M
ZPL(K)=ZP(K)
10 IF(ZPL(K).GT.ZPLUS) ZPLUS=ZPL(K)
DO 20 I=1,MH
20 XPLUS(I)=X(I)
WRITE(6,1) ZPLUS,(XPLUS(I),I=1,MH)
WRITE(6,1) (ZPL(K),K=1,M)
1 FORMAT(1X,10I5)
RETURN
END

```

```

SUBROUTINE ADDEM
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
LOGICAL NOLIFO
IF(NXTAV.GE.4998) GO TO 1000
IF(NXTAV.GT.MNOD) MNOD=NXTAV
NUMAD=2
NN1=NXTAV
NXTAV=PRED(NN1)
PREO(NN1)=NXTCST
END(NN1)=0
END(NXTCST)=1
VN(NN1)=IFX
AN(NN1)=KFX
BND(NN1)=ZBND
TNOD=TNOD+1
IF(ZBND.LT.BOUND(IFX,KFX)) BND(NN1)=BOUND(IFX,KFX)
IF(BOUND(IFX,K2).GE.ZPLUS) NUMAD=1
IF(NUMAD.EQ.1) GO TO 10
IF(NXTAV.GT.MNOD) MNOD=NXTAV
NN2=NXTAV
NXTAV=PRED(NN2)
PREO(NN2)=NXTCST
END(NN2)=0
END(NXTCST)=2
VN(NN2)=-IFX
AN(NN2)=KFX
BND(NN2)=ZBND
TNOD=TNOD+1
IF(BOUND(IFX,K2).GT.ZBND) BND(NN2)=BOUND(IFX,K2)
10 IF(NXTCST.NE.LSTBND) GO TO 20
NXTCST=NN1
LSTBND=NN1
IF(NUMAD.EQ.1) RETURN
NXTBND(NN1)=NN2
LSTBND=NN2
RETURN
20 IF(NOLIFO) GO TO 40
NXTBND(NN1)=NXTBND(NXTCST)
NXTCST=NN1
IF(NUMAD.EQ.1) RETURN
NUMAD=1
NODE=NN2
IF(BND(NN2).LT.BND(LSTBND)) GO TO 30
NXTBND(LSTBND)=NN2
LSTBND=NN2
RETURN
30 II=NXTBND(NN1)
IF(BND(NN2).GT.BND(II)) GO TO 90
NXTBND(NN2)=II
NXTBND(NN1)=NN2
RETURN
40 NXTCST=NXTBND(NXTCST)
IF(BND(NN1).GT.BND(NXTCST)) GO TO 60
NXTBND(NN1)=NXTCST
NXTCST=NN1
IF(NUMAD.EQ.1) RETURN
II=NXTBND(NN1)
IF(BND(NN2).GT.BND(II)) GO TO 50
NXTBND(NN2)=II
NXTBND(NN1)=NN2
RETURN

```

```

50 NODE=NN2
   NUMAD=1
   IF(BND(NN2).LT.BND(LSTBND)) GO TO 90
   NXTBND(LSTBND)=NN2
   LSTBND=NN2
   RETURN
60 IF(NUMAD.EQ.1) GO TO 70
   IF(BND(NN2).LT.BND(LSTBND)) GO TO 70
   IF(BND(NN1).LT.BND(LSTBND)) GO TO 65
   NXTBND(LSTBND)=NN1
   LSTBND=NN1
   NUMAD=1
65 NXTBND(LSTBND)=NN2
   LSTBND=NN2
   IF(NUMAD.EQ.1) RETURN
   NUMAD=1
70 NODE=NN1
   IF(BND(NN1).LT.BND(LSTBND)) GO TO 75
   NXTBND(LSTBND)=NN1
   LSTBND=NN1
   RETURN
75 JJ=NXTCST
   II=NXTBND(JJ)
80 IF(BND(NODE).GT.BND(II)) GO TO 90
   NXTBND(NODE)=II
   NXTBND(JJ)=NODE
   GO TO 100
90 JJ=II
   II=NXTBND(JJ)
   GO TO 80
100 IF(NUMAD.EQ.1) RETURN
   II=NODE
   NODE=NN2
   NUMAD=1
   GO TO 80
1000 OUTRM=.TRUE.
   RETURN
   END

```

