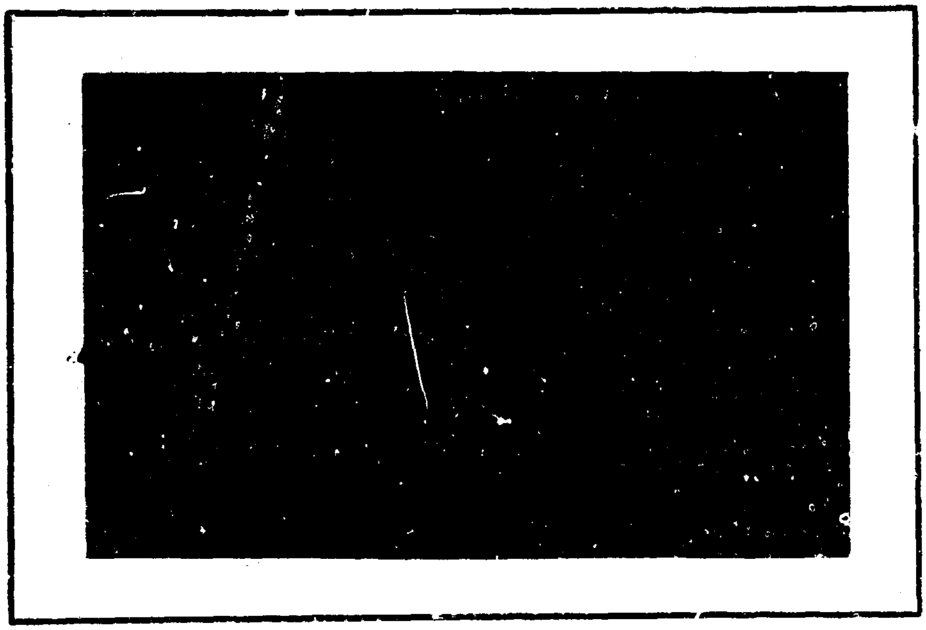
AD 673528

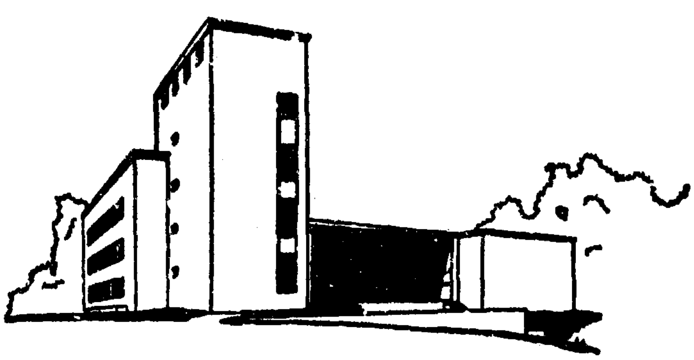


Carnegie-Mellon University

PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



213

Management Sciences Research Report No. 138

DECISION NETWORK PLANNING MODELS

by

Wallace B. S. Crowston*

May, 1968

* Carnegie-Mellon University

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under Contract NONR 760(24) NR 047-048 with the U. S. Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania

Carnegie Mellon University

Committee on Graduate Degrees in the Social Sciences
and
Industrial Administration

Dissertation

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Industrial Administration.

Title: DECISION NETWORK PLANNING MODELS

Presented by W. B. S. Crowston,

B.A.Sc. University of Toronto
(1956)

S.M. Massachusetts Institute of Technology
(1958)

M.Sc. Carnegie Institute of Technology
(1965)

Approved by the Dissertation Committee

Gerald L. Thompson
Chairman

May 29, 1968
Date

Approved by the Committee on Graduate Degrees in the Social Sciences and Industrial Administration

. . . .
Chairman

. . . .
Date

DECISION NETWORK PLANNING MODELS

by

Wallace Bruce Stewart Crowston

Submitted to the Graduate School of Industrial Administration on May 1, 1968, in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

ABSTRACT

This thesis develops project planning models that allow the possibility of specifying alternate ways of performing any of the jobs in the project. The "job alternatives" for any task may have different times, costs, resource requirements and possibly different precedence relations with other jobs in the project network. The problem is to select the particular way in which each job will be performed and schedule the resulting jobs so as to minimize the cost of the jobs plus the cost associated with the completion date of the project.

The problem of selecting the optimal job alternatives in networks with no resource constraints is formulated as an integer programming problem. One constraint is required for each set of job alternatives and one for each possible path in the original project network. Arguments are developed to show that a substantial number of the precedence constraints are redundant and may be eliminated. To accomplish this reduction in problem size an algorithm related to the critical path algorithm is developed to reduce each network to an equivalent network containing only job alternatives and maximal distances between them. Jobs with no alternative are eliminated.

Two branch and bound routines are then developed to solve the problem. One of these is tested on a series of problems and is shown to be efficient. An integer programming algorithm is developed to serve as a sub-routine in the branch and bound algorithms. It is fast in that it uses the critical path algorithm to solve problems.

When resource requirements are added to the tasks of the project, and the total availability of resource per period is constrained, the problem of scheduling the jobs so as to minimize completion date becomes extremely difficult. Nine heuristic routines for the loading problem

are developed and tested. Of these a serial loading rule, operating on a job list ordered by late start, with no job bumping, proves superior. Three methods of generating combinations of job alternatives to be loaded were examined. These were complete enumeration, pairwise interchange and multiple pairs interchange. None of the methods provided good solutions in reasonable amounts of time.

To show the generality of the planning model developed the integer programming formulation of the project problem was adapted to the m x n job-shop scheduling problem, the single product assembly-line balancing problem and the problem of planning projects under incentive contracts.

Thesis Supervisor: G. L. Thompson

ACKNOWLEDGMENT

In my first semester at Carnegie Institute I began my work for Professor G. L. Thompson on a series of job-shop and project scheduling problems. This association developed my interest in the research reported in this dissertation. I would like to take this opportunity to thank Professor Thompson for his guidance and for the support he has given me as my thesis advisor.

I would also like to thank the other members of my committee, Professors Kriebel and MacCrimmon for the valuable suggestions they offered. I would also like to acknowledge the contribution of my present colleagues, Professors Carroll and Pierce. Their advice, their encouragement and support directly assisted the completion of this work.

Financially the research was made possible by a doctoral dissertation grant from the Ford Foundation. However, the conclusions, opinions and other statements in this work are those of the author and are not necessarily those of the Foundation. Additional support was provided by the Management Science Research Group, Carnegie Institute of Technology, under Contract Nonr 760(24), NR 047-048 with the U.S. Office of Naval Research.

Computation was supported in part by Project MAC, an M.I.T. research program sponsored by the Advanced Research Projects Agency, Department of Defense, under Office of Naval Research Contract Number Nonr-4102(01). Additional support was given by the Graduate School of Industrial Administration, Carnegie Institute of Technology and the Alfred P. Sloan School of Management, M.I.T.

Finally, I would like to express my appreciation for the unlimited patience and cooperation of my wife, Taka. Without her support the dissertation could not have been completed at this time.

TABLE OF CONTENTS

CHAPTER | Page

I. Decision Network Planning Models. 1

II. A Review of Selected Planning Literature. 9

1. Introduction 9
2. Simple Time Constraints. 11
3. Simple Interdependency 13
4. Simple Resource Constraints. 13
5. Time and Interdependency Models. 15
6. Resource and Interdependency Constraints 23
7. Time and Resource Problems 25
8. Models with Resource, Time and Interdependency Constraints. 32
9. Discussion of Constraint Categories. 36
10. Solution of Planning Models. 37

III. Decision CPM Models 40

1. Introduction 40
2. The Mathematical Basis of DCPM 40
3. Decision Project Graphs. 45
4. Decision Graph Solution by Integer Programming 45

IV. Decision Network Reduction. 53

1. Introduction 53
2. Dominance Tests for Constraint Elimination 53
3. Implementation of Dominance Tests for Path Elimination. 56
4. Feasibility Tests for Path Elimination 59
5. Application of Dominance and Feasibility Tests to a DCPM Network 63
6. Lower Bound Calculation. 65
7. An Application of Lower Bound Calculation. 67

V. A Network Algorithm for Restricted Types of Integer Programs. 70

1. Introduction. 70
2. The Network Algorithm 70
3. A Numerical Example 78
4. Violations of the Assumptions 80
5. Application to a DCPM Problem 82

TABLE OF CONTENTS (Continued)

CHAPTER — Page

VI. Branch and Bound Algorithm for the DCPM problem. . . . 85

1. Introduction. 85
2. Reduced Constraint Algorithm. 86
3. Application of the Reduced Constraint Algorithm . 95
4. Fixed Order Algorithm 100
5. Application of the Fixed Order Algorithm. 104
6. Computational Results with the Fixed Order Algorithm 106

VII. Resource Constrained Decision Networks 112

1. Introduction. 112
2. Project Scheduling Heuristics 113
3. Experimentation with Project Scheduling Heuristics 117
4. Total Enumeration 119
5. Pairwise Interchange. 121
6. Multiple Pairs Switching. 124
7. Discussion of Results 125

VIII. Applications of the DCPM Model 128

1. Introduction. 128
2. The m x n Job-Shop Scheduling Problem 128
3. The Single Product Assembly-Line Balancing Problem 132
4. An Application of DCPM to Incentive Contracts. . 134