```

SUBROUTINE INITAL
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
LARGE=99999999
DO 10 I=1,MH
10 X(I)=0
   ZPLUS=LARGE
   DO 20 NPR=1,M
   CALL SEQNCE
   ZERO(NPR)=ZCOST
20 LERO(NPR)=LMAX
   DO 30 NODES=2,4999
30 PREO(NODES)=NODES+1
   NXTAV=2
   NXTCST=1
   LSTBND=1
   ZBND=-1
   DO 40 K=1,M
   ZP(K)=ZERO(K)
   LP(K)=LERO(K)
40 IF(ZBND.LT.ZP(K)) ZBND=ZP(K)
   RETURN

```

```

SUBROUTINE SETUPM
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
L=0
TCOST=0
II=0
DO 5 I=1,MH
5 XX(I)=0
MST=IST(NPR)
MEND=IEN(NPR)
DO 20 I=1,MH
IF(X(I).NE.NPR) GO TO 20
II=II+1
XX(II)=I
JJ=0
DO 10 J=1,MH
IF(X(J).NE.NPR) GO TO 10
JJ=JJ+1
C(II,JJ)=COST(I,J)
10 CONTINUE
DO 15 J=MST,MEND
JJ=JJ+1
15 C(II,JJ)=COST(I,J)
20 CONTINUE
N1=IH(NPR)+II
N=N1+1
DO 25 I=1,N
ZT(I)=0
IRIN(I)=1
ICIN(I)=1
LCO(I)=0
25 LRO(I)=0
MH1=II+1
MHP=II
II=MST-1
DO 40 I=MH1,N1
II=II+1
XX(I)=II
JJ=0
DO 26 J=1,MH
IF(X(J).NE.NPR) GO TO 26
JJ=JJ+1
C(I,JJ)=COST(II,J)
26 CONTINUE
JJ=MST-1
DO 30 J=MH1,N1
JJ=JJ+1
30 C(I,J)=COST(II,JJ)
C(I,N)=COST(II,T)
40 C(N,I)=COST(T,II)
II=0
DO 50 I=1,MH
IF(X(I).NE.NPR) GO TO 50
II=II+1
C(II,N)=COST(I,T)
C(N,II)=COST(T,I)
50 CONTINUE
C(N,N)=LARGE
XX(N)=T
DO 100 I=1,N
MIN=LARGE

```

```

      DO 80 J=1,N
80  IF(MIN.GT.C(I,J)) MIN=C(I,J)
      DO 90 J=1,N
90  C(I,J)=C(I,J)-MIN
      TCOST=TCOST+MIN
100 CONTINUE
      IF(TCOST.GE.ZPLUS) GO TO 140
      DO 130 J=1,N
      MIN=LARGE
      DO 110 I=1,M
110  IF(MIN.GT.C(I,J)) MIN=C(I,J)
      IF(MIN.EQ.0) GO TO 130
      TCOST=TCOST+MIN
      DO 120 I=1,M
120  C(I,J)=C(I,J)-MIN
130 CONTINUE
      IF(TCOST.GE.ZPLUS) GO TO 140
      RETURN
140 BKTRK=.TRUE.
      RETURN
      END

```

```

SUBROUTINE PICKIT
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
MAX=-1
DO 30 I=1,MH
IF(X(I).GT.0) GO TO 30
MIN1=LARGE
MIN2=LARGE
DO 20 K=1,M
IF(P(I,K).GT.0) GO TO 20
BVAR=ROUND(J,K)
IF(BVAR.GT.MIN1) GO TO 10
MIN2=MIN1
KKK=K1
MIN1=BVAR
K1=K
GO TO 20
10 IF(BVAR.GT.MIN2) GO TO 20
MIN2=BVAR
KKK=K
20 CONTINUE
IF(MIN1.GE.ZPLUS) GO TO 40
IF(MIN2.LE.MAX) GO TO 30
MAX=MIN2
IFX=I
KFX=K1
K2=KKK
30 CONTINUE
RETURN
40 BKTRK=.TRUE.
RETURN
END

```

```

SUBROUTINE BOUNDS
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
DO 40 I=1,MH
IF(X(I).GT.C) GO TO 40
DO 30 K=1,M
IF(P(I,K).EQ.0) GO TO 5
BOUND(I,K)=LARGE
GO TO 30
5 MINR=COST(I,T)
MINC=COST(T,I)
DO 10 J=1,MH
IF(X(J).NE.K.AND.X(J).NE.0) GO TO 10
IF(P(J,K).GT.0) GO TO 10
IF(MINR.GT.COST(I,J)) MINR=COST(I,J)
IF(MINC.GT.COST(J,I)) MINC=COST(J,I)
10 CONTINUE
JJ=IST(K)
JJJ=IFN(K)
DO 20 J=JJ,JJJ
IF(MINR.GT.COST(I,J)) MINR=COST(I,J)
20 IF(MINC.GT.COST(J,I)) MINC=COST(J,I)
BOUND(I,K)=ZP(K)-LP(K)+MINR+MINC
IF(BOUND(I,K).LT.ZP(K)) BOUND(I,K)=ZP(K)
30 CONTINUE
40 CONTINUE
RETURN
END

```

```

SUBROUTINE PRUNIT
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
NOEND=.TRUE.
IF(BND(LSTBND).LT.ZPLUS) RETURN
IF(BND(NXTCST).GE.ZPLUS) GO TO 50
I=NXTBND(NXTCST)
J=NXTCST
10 IF(BND(I).GE.ZPLUS) GO TO 20
J=I
I=NXTBND(J)
GO TO 10
20 II=I
30 JJ=PRD(II)
PRD(II)=NXTAV
NXTAV=II
END(JJ)=END(JJ)-1
IF(END(JJ).GT.0) GO TO 35
II=JJ
GO TO 30
35 IF(I.EQ.LSTBND) GO TO 40
I=NXTBND(I)
GO TO 20
40 LSTBND=J
RETURN
50 NOEND=.FALSE.
RETURN
END

```

```

SUBROUTINE DELND
  IMPLICIT INTEGER (A-Z)
  LOGICAL BKTRK,FEAS,NOEND,OUTRM
  NOEND=.TRUE.
  I=N*TCST
  NXTCST=NXTBND(I)
10 J=PRED(I)
  PRED(I)=NXTAV
  NXTAV=I
  END(J)=END(J)-1
  IF(END(J).GT.0) RETURN
  I=J
  IF(I.NE.1) GO TO 10
  NOEND=.FALSE.
  RETURN
END

SUBROUTINE FEASBL
  IMPLICIT INTEGER (A-Z)
  LOGICAL BKTRK,FEAS,NOEND,OUTRM
  FEAS=.TRUE.
  DO 10 I=1,MH
  IF(X(I).GT.0) GO TO 10
  FEAS=.FALSE.
  RETURN
10 CONTINUE
RETURN
END

SUBROUTINE OUTPUT
  IMPLICIT INTEGER (A-Z)
  LOGICAL BKTRK,FEAS,NOEND,OUTRM
  1 FORMAT(' THE PROBLEM SOLUTION IS: ',/)
  2 FORMAT(' THE ASSIGNMENT IS: ',/,15(1X,I2),/)
  3 FORMAT(' PROCESSOR',I4,' COST IS',I8)
  4 FORMAT(' THE SEQUENCE IS: ',/,1X,'0-',15(12,'-'))
  WRITE(6,1)
  WRITE(6,2) (XPLUS(I),I=1,MH)
  DO 10 I=1,MH
10 X(I)=XPLUS(I)
  ZPLUS=LARGE
  DO 20 NPR=1,M
  CALL SEQNCE
  WRITE(6,3) NPR,ZCOST
  WRITE(6,4) (ZT(I),I=1,N)
20 CONTINUE
RETURN
END

SUBROUTINE READIT
  IMPLICIT INTEGER (A-Z)
  LOGICAL BKTRK,FEAS,NOEND,OUTRM
  1 FORMAT( )
  READ(5,1) MH,M
  READ(5,1) (IH(K),K=1,M)
  IST(1)=MH+1
  IEN(1)=MH+IH(1)
  DO 10 K=2,M
  J=K-1
  IST(K)=IEN(J)+1
10 IEN(K)=IEN(J)+IH(K)
  T=LEN(M)+1
  READ(11) COST

```

APPENDIX E  
COMPUTER CODE FOR THE SOLUTION OF SEQUENCE  
DEPENDENT PROCESSING TIME  
SCHEDULING PROBLEMS