IX. Conclusion. 141

1. Summary and Conclusion 141
2. Suggestions for Future Research. 146

APPENDIX A: Problem Generation 148

APPENDIX B: Computational Results -- Reduced Network Algorithm . 156

APPENDIX C: Project Loading Heuristic Results. 160

APPENDIX D: Computational Results -- Resource Constrained Decision Network Problems. 164

APPENDIX E: Branch and Bound Program Description 184

BIBLIOGRAPHY. 194

LIST OF TABLES

TABLE		Page
IV-1	Calculations for Network Reduction	60
IV-2	Calculations for Network Reduction	61
IV-3	Reduced Network Matrix	62
IV-4	Reduced Constraint Set	64
IV-5	Computation of Lower Bound	69
VI-1	Reduced Constraint Matrix	87
VI-2	Reduced Network Matrix	100
VI-3	Computational Results for the Fixed Order Algorithm	111
VII-1	Number of Best Schedules	118
VII-2	Number of Unique Best Schedules	119
VII-3	Number of Worst Schedules	119
VII-4	Test Summary	127

LIST OF FIGURES

FIGURE		Page
I-1	Precedence Network.	2
I-2	Design Tree	5
III-1	Decision Graph.	46
V-1	Integer Programming Network	79
V-2	Integer Programming Network	79
V-3	Integer Programming Network	81
V-4	Decision Node Network	84
VI-1	Integer Programming Network	96
VI-2	Solution Tree Reduced Constraint Algorithm.	98
VI-3	Reduced Network	105
VI-4	Solution Tree Fixed Order Algorithm	107
VI-5	Solution Tree Fixed Order Algorithm	108
VII-1	Total Enumeration Tree.	122
VII-2	Pairwise Interchange Enumeration Tree	122
VIII-1	Fee, Point and Cost Relations	136
VIII-2	Incentive Contract Fee Structure.	137
VIII-3	Solution Summaries.	140

Chapter I

DECISION NETWORK PLANNING MODELS

The growth of interest in quantitative solutions to management problems has resulted in a rapid development of planning models, based on network representation of the activities to be performed. Process charts have been used to show basic work elements in a single task and the order in which they must be performed. Networks have been used to show the required job ordering in large construction projects and network based algorithms have been developed to find the total time required to complete such a project.

Although the applications of the models are at different levels of detail, they have many common characteristics. In each case there may be constraints in the problem that effect the "time" at which the individual planning units, either work elements or jobs may be performed. These may take the form of an explicit restriction that a job must start on a particular day or that the job cannot be started before a given day. Alternately, a job may be constrained not to start until some prior job is finished. For example, the cellar walls of a house cannot be constructed until the footings are laid. This second type of time constraints will be called "precedence" constraints. The graph of Figure I-1 shows a series of tasks S_1 to S_7 related by precedence constraints which are graphically illustrated by directed line segment. In a particular case, say $S_1 \longrightarrow S_2$, we imply that S_1 is a predecessor of S_2 and conversely S_2 is a successor of S_1. The nodes representing planning units and the directed line segments make up the network to which we have referred.

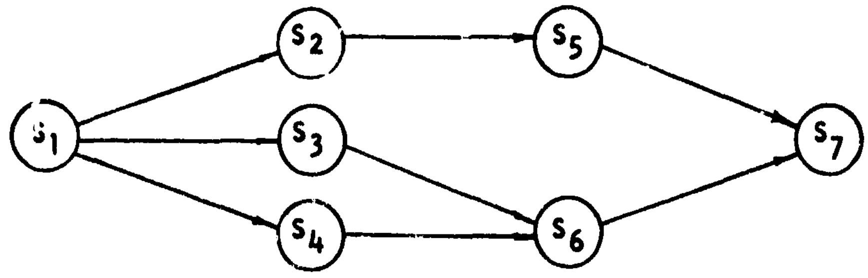


Figure 1-1

In addition to the precedence relations, planning units may be related by a mutual dependence on a limited resource. In a process chart of the man-machine variety, both the man and the machine are considered to be resources and they may be physically restricted to perform only one task at a time. Thus, if the man is required to perform both S_3 and S_4, even though no technological constraint exists between them, the jobs must be performed serially. That is, he must perform S_3 then S_4 of S_4 then S_3, but not both together. For problems of practical interest, the number of feasible sequences may be large and the problem of find the "best" sequence on all resources is a difficult combinatorial problem.

Finally, the planning units may be related technically by the nature of the project that is being performed. It is conceivable that

In a construction project there may be two methods for performing a particular job, with different costs and different performance times. For example, if wooden partitions are required, they may either be purchased in an assembled form and be quickly installed or they could be fabricated on the site by carpenters. We call this kind of mutually exclusive alternative a "job alternative" interdependency because we must choose between the two methods of performing the task. Many additional types of interdependency could exist between planning units. Perhaps if the partitions are pre-assembled, then a particular design for the electrical system is required. We will term all such relations which are not the "job alternative" interdependency described above, "other" interdependency.

This thesis will develop network models that include precedence constraints and the possibility of resource constraints. Each model that is discussed will include sets of mutually exclusive tasks, that is "job alternative" interdependencies or as we shall term them, "decision" nodes. In addition, "other" interdependencies may be imposed on the sets of mutually exclusive tasks. The functional setting of the models will be the (construction) project scheduling problem but, in fact, the theory developed would be applicable to a wide range of planning models.

In one chapter of the thesis, we will examine the problem of selecting from the sets of "job alternatives" the particular jobs we wish to perform. In terms of our original example, this might be the choice of pre-assembled partitions. If there is a large number of decision nodes in the network, there will be very many possible combinations of decision jobs that we can select. To evaluate a particular solution, that is, the choice of a particular set of decision jobs, one from each mutually exclusive set, we must evaluate the cost of the jobs plus the effect that the choice of these jobs have on the completion date and thus the completion cost of the project. We will call the choice of a set of decision jobs a "design" problem and the calculation of the minimum time for completion of the project, given a choice of decision jobs, an "operating" problem. Note that it is necessary to solve an "operating" problem to properly evaluate any "design" and that an optimal "design" is one which minimizes the sum of job cost and completion date cost.

The interaction of "design" and "operating" problems can be seen in many areas of planning. If we wish to establish warehouses in a manufacturer's distribution system, the "design" decision is the selection of the quantity, size and location of the warehouses. To evaluate such a "design" we must find the total cost of establishing the warehouses plus the minimum cost for "operating" the warehouses. The "operating" problem is the optimal allocation of customer demands to warehouses and warehouse demands to factories so as to minimize production, shipping and inventory costs.

The problem may be illustrated graphically with the following design problem. If we have two design variables, x and y, each having feasible levels 1, 2, 3, all possible designs are represented by paths in the tree of Figure 1-2. For each path or each possible design, it may be required that we solve an operating problem. In a facilities problem, for example, plant layout, this implies that we determine the best method of scheduling production for each possible

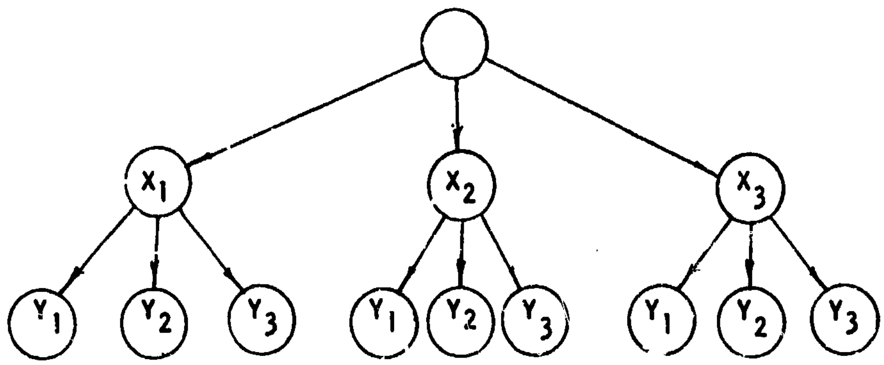


Figure 1-2

layout. These illustrations suggest that if problems have many discrete design variables or if the operating problem we must solve is a complex one, the determination of an overall optimum solution may be difficult. If methods can be found to reduce the number of designs that it is necessary to evaluate or if efficient methods can be found for solving the operating problems, then it will not be necessary to solve design problems sequentially. In this thesis, methods for the elimination of some designs and methods for solving operating problems in the area of project planning will be developed.

Chapter II contains a review of a wide variety of planning literature. The work is categorized by the particular sets of constraints that are to be found in the models. That is, we consider most combinations of precedence, resource and both "job alternative" and "other" types of interdependencies.

Chapter III develops a model for project planning that contains precedence constraints and both "job alternative" and "other" interdependency constraints. It is assumed that in the project there are a number of competing methods for performing some of the jobs, each method having a different cost, a different time, possibly different precedence relations with other jobs and different interdependencies with other jobs. All possible jobs are considered in the project graph and then in the scheduling phase the job alternatives that minimize total cost are selected. A numerical problem is introduced here that will be used to illustrate the material of Chapters III, IV and VI. In Chapter IV methods are developed to reduce the original decision network so that the problem may reasonably be solved with standard integer programming techniques.

In the solution of a decision network, it may be necessary to solve sub-problems that minimize the cost of the "design" selected with no regard to the cost of the "operating" problem, that is, the cost of the minimum completion date. Essentially, the sub-problem is to find the set of decision jobs that meets the "job alternative" and "other" interdependencies with minimum job cost. Chapter V develops an integer programming algorithm specifically for this problem. Chapter VI develops two branch and bound algorithms which solve for the best "design" given the cost of the decision jobs, the cost of the completion date and the interdependency constraints. They solve the "design" and the operating "problem" simultaneously. Computational results are given for one of these methods.

In Chapter VII we consider the full model, that is, project planning problems with precedence resource and interdependency constraints. To solve the "operating" problem in models having no limit on resource usage is relatively straight forward. It is simply a matter of calculating the length of the critical path and evaluating the cost of that finish date. When we add resource constraints, the problem of calculating minimum project length for any given design may be a very complex combinational problem. In fact, for large projects, given current techniques and reasonable limits computer time, it is not possible to find optimum or minimum length projects. For this reason, we develop and experimentally test several heuristic loading techniques. The best of these heuristics is then used as our "operating" rule to evaluate various designs. The designs to be tested are generated first by complete enumeration, then by pairwise interchange and finally by multiple pairs interchange.

As we have stated above, many planning problems can be represented by the combination of constraints we have discussed. To illustrate this point, in Chapter VIII the basic integer programming formulation of our decision network planning problem is used to formulate the job-shop scheduling problem and the assembly-line balancing problem. These formulations prove to be substantially more compact than competitive formulations of the problems. Finally, the model is adapted to projects with more complex criterion functions, specifically the cost structure of incentive contracts. Chapter IX contains a summary of the work, conclusions and recommendations for further research.

Several terms that are common in the literature of project scheduling, such as early start, will be used frequently in the following chapters. These terms are defined rigorously elsewhere [45, 49] so that we will review them only briefly here.

A "path" through a project network is connected sequence of nodes (jobs) and directed line segments extending from one node to some other. In Figure 1-1 we have a path from S_1 through S_2 and S_5 to S_7. The nodes S_1, S_4 and S_5 do not lie on a common path. The length of any path is simply the sum of the job times for all jobs on the path. We now define the "early finish" of a job to be the longest path from the first job in the network to the job under consideration. "Early start" time is simply early finish time of a job less its job time. The "critical path" of a project is the longest path from the first job in the network to the final job in the network and is equivalent to the minimum number of days required to complete the project.

The "latest start" date for a job is defined as the day on which the job must start if the project is to finish exactly on its due date. We may calculate the value of late start for a job by subtracting the length of the longest path from the job to the end of the project from the due date of the project. Thus a job can begin no earlier than the early start time because its predecessors must first be completed and no later than late start or it will delay the finish of the project beyond the due date. The difference between late start and early start time is defined as job "slack" time, the measure of permissible delay for a job. Other terms more uniquely related to the models to be discussed will be defined as required.

Chapter II

A REVIEW OF SELECTED PLANNING LITERATURE

The management planning literature, like many other areas of management activity, is susceptible to many possible categorizations. Perhaps the most obvious breakdown follows the functional area of an industrial concern and within this is a sub-grouping by problem area. Thus under the production heading we have extensive and largely independent literature growing up around the "job-shop problem" or the "assembly-line balancing problem". In finance we have the "capital budgeting of interrelated projects".

A second categorization might be by solution technique. Here is a list of functional area problems best solved by linear programming; this list requires integer linear programming and so on. This approach is closely related to a third categorization, the one to be used as the framework for our discussion. The third structure divides planning problems by the type of constraint found in the problem. To be explicit, we have defined in Chapter I time constraints, resource constraints and interdependency constraints as possible dimensions of planning problems. For simplicity we will not discuss the dimension "uncertainty", nor will we consider motivational and social problems of planning.

The possible combinations of the three dimensions and therefore our subheadings will be simple time constraints, simple resource limits, simple interdependency, problems with time and interdependency, with time and resource constraints, with precedence and interdependency and finally models with all three characteristics. To be included in any two or three dimension category the model must emphasize some interesting interaction of the relevant dimensions.

The following symbols will be used throughout this literature review and the rest of the thesis:

S_i an individual job or planning unit

$$d_i = \begin{cases} 1 & \text{if task } S_i \text{ is to be performed} \\ 0 & \text{otherwise} \end{cases}$$

$$v_i = \begin{cases} 1 & \text{if task } S_i \text{ is on the critical path} \\ 0 & \text{otherwise} \end{cases}$$

c_i the cost or revenue of task S_i

t_i the time required to perform task S_i

U_i the maximum length of task S_i if t_i is variable

L_i the minimum length of task S_i if t_i is variable

a_i the reduction in cost c_i per unit increase in t_i,

t_i the time to perform S_i

W_i the early start time of task S_i

W_f the early start time of S_f, an artificial FINISH job that is constrained to start after all other jobs in a project are finished

D the desired completion date or due date of a project

$$W_f^{+} = \begin{cases} W_f - D, & W_f - D > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$W_f^{-} = \begin{cases} D - W_f, & D - W_f > 0 \\ 0 & \text{otherwise} \end{cases}$$

k_i^r the usage per time period of resource r by job S_i

K^r	the availability per time period of resource r
P_i	the probability of job S_i occuring
S_s	an artificial START job that must be completed before any other task can be started

Simple Time Constraints

The basic PERT or CPM model [43, 44, 49] is taken as the example of simple time constraints. If job S_i is an immediate predecessor of job S_m this relation will be shown symbolically as $S_i \quad S_m$, graphically as

and in mathematical programming notation as $W_i + t_i \leqslant W_m$. An alternate graphical notation in common use represents the jobs as lines rather than as nodes and represents immediate predecessor relations by intersections of jobs rather than by directed line segments.

The length of the critical path in any network, say that of Figure I - 1, may be determined as the value of W_f in a solution to the following problem.

Minimize W_f

Subject to
$$
\begin{aligned}
W_1 + t_1 &\leqslant W_2\\
W_1 + t_1 &\leqslant W_3\\
W_1 + t_1 &\leqslant W_4\\
W_2 + t_2 &\leqslant W_5\\
W_3 + t_3 &\leqslant W_6\\
W_4 + t_4 &\leqslant W_6\\
W_5 + t_5 &\leqslant W_7
\end{aligned}
$$

$$W_6 + t_6 \leqslant W_7$$

$$W_7 + t_7 \leqslant W_f$$

Here we minimize the length of the project, subject to a set of time constraints, one for each immediate predecessor relation in the graph.

The dual problem formulated by Charnes and Cooper [15] defines a variable v_i for each job in the network. Then all jobs that have no predecessors are included in an equation of the form

$$V_1 = 1$$

to initiate an artificial flow of one unit into the network. For all other jobs, not including S_f, they constrain the flow in from the predecessors of the job (a maximum of 1 unit) to equal the flow out to immediate successors.

$$-V_3 - V_4 + V_6 = 0$$

For the final node they establish a sink for the one unit flow.

$$-V_5 - V_6 = -1$$

These constraints guarantee that a set of jobs will be chosen that will form a path through the network. Finally the criterion function is

$$\text{Maximize} \sum_{i=1}^{N} t_i \; V_i \;, \quad N = \text{total number of jobs in the project}$$

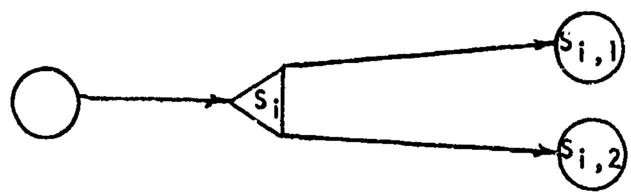
Thus we select the longest path in the network.

A third possible formulation would establish a constraint for each path in the network and constrain W_f to be longer than all paths. Since this approach implies complete knowledge of all paths, it would be a simple matter to select the longest path directly.

Simple Interdependency.

"Job alternative" interdependency has been defined as a set of mutually exclusive alternative planning units. Originally we considered the units to be alternative methods of performing some job, but we may also consider them different results of some stochastic process. A natural source of such outcomes would be a research or development project.

Eisner [10] proposes a network with decision nodes to represent such situations.



If we arrive at decision node S_i then outcome S_{i2} will occur with probability $P_{i,1}$ or $S_{i,2}$ will occur with probability $P_{i,2}$ where $P_{i,1} + P_{i,2} = 1$. Essentially he constructs a decision tree with time value and job name labelling. Then using standard probability calculations he obtains the probability of various network outcomes and attaches to them the sum of the times from the relevant path. We categorize this as simple interdependency because there is no interaction between times and probabilities.

Other work that can be classed as pure interdependency emphasizes solution techniques for 0,1 combinatorial problems rather than the application of techniques to planning problems. Examples of such articles are references [62, 68].

Simple Resource Constraints.

The general knapsack problem may be interpreted as an example of a simple resource constraint. We have a number of available "planning units" which may be selected, each using some amount of a scarce resource, or resources. The object is to select the set of these units which will optimize a linear criterion function, subject to constraints in the amount of each available resource. In our notation this may be written

$$\text{Max} \quad \sum_{i=1}^{m} c_i \; d_i$$

$$\text{Subject to} \quad \sum_{i=1}^{m} k_i^r \, d_i \leqslant K^r \quad r = 1, 2, \ldots R$$

$$0 \leqslant d_i \leqslant 1 \quad \text{integer}$$

Here again $d_i = 1$ implies that a planning unit is selected (for the knapsack) and $d_i = 0$ implies that is is rejected. C_i is the value to us of unit S_i.