```

SUBROUTINE SEQNCE
  IMPLICIT INTEGER (A-Z)
  LOGICAL BKTRK,FEAS,NOEND,OUTRM
  BKTRK=.FALSE.
  CALL SETUPM
  IF(BKTRK) RETURN
10 CALL REGRET
  IF(IEQ.EQ.1) CALL BRKTIE
  CALL FIXONE
  IF(L.EQ.N) GO TO 20
  IF(IEQ.EQ.1.AND.MINRED.EQ.0) GO TO 10
  CALL REDUCE
  IF(BKTRK) RETURN
  GO TO 10
20 CALL SOLVED
  RETURN
  END

```

```

SUBROUTINE REGRET
  IMPLICIT INTEGER (A-Z)
  LOGICAL BKTRK,FEAS,NOEND,OUTRM
  IEQ=0
  MAX=-1
  DO 5 I=1,N
  DO 5 J=1,N
5  R(I,J)=-1
  DO 40 I=1,N
  IF(IRIN(I).EQ.0) GO TO 40
  DO 30 J=1,N
  IF(ICIN(J).EQ.0) GO TO 30
  IF(C(I,J).NE.0) GO TO 30
  R(I,J)=0
  MIN=LARGE
  DO 10 II=1,N
  IF(IRIN(II).EQ.0) GO TO 10
  IF(II.EQ.I) GO TO 10
  IF(MIN.GT.C(II,J)) MIN=C(II,J)
10 CONTINUE
  R(I,J)=MIN
  MIN=LARGE
  DO 20 JJ=1,N
  IF(ICIN(JJ).EQ.0) GO TO 20
  IF(JJ.EQ.J) GO TO 20
  IF(MIN.GT.C(I,JJ)) MIN=C(I,JJ)
20 CONTINUE
  R(I,J)=R(I,J)+MIN
  IF(R(I,J).LT.MAX) GO TO 30
  MAX=R(I,J)
  IMAX=I
  JMAX=J
30 CONTINUE
40 CONTINUE
  IF(L.EQ.N1) RETURN
  II=0
  DO 60 I=1,N
  DO 60 J=1,N
60 IF(R(I,J).EQ.MAX) II=1+II
  IF(II.GT.1) IEQ=1
  MAXRED=MAX

```

```

SUBROUTINE REDUCE
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEFD,OUTRM
DO 30 I=1,N
IF(IRIN(I).EQ.0) GO TO 30
IF(C(I,JMAX).NE.0) GO TO 30
MIN=LARGE
DO 10 J=1,N
IF(ICIN(J).EQ.0) GO TO 10
IF(MIN.GT.C(I,J)) MIN=C(I,J)
10 CONTINUE
IF(MIN.EQ.0) GO TO 30
DO 20 J=1,N
20 C(I,J)=C(I,J)-MIN
TCOST=TCOST+MIN
30 CONTINUE
IF(TCOST.LT.ZPLUS) GO TO 35
BKTRK=.TRUE.
RETURN
35 DO 60 J=1,N
IF(ICIN(J).EQ.0) GO TO 60
IF(C(IMAX,J).NE.0) GO TO 60
MIN=LARGE
DO 40 I=1,N
IF(IRIN(I).EQ.0) GO TO 40
IF(MIN.GT.C(I,J)) MIN=C(I,J)
40 CONTINUE
IF(MIN.EQ.0) GO TO 60
DO 50 I=1,N
50 C(I,J)=C(I,J)-MIN
TCOST=TCOST+MIN
60 CONTINUE
IF(TCOST.LT.ZPLUS) GO TO 70
BKTRK=.TRUE.
RETURN
70 IF(INDIC.EQ.0) RETURN
MIN=LARGE
DO 80 I=1,N
IF(IRIN(I).EQ.0) GO TO 80
IF(MIN.GT.C(I,JSTAR)) MIN=C(I,JSTAR)
80 CONTINUE
IF(MIN.EQ.0) GO TO 100
TCOST=TCOST+MIN
IF(TCOST.GE.ZPLUS) GO TO 130
DO 90 I=1,N
90 C(I,JSTAR)=C(I,JSTAR)-MIN
100 MIN=LARGE
DO 110 J=1,N
IF(ICIN(J).EQ.0) GO TO 110
IF(MIN.GT.C(ISTAR,J)) MIN=C(ISTAR,J)
110 CONTINUE
IF(MIN.EQ.0) RETURN
TCOST=TCOST+MIN
IF(TCOST.GE.ZPLUS) GO TO 130
DO 120 J=1,N
120 C(ISTAR,J)=C(ISTAR,J)-MIN
RETURN
130 BKTRK=.TRUE.
RETURN
END

```