In his discussion of this problem, Dantzig [24] points out that linear programming solutions of this problem give values of the d_i which will not all be 0-1 but instead have fractional values. If an integer solution is required, he suggests that rounding is usually "good enough" for most practical problems. For exact integer solutions in problems of one constraint, he suggests the dynamic programming approach of Bellman [5]. If problems have two or more constraints, he suggests the use of linear programming with constraints added to eliminate fractional extreme points. More recently Glover [40] and Weingartner and Ness [74] have developed truncated enumeration methods for the solution of this problem. More generally the large number of integer programming routines now available may be applied to this problem.

Weingartner [75, 76] shows that the Lorie-Savage [52] capital budgeting problem is essentially the knapsack problem as described above. The cutting stock problem has been formulated by Gilmore and Gomory [37] and Pierce [61] as a form of the knapsack problem. To begin we will define a resource as one of the possible customer order widths. Thus given customer orders for K^r units of item r we must schedule our cuts so as to produce at least K^r units. The planning unit, S_i, is one pattern of cuts that may be made from a stock roll. Thus we must enumerate all possible combinations of order sizes that might be taken from each stock roll and define each different combination as a separate unit S_i -- for, say, a total of m units. Then, if we wish to minimize the stock rolls used and yet meet customer requirements, we can solve the problem.

$$\text{Min.} \sum_{i=1}^{m} d_i$$

$$\text{Subject to} \quad \sum_{i\ 1}^{m} k_i^r \ d_i \leqslant K^r \quad r = 1, \ldots, R$$

$$0 \leqslant d_i \leqslant 1 \text{ integer}$$

Gilmore and Gomory, and Pierce have developed special techniques for the solution of large cutting stock problems.

Time and Interdependency Models.

The articles to be discussed in this section are typical of the general literature in the class of interdependency included. In terms of our definitions, "job alternative" interdependency is more common than "other" interdependency. It is true, however, that in the models with job alternatives generated by probabilistic research or production outcomes, a job whose sole predecessor may not be performed may be considered to be contingent on that predecessor. This, for example, is true of the article by Eisner [28] discussed above.

Elmaghraby [29] introduces a series of logical relations to standard network formulations. Converting his work to our notation, we have a planning unit or task S_i with a probability of occurrence and a vector of parameters such as time, cost, etc., attached to it. We now define logical relations that may exist between the planning units.

1. "and" or logical intersection of two events, propositions or activities. For example, unit S_3 will occur if both events S_1 and S_2 occur

$$d_3 \leqslant bd_1 + (1-b)d_2$$

$$0 < b < 1$$

$$d_i \quad \text{0-1 integer}$$

2. "inclusive-or" i.e., the union of two or more propositions. Node S_3 will occur if S_1 or S_2 or both occur

$$d_3 \leqslant d_1 + d_2$$

3. "exclusive or", often referred to as the ring sum. S_3 occurs if either S_1 or S_2 but not both occur.

$$d_1 + d_2 + d_3 = 2$$

4. "decision" node, i.e., a node at which the system may transfer along one path or the other with known probabilities. Elmaghraby uses this node solely for non-deterministic branches.

$$d_2 + d_3 \leqslant 1$$

Note that relations 1, 2, and 3 would be included in our definition of "other" interdependency and 4 would be considered as "job alternative" interdependency. A graphical symbol is defined for each interdependency relation so that a set of tasks and the relations between them may be expressed graphically. However, if the nodes are given performance times and the $\bigcirc\!\!\longrightarrow\!\!\bigcirc$ relation is interpreted as an immediate predecessor relationship, then it is only possible to have interdependency relations between units that also have immediate predecessor-successor relations unless new symbols are defined. This is an unnecessary restriction introduced by his attempt to show all relations graphically. As we shall see, a programming formulation has no such restriction.

Given the model as described above, Elmaghraby suggests a complete enumeration of paths and shows algebraically that for each such path a time and probability of occurrence may be determined. He concludes by combining path time and probability information for an overall expected value for project completion.

The problem of determining a project cost function, that is, total cost at various completion dates, for the deterministic case is discussed by Kelley and Walker [43]. The length of the project may vary because for each job there is a series of "job alternatives" (actually a continuous linear function) with increasing cost, c_i, and decreasing time, t_i. The operation time for job S_i is constrained to be within the upper time limit U_i and the lower limit L_i, that is

$$0 \leqslant L_i \leqslant t_i \leqslant U_i$$

and cost $C_i = b_i - a_i t_i$

where b_i is the cost of job S_i performed in time L_i

and a_i is the cost of decreasing t_i by one time unit.

Then given an absolute due date, D , the objective function is

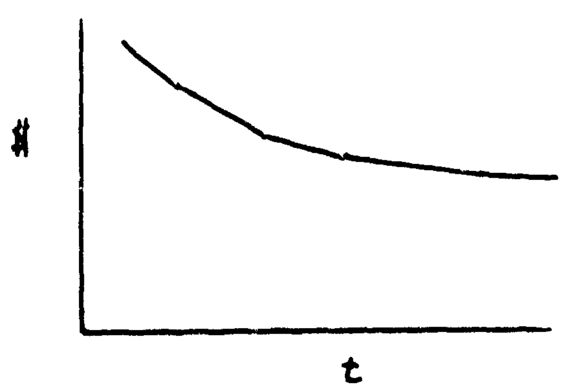
$$\text{Min} \sum_{i=1}^{m} b_i - a_i t_i$$

Subject to precedence constraints, one for each link in the graph

$$W_i + t_i \leqslant W_m, \quad S_i \longrightarrow S_m$$

and finally $\quad W_f \leqslant D$

Kelley refers to a form of the Ford-Fulkerson algorithm [33] for an efficient solution to the problem. Fulkerson [35] presents a similar algorithm in full detail. Essentially, he interprets the problem as one of network flow and solves the dual of this problem. The algorithm begins by setting all jobs at their cheapest (longest) value and calculating the project length that results, setting D, (due date) equal to this value, and calculating the cost, $P(D)$. Then D is decreased and a new value of $P(D)$ as well as W_i, t_i for all jobs is calculated. The process is continued until a shortest feasible length for D is obtained. It is then possible to plot the project time-cost curves.



Fulkerson points out that the breakpoints of this piecewise linear function occur at integer values of time if the bounds on job lengths are integers. The algorithm has been shown to be efficient in practical applications to large problems [60]. In this formulation it may be assumed that the variation in job time and cost is a result of the application of more or less resources to the job. Since there is no attempt to restrict the amount of resource used at a particular point in time, by all jobs, the model is not considered to have resource constraints.

In some instances it may be unrealistic to assume a continuous linear relation between time and cost for a task in a project. For example, if a job may be only performed by an eight man crew on regular time, or by an eight man crew on regular time plus two hours overtime, there are two discrete ways to perform the job (two "job alternatives") and linear combinations of these methods may be technologically or contractually infeasible. The problem of discrete "job alternatives" in the time/cost problem is discussed in references [19, 26, 57, 58]. Moder and Philips [58] summarize some work by Meyer and Shaffer [57] on this problem. Their formulation is as follows:

$$\text{Min.} \sum_{i\,j} c_{ij}\, d_{ij}$$

S.t. precedence

$$W_i + t_i \leqslant W_j, \quad S_i \longrightarrow S_j$$

or if a job alternative situation is involved

$$W_i + t_{i1}\, d_{i1} + t_{i2}\, d_{i2} \cdots + t_{ik(i)}\, d_{ik(i)} \leqslant W_j$$

$$\text{and} \quad W_F \leqslant 0$$

where

$$d_{ij} = \begin{cases} 1 & \text{if job } S_{ij} \text{ is performed} \\ 0 & \text{otherwise} \end{cases}$$

This assumes that each job alternative

$$S_{ij}, \quad j=1 \ldots .k(i)$$

has identical precedence and successor relations.

Finally, we require an interdependence constraint

$$\sum_{j\ 1}^{k(i)} d_{ij} = 1$$

$$0 \leqslant d_{ij} \leqslant 1 \text{ integer}$$

A more general and more efficient integer programming formulation is presented by Crowston and Thompson [19], 1967. This model will be presented in detail later, but a short summary is included here. The job alternatives are again represented by 0-1 variables, d_{ij}, which for any particular job, are constrained

$$\sum_{j=1}^{k(i)} d_{ij} = 1$$

In addition, however, any set of alternative interdependence constraints may be written on these variables.

$$d_{ij} + d_{mn} \leqslant 1$$

$$d_{ij} \leqslant d_{mn}$$

$$d_{ij} < d_{mn}$$

etc.

Note that although resource constraints are not considered specifically here, alternative interdependency constraints could be written to constrain the total usage of a consumable resource over the life of the project.

The length of the critical path as defined in Chapter I is constrained by a set of equations which represent paths in the network. Paths which cannot become critical may be dropped from the problem. Rather than generate a time/cost curve, they establish a due date, D, with overtime penalty and undertime premium and solve directly for the optimum set of jobs to be performed. It would be possible, however, to solve a series of problems, setting progressively tighter upper limits on the critical path ($W_F \leqslant D$) and thus generate a time/cost curve.