```

SUBROUTINE BRKTIE
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
MINRED=LARGE
DO 60 I=1,N
IF(IRIN(I).EQ.0) GO TO 60
DO 50 J=1,N
IF(R(I,J).NE.MAXRED) GO TO 50
IF(ICIN(J).EQ.0) GO TO 50
DIJ=0
JJ=J
NTM=N-L
IF(NTM.LE.2) GO TO 5
1 KK=LRO(JJ)
IF(KK.EQ.0) GO TO 2
JJ=KK
GO TO 1
2 IS=JJ
II=I
3 KK=LCO(II)
IF(KK.EQ.0) GO TO 4
II=KK
GO TO 3
4 JS=II
CTEMP=C(IS,JS)
C(IS,JS)=LARGE
5 DO 20 II=1,N
IF(IRIN(II).EQ.0) GO TO 20
IF(II.EQ.I) GO TO 20
MIN=LARGE
DO 10 JJ=1,N
IF(ICIN(JJ).EQ.0) GO TO 10
IF(JJ.EQ.J) GO TO 10
IF(MIN.GT.C(II,JJ)) MIN=C(II,JJ)
10 CONTINUE
DIJ=DIJ+MIN
20 CONTINUE
DO 40 JJ=1,N
IF(ICIN(JJ).EQ.0) GO TO 40
IF(JJ.EQ.J) GO TO 40
MIN=LARGE
DO 30 II=1,N
IF(IRIN(II).EQ.0) GO TO 30
IF(II.EQ.I) GO TO 30
IF(MIN.GT.C(II,JJ)) MIN=C(II,JJ)
30 CONTINUE
DIJ=DIJ+MIN
IF(DIJ.GT.0) GO TO 40
MINRED=0
IF(NTM.GT.2) C(IS,JS)=CTEMP
IMAX=I
JMAX=J
RETURN
40 CONTINUE
IF(NTM.GT.2) C(IS,JS)=CTEMP
IF(MINRED.LT.DIJ) GO TO 50
MINRED=DIJ
IMAX=I
JMAX=J
50 CONTINUE
60 CONTINUE
RETURN

```

```

SUBROUTINE FIXONE
  IMPLICIT INTEGER (A-Z)
  LOGICAL BKTRK,FEAS,NOE'D,OUTRM
  INDIC=0
  LCO(JMAX)=IMAX
  LRO(IMAX)=JMAX
  L=1+L
  IF(L.EQ.N) GO TO 130
  IRIN(IMAX)=0
  ICIN(JMAX)=0
  IF(L.EQ.N1) GO TO 130
  IF(IRIN(JMAX).EQ.1) GO TO 80
  IF(ICIN(IMAX).EQ.1) GO TO 50
  I=IMAX
10  J=LCO(I)
  IF(J.EQ.0) GO TO 20
  I=J
  GO TO 10
20  JSTAR=I
  J=JMAX
30  I=LRO(J)
  IF(I.EQ.0) GO TO 40
  J=I
  GO TO 30
40  ISTAR=J
  GO TO 120
50  I=JMAX
60  J=LRO(I)
  IF(J.EQ.0) GO TO 70
  I=J
  GO TO 60
70  ISTAR=I
  JSTAR=IMAX
  GO TO 120
80  IF(ICIN(IMAX).EQ.1) GO TO 110
  I=IMAX
90  J=LCO(I)
  IF(J.EQ.0) GO TO 100
  I=J
  GO TO 90
100 JSTAR=I
  ISTAR=JMAX
  GO TO 120
110 ISTAR=JMAX
  JSTAR=IMAX
120 IF(C(ISTAR,JSTAR).EQ.0) INDIC=1
  C(ISTAR,JSTAR)=LARGE
130 RETURN
  END

```