The article also gives an outline of a heuristic technique for solving problems with time constraints and the "job alternative" type of interdependency. It is assumed that all alternatives for a given job have identical predecessor - successor relations and that the following inequalities hold:

$$t_{i,1} < t_{i,2} \ldots . < t_{i,k(i)}$$

$$c_{i,1} > c_{i,2} \ldots > c_{i,k(i)}$$

The routine is described as follows:*

1. Technologically order the jobs

2. Set each decision node to the alternative having lowest cost.

3. Calculate the critical path.

4. Reorder by Early Start

---

* Crowston and Thompson [19], pp. 20-21.

5. Go to 7
6. Recalculate the critical path starting at the position in the ordered job list held by the decision node of step (10)
7. Identify all decision nodes in the critical path
8. For all the nodes of step (7) calculate the net reduction in total project cost achieved by substituting the more costly alternatives
9. If no alternative reduces overall cost, go to (12)
10. Find the alternative that gives the maximum cost reduction and switch the relevant decision node to that alternative
11. Go to step (6)
12. Review all decision nodes that were previously chosen to see if sufficient slack has been generated to allow the reintroduction of a longer but cheaper alternative. If no such opportunity exists, go to (14)
13. Introduce the cheaper alternative found in step (12). Go to Step (12)
14. HALT

The two small problems tested by this routine gave the optimum solution although, as the authors state, it will not always do so.

A routine originally published in the D.O.D. and N.A.S.A. Guide PERT Cost [26] and in Alpert and Orkand [2], 1962, and extended in Moder and Phillips follows a somewhat similar routine.* At step (8), however, the replacement job chosen is the one with minimum incremental cost per day. If job S_{ij} were originally chosen, the measure would be

---

* Moder and Philips [58], pp. 109-122.

$$\frac{c_{im} - c_{ij}}{t_{ij} - t_{im}}$$ where S_{im} is the alterna-

tive being considered. However, if this would cause the critical path to shift, then, rather than use the incremental cost criterion, they choose the replacement job so as to minimize $[c_{im} - c_{ij}]$. Each switch is followed by a review of previously selected jobs to see if sufficient slack has been generated to allow a switch back to an original, cheaper job. This is similar to steps (12) and (13) of Crowston-Thompson. As the process continues, the cost of jobs chosen increases and the project length decreases, mapping out a time/cost curve, but not necessarily the optimum one.

Resource and Interdependency Constraints

Models in this category are essentially knapsack problems with interdependency constraints added. For example, Weingartner [75, 76] adds both "job alternative" and "other" types of interdependencies to the Lorie-Savage capital budgeting problem. The resource constraints are budget limits on the capital expenditure by period. Thus in each period we sum the capital requirements of the projects to be operating in that period and constrain the total amount to be less than the budget limit. In our notation his model is

$$\text{Maximize} \quad \sum_{i=1}^{m} C_i \, d_i$$

where C_i is the net present value of project S_i

Subject to $\sum_{i=1}^{m} k_i^r d_i \leqslant K^r$ $r = 1.....m$

where K^r is the budget limit in period r and k_i^r is the cash used by project S_i in period r. The d_i are again 0-1 variables. Any linear constraints in the variables d_i may be written to express interdependency conditions. The model may then be solved by integer linear programming.

In the later paper [75] Weingartner adds to this model a time dimension by allowing a given project to be represented by a mutually exclusive set of projects (job alternatives), one beginning at each feasible starting day of the project $[S_{i,t}, S_{i,t+1}, S_{i,t+2} .....]$. This follows the practice of Marglin [56]. Even though time is introduced into the model, no explicit provision is made for time precedence constraints, a natural dimension of the capital budgeting problem. In this article a new and reportedly efficient algorithm based on the dynamic programming solution to the knapsack problem is presented. Unfortunately this method will not handle the full range of possible interdependencies due to a restriction on the inclusion of negative variables.

A very similar problem is discussed by Root [63] and termed the "selection problem". Explicitly, the problem he wishes to solve is

Minimize $\sum_{i=1}^{m} c_i d_i$

Subject to $\sum_{i=1}^{m} k_i^r d_i \leqslant K^r$ $r=1....R$

where the K^r are assumed to be integer and a set of linear interdependency constraints are written in the variables d_i $i=1,\ldots,m$. For example, the fact that a job may be performed by several resources or resources in combination may be written as

$$\sum_{j=1}^{k(i)} d_{ij} = 1$$

where the k_{ij}^r would differ for the various alternatives. Root solves the problem by applying a theorem from symbolic logic to reduce the total set of possible solutions. Then by costing each remaining solution, he can select the one with minimum cost.

Time and Resource Problems

Many important scheduling problems may be described as time and resource constraint problems. Among these are forms of the project scheduling problem, the job-shop scheduling problem, and the assembly-line balancing problem. The project scheduling problem, of course, appears in contexts such as marketing and economic planning [31] as well as production. All of these problems have been formulated as integer linear programming problems but because of the high number of constraints involved, these are not suggested as possible solution methods for real problems. For example, Wiest [80] estimates that a project with 55 jobs in 4 shops with a time span of 30 days would have some 5,275 equations and 1,650 variables, not including slack variables or constraints added to assure an integer solution. As a result, heuristic solution methods have been developed for these problems. The essential problem is that the level of resources is constrained by period and the jobs, given the usual time constraints, may shift through time. Thus, in programming formulations, it is necessary to include the possibility that the job may shift through time, and this requires many variables and many constraints. Heuristic solution techniques can handle this problem with concise bookkeeping techniques. The solutions, however, are not necessarily optimal.

We will examine several heuristic approaches to the resource-levelling problem in project scheduling. It is assumed that in this problem the resources are not fixed but that the criterion function is so related to period by period resource levels that we are motivated to smooth the daily resource usage. Burgess and Killibrew [12] describe an iterative procedure that attempts to minimize the sum of squares of daily resource usages. This criterion, while minimizing the standard deviation from the project mean, still might allow a high peak in any given time period.

The routine first technologically orders the jobs by Early Start, and if jobs are tied, ranks the shortest first. In the second stage the jobs are loaded, beginning at the bottom of the technological list. Each job is scheduled as late as possible, subject to the condition that the daily usage should not be too far above or below the predetermined average daily resource usage. The late start of each job is strictly set by the assigned start time of its successors. The cycle is repeated, each time attempting to reduce deviations from the mean, until no further improvements are made. The best schedule is then chosen.

The method of Levy, Thompson and Wiest [50] concentrates on reducing the peak usage of the project resource. Heuristics which drive the solution to this goal would also work to meet the Burgess criterion. The jobs from all projects to be simultaneously scheduled are scheduled at Early Start and then the daily demand for each resource is plotted. For the first resource an initial trigger level is set, one unit below the maximum usage. It is assumed that the resources are ordered based on some priority system, perhaps daily cost per unit. Then all the jobs contributing to the particular peak and in addition having enough slack so that they may be scheduled beyond the peak are listed. Then one of the jobs is chosen probabilistically, the probabilistic weight being proportional to the job's slack time, and the job is shifted a random amount within the slack, forward. The resource peak again is calculated and a lower trigger level set. This process continues for the first resource until no further improvement is realized. Then freezing the lowest feasible limit for the first resource, the procedure is repeated for the second resource and so on through all resources. Now the jobs from each project are segregated and again an attempt is made to shift them and reduce the aggregate trigger level for all projects. Finally, when no further improvement is possible, the final schedule and trigger levels are stored and the process is completely repeated. Due to the probabilistic element in the decision rule, a new schedule will result. After several repetitions, the best schedule of those generated can be chosen.

The next problem class to be discussed will be scheduling to meet stated resource constraints. The early solutions to problems of this type were obtained with Gantt charts and their use in job-shop scheduling continues. The resource constraint in the job-shop problem will be the limit on machine availability and the time constraints are provided by technological ordering of operations on a particular work order. The criterion function may be to minimize completion time of the whole job file as in the integer programming formulations of Bowman [8], Manne [54], and Wagner [73]. Alternately, it might be the minimization of idle machine time (identical to minimum final completion time for the fixed job file case) as in Conway and Maxwell [18] or a complex function of order delay cost as in Carroll [14].

Many heuristic approaches have been developed for this problem [18, 20, 38] and in many of these approaches similar decision rules are used singly or in probabilistic combination to dispatch jobs from a queue to the machine. We will quote from a description of several such rules: *

1. SIO (shortest imminent operation)
   When a facility is available, select that item in the queue which has the shortest machining time on the facility.
2. LRT (longest remaining time)
   Select the item which has the most total machining time remaining.
3. J.S. (job slack per operation remaining)

---

* Crowston, Glover, Thompson and Trawick [20], p. 2.

Subtract the total remaining machine time for a given job from an arbitrary finish date, DD. Divide this slack by the number of operations remaining. Select the item with minimum job slack per operation.

4. LIO (longest imminent operation)

5. FIFO (first in, first out)

   Select the item that arrived first in the queue.

6. MS (machine slack)

   For each machine calculate the total machining time remaining. Select the job in the queue that goes to the most heavily laden machine next. Break ties with S.I.O.

Many other rules have been attempted to specifically meet more complicated criterion than the minimization of overall completion time. It will be interesting to note below that attempts have been made to apply several of the simple rules to the project scheduling problem, but that more complex rules have not as yet been translated to the project problem.