```
SUBROUTINE SOLVED
IMPLICIT INTEGER (A-Z)
LOGICAL BKTRK,FEAS,NOEND,OUTRM
ZCOST=0
LMAX=-1
II=1
I=N
5 J=LRO(I)
ZT(II)=XX(J)
II=II+1
I=J
IF(I.NE.N) GO TO 5
II=1
I=1
10 J=ZT(II)
ZCOST=ZCOST+COST(I,J)
IF(COST(I,J).GT.LMAX) LMAX=COST(I,J)
I=J
II=II+1
IF(I.NE.T) GO TO 10
IF(ZCOST.GE.ZPLUS) BKTRK=.TRUE.
RETURN
END
```

## BIBLIOGRAPHY

1. Ashour, Said and S. R. Hiremath, "A Branch-and-Bound Approach to the Job-Shop Scheduling Problem," *International Journal of Production Research*, vol. 11, no. 1, 47-58 (1973).
2. Ashour, Said, J. F. Vega, and R. G. Parker, "A Heuristic Algorithm for Traveling Salesman Problems," *Transportation Research*, vol. 6, 187-195 (1972).
3. Baker, K. R., Introduction to Sequencing and Scheduling, John Wiley and Sons, Inc., New York (1974).
4. Conway, R. W., W. L. Maxwell, and L. W. Miller, Theory of Scheduling, Addison-Wesley, Reading, Massachusetts (1967).
5. Geoffrion, A. M., "Integer Programming by Implicit Enumeration and Balas' Method," *SIAM Review*, vol. 9, 178-190 (1967).
6. Giffler, B. and G. Thompson, "Algorithms for Solving Production Scheduling Problems," *ORSA*, vol. 5, 487-499 (1960).
7. Hu, T. C., "Parallel Sequencing and Assembly Line Problems," *Operations Research*, vol. 9, no. 6 (1961).
8. Kilbridge, M. D. and L. Wester, "A Heuristic Method of Assembly Line Balancing," *Journal of Industrial Engineering*, vol. 12, no. 4 (1961).
9. Kvalseth, T. O. and R. G. Parker, "Ordering Human Activities with Respect to Overlap Considerations," Working Paper, School of Industrial and Systems Engineering, Georgia Institute of Technology (1973).
10. Little, J. D. C., K. G. Minty, D. W. Sweeney, and C. Karel, "An Algorithm for the Traveling Salesman Problem," *ORSA*, vol. 11, 979-989 (1963).
11. Marsh, J. D., "Scheduling Parallel Processors," dissertation, School of Industrial and Systems Engineering, Georgia Institute of Technology (1973).
12. Muntz, R. R., and E. G. Coffman, "Optimal Preemptive Scheduling on Two Processor Systems," *IEEE Transactions on Computers*, vol. 18, no. 11 (1969).

13. Muntz, R. R. and E. G. Coffman, "Preemptive Scheduling of Real-Time Tasks on Multiprocessor Systems," *Journal of the ACM*, vol. 17, no. 2 (1970).
14. Pritsker, A. A. B., L. J. Watters, and P. M. Wolfe, "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach," *Management Science*, vol. 16, no. 1, 93-108 (1969).
15. Redwine, C. N. and D. A. Wisner, "A Mixed Integer Programming Model for Scheduling Orders in a Steel Mill," *Journal of Optimization Theory and Applications*, vol. 14, no. 3, 305-318 (1974).
16. Ryser, H. J., Combinatorial Mathematics, John Wiley & Sons, New York, New York (1963).
17. Thangavelu, S. R., and C. M. Shetty, "Assembly Line Balancing by Zero-One Integer Programming," *AIIE Transactions*, vol. III, no. 1 (1971).
18. Tomlin, J. A., "An Improved Branch and Bound Method for Integer Programming," *ORSA*, vol. 19, no. 4, 1070-1075 (1971).

## VITA

Robert Lee Bulfin, Jr. was born October 21, 1946 in Atlanta, Georgia. He was graduated from Forest Park Senior High School in May of 1964. He entered the Georgia Institute of Technology in September 1964, and was awarded the Bachelor of Industrial Engineering in December 1968.

After graduation, Mr. Bulfin accepted a position as Industrial Engineer with Celanese Fibers Company in Rome, Georgia. He returned to the Georgia Institute of Technology in January 1970, and was awarded the Master of Science in March 1972. He received the Ph.D. degree from the School of Industrial and Systems Engineering in June 1975.

Mr. Bulfin is married to the former Lynn Glasgow, and has one son, Ben. He is currently an Assistant Professor in the Department of Systems and Industrial Engineering at the University of Arizona.