Two main approaches have been suggested for solving the fixed resource problem in the project scheduling context. Neither approach guarantees an optimum solution, nor consistently outperforms the other. Representative of the first approach is an article by Kelley [44]. The routine may be summarized as follows:

1. Technologically order the jobs, calculate early start, late start, and slack. Within the technological ordering, reorder by slack, lowest slack first.

2. Start with the first activity and continue down the list.

(a) Find the early start of the activity.

(b) Schedule the activity at early start if sufficient resources are available. If resources are not available, two alternative routines are suggested c1, c2.

(c1) Serial Method: Begin the job at the earliest time that resources are available to work it for one day. The job may be split if necessary.

(c2) Parallel Method: Find the set of all jobs causing the resource violation and rank them by total slack. Delay a sufficient number of jobs with the highest slack so that those remaining on the list can be scheduled.

3. Repeat step 2 for all jobs.

4. Repeat the whole routine with various orderings on the technological list.

In addition to the possibility of allowing job-splits, Kelley suggests that we also allow resource limits to be violated by small amounts. The basic heuristic in the original job ordering and in the parallel loading technique is the use of job-slack (JS) as a criterion for shifting jobs forward. This is a reasonable approach similar to that used by Levy [50].

The second approach, detailed in Moder and Philips [58] uses Maximum Remaining Path Length, or equivalently Late Start, as a basis for delaying jobs (LRT). Note that if two jobs have a common early start-time, both measures are equivalent. That is, the job delayed because of a higher slack would in the same way be delayed because of a lower Late Start time. Some other features of the routine are different than that suggested by Kelley. A main feature is a list of unscheduled jobs whose predecessors have all been scheduled, ordered by Late Start, and an ordered list of finish times of scheduled jobs. Thus the routine steps through time to only those days when a change of resource usage is possible. On such a day it examines the list of available jobs and ends either when the resources are exhausted or the job file is completed. At this point it again jumps ahead. As we implied at the beginning of this section, examples can be constructed to favor or penalize either of these approaches.

Finally, we will show that Salveson's [66] formulation of the assembly-line balancing problem has the structure of a time-resources constraint problem. The resources, in this instance, are the work stations and the work times at the station will be the resource level. Then, if a precedence ordering exists between jobs, as would usually be the case, a job may not be assigned to a work station unless all its predecessors have previously been assigned to that work station or earlier work stations. In addition to the above statement, the problem may be complicated by any of the interdependencies we have discussed. For example, it may not be feasible to do a particular pair of jobs at one station, and therefore, we must constrain the problem to prohibit this. Also, it should be pointed out that "alternative interdependency" relations would probably be the most efficient method of actually formulating the problem. Finally, as in the other problems of this section, it must be noted that practical problems must be solved by heuristic methods [72].

Models with Resource, Time and Interdependency Constraints.

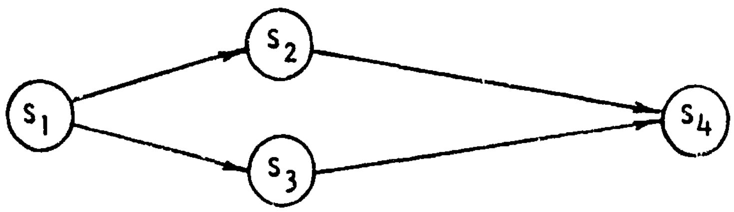
Problems in this category abound in the industrial world. Wherever there is an element of physical design connected with a scheduling problem, then interdependency is an implicit part of the problem. For example, we may consider the problem of an industrial engineer attempting to design a job with the graphical technique of man-machine analysis [4]. There are several alternate ways for an operator to perform each necessary function and operations when combined may require less time than the sum of the same operations performed singly. Finally, many possible orderings may be possible, although some technological ordering constraints are involved in many problems. Each production alternative may have an important effect on the scheduling problem. Similarly, design of construction projects will have important scheduling implications.

With the exception of the Weingartner article [75] which does not fully include the possibility of time ordering, no model was found to include resource, time and both types of interdependency constraints. It is true, however, that the job-shop models of Bowman [8] and Wagner [73] could be easily generalized to include this possibility. The use of job alternative interdependency in heuristic programs is not uncommon.

Two master's theses written in the Sloan School of Management, M.I.T., expand the usual statement of the job-shop scheduling problem to include the possibility of job alternatives, with two different practical interpretations of what these alternatives might be. First, a thesis by Russo [65] reports on a simulation of a job-shop with the possibility of alternate job routing. In the usual simulation we assume for a given job that the order of operation isstrictly determined, that is

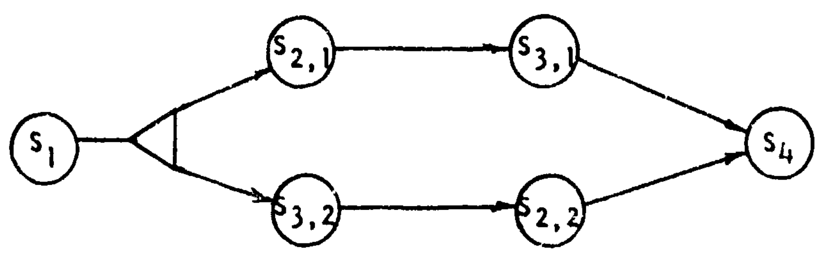
He suggests that in some instances the ordering would be

a.



and interprets and solves this as a case of

b.



$$d_{2,1} + d_{2,2} = 1$$

$$d_{3,1} + d_{3,2} = 1$$

$$d_{2,1} = d_{3,1}$$

It is clear that this situation can be categorized as a time and resource model, however, since his heuristics specifically detailed each alternative route, then calculated information for local dispatching rules based on the two alternatives and then decided between them, it is included here. One of the most effective approaches he discusses, allows all jobs in an alternate chain to enter queues for their respective machines when the predecessor of the alternate chain is completed. Then in each queue the jobs from the alternate chain are ranked by a standard dispatching rule. He then allows that job to go first, which is selected by the machine queue discipline.

Clermont [17] allowed for the possibility of several machines performing a given job. This more clearly resembles our job alternative interdependency case. His results show that the heuristic Russo found so successful only improved performance of the simple dispatching rules when the total machine loadings were highly imbalanced. In the case of balanced loads, switching actually decreased the performance of simple decision. For the balanced case only the simple switching heuristic "switch if the alternate queue is empty" consistently improved performance of the simple dispatching rules. His results also indicate that a dispatching rule "Covert" derived by Carroll [14] completely dominates all conventional rules tested.

Job alternative interdependency is also found in scheduling techniques designed for the large project problem. The series of SPAR programs by Wiest [80] will now be discussed. With each job he associates three operating levels, maximum crew size, normal crew size and minimum crew size. Of course, the length is inversely a function of resource level. The jobs originally at normal crew size are originally ordered by early start time, one possible technological order. Then, as in the Moder and Philip routine discussed above, a sub-list is generated and continually updated of jobs available for scheduling. From this list jobs are selected for scheduling with a probability inversely related to the available slack of the job. If a job is selected to be scheduled and the resources are unavailable, it is left to be schedule in a subsequent period.

Several subsidiary heuristic routines operate within the basic framework outlined above. If the slack of a job is low, an attempt is made to schedule it at maximum resource usage. If the resources are not available for this, a subroutine attempts to borrow resources from jobs operating on that day with normal or maximum resource. A second approach is to find jobs using the tight resource and delay their start for one or more periods. This frees their resources for the critical jobs. Finally, if all else fails, the critical job would be delayed one period. After the application of these and other routines, a final schedule is produced. Since there is a random element in the choice of jobs to schedule, it is suggested that the process be repeated several times and the best schedule selected.

An important addition to the program is a search routine that progressively shifts the level of initial resources from solution to solution in attempt to find those limits that will minimize the sum of resource cost and project completion time cost.

Discussion of Constraint Categories

As we have seen the categories chosen do not allow a categorization of all planning models. In several cases a good argument could be made against the allocation we have decided on. Nevertheless, for several reasons, this particular categorization is useful. It does suggest that more complex models may have relevance in functional areas where they have not yet appeared. For example, time precedence constraints would seem relevant to the capital budgeting problem, further use of alternative interdependencies in the job-shop problem and the assembly-line problem.

An even more fruitful use of such a framework will be to suggest that researchers examine a broad range of literature in their search for suitable solution techniques. Examples of this certainly have already occurred. For example, Wiest [80] uses the Bowman [8] job-shop L.P. formulation as the base of his project scheduling formulation. A thesis by Knight [46] attempts to relate heuristics in the job-shop problem to heuristics in the project scheduling problem. Finally, Wilson [82] shows how several line balancing algorithms may be applied to resource leveling. This comparison also allows us to make some general statements about the efficiency of various solution techniques.

Solution of Planning Models

For several categories of problems, algorithms have been developed that are much more efficient than any of the existing programming routines. For example, the longest path algorithm very quickly determines the critical path of a simple time constraint problem. Similarly, the Ford-Fulkerson technique efficiently finds the optimum job lengths and job start times for a project given a fixed due date and a bounded inverse linear relation between job cost and job time. This, of course, is a special case of the time-interdependency constraint case.

There is a second set of problems for which special efficient solution techniques have been developed, but because of the nature of the problem, the techniques may be considered as restricted integer programming routines. In this category I would include the truncated enumeration methods of Weingartner and Ness and that of Glover applied to the simple resource (or knapsack) problem. Also Root's algorithm for the selection problem, which is a combined resource-interdependency problem, is in this group. We may summarize the above cases, then, by saying that we may obtain optimum solutions more efficiently with existing techniques than with programming methods.

We now shift in the spectrum to those problems for which optimum solutions are only available through programming methods. It should be emphasized that no clear line can be drawn between problems in this group and those in the previous one. An example of this would be problems with pure interdependency constraints. We have stated above that certain simple resource or resource-interdependency constraint problems may be solved with special computational methods. On the other hand, if many resources were involved, it would be necessary to turn to conventional integer programming techniques. Similarly, for many large problems with pure interdependency constraints, the 0-1 tree search algorithms would give the most efficient solutions. Finally, the time-interdependency problem, when the job alternatives are discrete, is best solved by conventional integer programming routines [70]. Techniques are available, however, for this problem, as we will show later, that will significantly reduce the number of required constraints and the number of variables. In this regard, it is interesting to observe that for integer programming problems in 0-1 variables, added interdependency constraints, by eliminating branches in the feasible solution tree, actually make problems easier to solve.

Heuristic routines are available for the above problems, and we have covered several approaches to the time-interdependency problem, given discrete jobs, no alternative interdependency and common precedence-successor relations for job alternatives. It can be shown that existing heuristic methods are not at all appropriate for problems with any reasonable alternative interdependency complications.

The final category finds combinations of time and resource constraints and, as we have discussed, this grouping requires large numbers of constraints and variables for programming solutions. As a result, these problems are solved almost exclusively in practice by heuristic methods. As noted above some of the simpler job-shop problem heuristics have been adapted to the project scheduling problem with some success. This suggests that more complex heuristics, based on variations of successful job-shop rules, should now be tested. When job alternative interdependencies are added to the problem, as Wiest [80] does, it is possible to build subsidiary switching routines based on local (in the time dimension) resource usage information. If alternative interdependencies were added to the problem, such local information might no longer be a sufficient base for a switching decision. In fact, as we have seen, any purely heuristic method might have difficulty approaching an optimum solution to this kind of problem.

Chapter III

DECISION CPM MODELS

The paper that will serve as the basis* for this chapter is that of reference [19] by Crowston and Thompson. This decision network formulation contains "time" constraints and both "job alternative" and "other" types of interdependency. The mathematical basis of Decision CPM will be discussed as well as several alternate integer programming formulations of the problem. A numerical problem is introduced in this chapter (FIGURE III-1) that will also serve as an example in Chapters IV and VI.

2. The Mathematical Basis of Decision CPM

This section will follow in part the article [2] by Levy, Thompson, and Wiest. Let $J = \{S_1, S_2, S_3 \ldots\}$ be a set of job sets that must be done to complete a project. Some job sets are unit sets $S_i = \{s_{i1}\}$ and other job sets have several members, $S_i = \{s_{i1}, s_{i2}, s_{i3}, \ldots\}$. In order to complete the project, one of the jobs from each job set must be completed. Associate with each job set

(1) $$S_i = \{s_{i1}, \ldots, s_{ik(i)}\}$$

$k(i)$ variables

---

*Sections 2, 3, 4 and 5 are taken essentially verbatim from [19] although a new numerical problem is introduced. Furthermore, this chapter will assume that exactly one job alternative is selected from each job set, that is

$$\sum_{j=1}^{k(i)} d_{ij} = 1$$

(2) $$d_{i1}, \ldots, d_{ik(i)}$$

having the property that

(3) $$d_{ij} = \begin{cases} 1 & \text{if job } S_{ij} \text{ is to be performed} \\ 0 & \text{otherwise} \end{cases}$$

Since exactly one of the jobs must be performed, then the <u>mutually exclusive</u> or job alternative interdependence condition is expressed by

(4) $$\sum_{j=1}^{k(i)} d_{ij} = 1$$

If all job sets are unit sets (implying condition (4) holds), then all of the jobs in the project are independent and the project reduces to the ordinary project of the usual CPM variety. If one or more of the job sets have more than one member, then for each such set a decision must be made as to which job of the set is to be done. Once such a decision is made for each job set, the result is an ordinary CPM project.

It should be noted that the decisions may be complicated by many other kinds of conditions than (4), which may be of the mutually exclusive or contingent kind. For instance, the following equations give examples of such interdependencies among decisions.

(a) $d_{ij} + d_{mn} = 1$

(b) $d_{ij} \leq d_{mn}$

(c) $d_{ij} = d_{mn}$

Note that (a) says that we cannot do both S_{ij} and S_{mn} ; (b) says that we can only do S_{ij} if we also do S_{mn} ; and (c) says that we either do S_{ij} and S_{mn} or we do neither. The above discussion illustrates some of the possible complexity of problem formulation that is possible within the Decision CPM framework.

In addition to the relations described above, there will be precedence relations between the jobs of a decision project. Let '$\ll$' denote a relation between pairs of jobs in J such that $S_{ij} \ll S_{mn}$ is defined for some pair of jobs S_{ij}, S_{mn} and is read S_{ij} is an <u>immediate predecessor</u> of S_{mn}. The interpretation of this statement is that all immediate predecessors of a job must be completed before that job can be started. A decision project is the set J together with the specified interdependencies and the relation $\ll$ defined on J.

The decision project graph of a project, G, is a planar graph with nodes representing jobs and a directed line segment, connecting two nodes S_{ij}, S_{mn} if and only if $S_{ij} \ll S_{mn}$ holds. A <u>path</u> in G is a set of nodes connected by immediate predecessor relations. A cycle in G is a closed path of the form $S_{ij} = a_1 \ll a_2 \ll \ldots \ll a_n = a_1 = S_{ij}$. A project graph is acyclic if and only if it has no cycles.

Definition: $S_{ij} < S_{mn}$ implies S_{ij} precedes S_{mn} (or alternatively S_{mn} succeeds S_{ij}) if and only if there is a set of jobs $a_1, a_2 \ldots a_n$ $n > 2$ such that

$$S_{ij} = a_1 \ll a_2 \ll a_3 \ldots \ll a_n = S_{mn}$$

In other words, S_{ij} precedes S_{mn} if and only if there is a path from

S_{ij} to S_{mn} in the decision project graph G.

Assumption 1:* The precedes relation is asymmetric, that is if $S_{ij} < S_{mn}$ then it is false that $S_{mn} < S_{ij}$ for all S_{ij} and S_{mn} in J. Definition: A relation that is transitive and asymmetric is said to be a preference relation.

Theorem** III-1: If assumption 1 holds, then the predecessor relation is a preference relation, and the graph G is acyclic.

Definition: A technologically ordered job list $\hat{J} = \{a_1, a_2 \ldots a_n\}$ is obtained from a set of jobs $J = \{a, b, c \ldots\}$ by listing them so that no job appears on the list until all of its predecessors have already appeared.

Theorem** III-2: Assumption 1 holds if and only if it is possible to list the jobs in J in a technologically ordered job list $\hat{J}$.

In addition to these definitions and theorems from reference [49], several additional conventions are necessary because of the fact that some jobs may be eliminated from the decision project graph as the result of decisions that are made. If we decide to do one of the jobs in a job set, then all immediate predecessor relations that the job satisfies must hold in the final graph. If we decide not to do that job,

---

*Note that this assumption differs from the corresponding assumption in [49] in that the requirement of K-intransitivity is omitted. For this reason theorem 1 of that reference does not hold in the present context.

**The proofs of these theorems are exactly as in reference [9].

then none of its immediate predecessor relations hold. In the decision project graph, if we decide not to do a given job, then we must remove that job together with all edges that impinge on it from the decision project graph to obtain the final project graph. It follows from this that if any job, S_{ij} has a sole immediate predecessor S_{mn}, and if that predecessor is a member of a job set, it will be necessary to create a dummy immediate predecessor relation between S_{ij} and a job which is a predecessor of S_{mn}. If this is not done, then it would be possible for the path containing S_{ij} to be broken and S_{ij} would lose its project time ordering. Similarily a dummy immediate successor relation must be established for jobs having only one immediate successor, if that successor is a member of a job set. In addition it may be necessary to create a dummy relation between two jobs even if both have several immediate predecessors and successors. If on any path, two jobs are separated by a job which could be eliminated, and if it is desired to maintain a technological ordering of the two jobs, a dummy immediate predecessor relation must be established between them.

For a given project, when the jobs are technologically ordered and all planning decisions are made designating the jobs to be performed in each set, the normal critical path analysis may then be carried out. The usual concepts of early start, late start, critical path, etc., will apply to this reduced graph. These terms will be used throughout the balance of the thesis without further definition.

3. Decision Project Graphs

A graphical representation of the combined planning and scheduling problem is shown in the decision project graph of Figure III-1. In this graph the circular nodes represent jobs and the triangular nodes introduce the mutually exclusive decision job nodes of a job set. In addition to the precedence constraints implied by the directed line segments, we impose several "alternative interdependencies". These are

(a) $d_{9,1} \leqslant d_{6,2}$

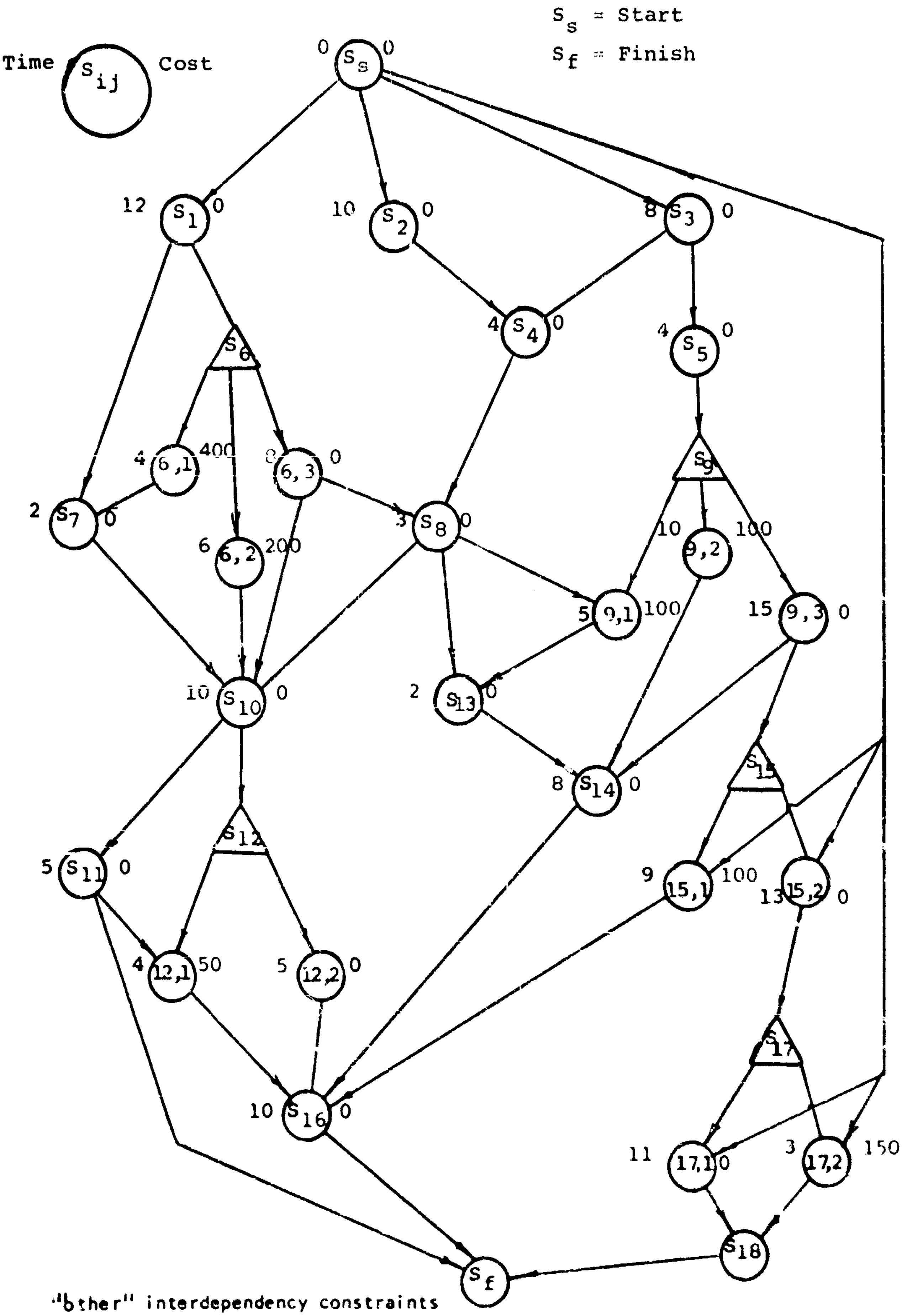
(b) $d_{6,2} = d_{12,1}$

where (a) says that $S_{9,1}$ may only be performed if $S_{6,2}$ is performed; and (b) says that $S_{6,2}$ and $S_{12,1}$ must both be performed or both not be performed. The problem, then, is to select the project graph which minimizes total project cost.

4. Decision Graph Solution by Integer Programming

Consider job set (1) and its associated decision variables with constraints given by (2) (3), and (4). Besides these, there may be any of the other constraints descussed in the second section of this chapter or other constraints showing various types of complicated interdependencies between jobs in the project.

With each job S_i, we associate a time, t_i, and a cost, c_{ij}. Also, we assume a reward of 'r' dollars per day for each day under D, the required due date of the project, and a penalty payment 'p' for each day beyond D. As defined earlier, W_f^+ will be the number of



"other" interdependency constraints

$d_{9,1} \leq d_{6,2}$

$d_{6,2} \leq d_{12,2}$

FIGURE III - 1

days after D that the project finishes and W_f^- will be the number of days before D that the project finishes. We can now formulate the integer programming problem of selecting that best project graph and finding its critical path

$$\text{Min} \sum_{i=1}^{\bar{n}} \sum_{j=1}^{k(i)} d_{ij}c_{ij} - r\,W_f^- + p\,W_f^+$$

The first term calculates the cost of all the decision jobs or job alternatives that are to be performed. It is governed by the constraints

$$0 \leq d_{ij} \leq 1$$

$$\sum_{j\ 1}^{k(i)} d_{ij} = 1 \qquad i = 1,2,\ldots,m$$

where d_{ij} is an integer. The second term is explained by the constraint

$$W_f - W_f^+ + W_f^- - D = 0$$

where W_f is the early start time of Finish, the last job in the project. If $W_f > D$, then the project is not completed until after the due date so that $W_f^+ = W_f - D$ and a penalty of pW_f^+ is incurred.

Other constraints must hold because of the precedence relations. For instance, if S_i and S_m are unit set jobs and $S_i \ll S_m$, we have

$$W_i + t_i \leq W_m$$

where W_i is the early start time of job S_i. If S_m is a unit job set j and S_{ij} is from a multi-job set and $S_{ij} \ll S_m$, then

$$-M(1 - d_{ij}) + W_{ij} + t_{ij} \leqslant W_m$$

where M is a large enough number so that the inequality does not constrain the variable d_{ij} unless $d_{ij} = 1$. Thus all paths through jobs that are not performed will be broken.

It is now possible to set up the problem of Figure III-1 as an integer programming problem. Note that it is not necessary to include the cost of unit set jobs in the functional or the finish node (S_f) in the precedence constraints.

$$\text{Min.} \quad 400\, d_{6,1} + 200\, d_{6,2} + 0d_{6,3} + 200\, d_{9,1} + 100\, d_{9,2}$$
$$+ 0d_{9,3} + 50d_{12,1} + 0d_{12,2} + 100d_{15,1} + 0d_{15,2} + 0d_{17,1}$$
$$+ 150d_{17,2} + pW_f^{+} - rW_f^{-}$$

St.

<u>Precedence Constraints</u> (link formulation)

$$t_1 \leqslant W_7$$
$$t_2 \leqslant W_4$$
$$t_3 \leqslant W_4$$
$$t_3 \leqslant W_5$$
$$t_1 \leqslant W_{6,1}$$
$$t_1 \leqslant W_{6,2}$$
$$t_1 \leqslant W_{6,3}$$
$$-M(1-d_{6,1}) + t_{6,1} + W_{6,1} \leqslant W_7$$
$$t_7 + W_7 \leqslant W_{10}$$
$$-M(1-d_{6,2}) + t_{6,2} + W_{6,2} \leqslant W_{10}$$
$$-M(1-d_{6,3}) + t_{6,3} + W_{6,3} \leqslant W_{10}$$
$$-M(1-d_{6,3}) + t_{6,3} + W_{6,3} \leqslant W_8$$

$$t_4+W_4 \leqslant W_8$$
$$t_8+W_8 \leqslant W_{10}$$
$$t_5+W_5 \leqslant W_{9,1}$$
$$t_5+W_5 \leqslant W_{9,2}$$
$$t_5+W_5 \leqslant W_{9,3}$$
$$t_8+W_8 \leqslant W_{9,1}$$
$$t_8+W_8 \leqslant W_{13}$$
$$-M(1-d_{9,1})+t_{9,1}+W_{2,1} \leqslant W_{13}$$
$$-M(1-d_{9,2})+t_{9,2}+W_{9,2} \leqslant W_{14}$$
$$-M(1-d_{9,3})+t_{9,3}+W_{9,3} \leqslant W_{14}$$
$$-M(1-d_{9,3})+t_{9,3}+W_{9,3} \leqslant W_{15,1}$$
$$-M(1-d_{9,3})+t_{9,3}+W_{9,3} \leqslant W_{15,2}$$
$$t_{10}+W_{10} \leqslant W_{11}$$
$$t_{11}+W_{11} \leqslant W_{12,1}$$
$$t_{10}+W_{10} \leqslant W_{12,1}$$
$$t_{10}+W_{10} \leqslant W_{12,2}$$
$$t_{13}+W_{13} \leqslant W_{14}$$
$$t_{14}+W_{14} \leqslant W_{16}$$
$$-M(1-d_{12,1})+t_{12,1}+W_{12,1} \leqslant W_{16}$$
$$-M(1-d_{12,2})+t_{12,2}+W_{12,2} \leqslant W_{16}$$
$$-M(1-d_{15,1})+t_{15,1}+W_{15,1} \leqslant W_{16}$$
$$-M(1-d_{15,2})+t_{15,2}+W_{15,2} \leqslant W_{17,1}$$
$$-M(1-d_{15,2})+t_{15,2}\ W_{15,2} \leqslant W_{17.2}$$
$$-M(1-d_{17,1})+t_{17,1}+W_{17,1} \leqslant W_{18}$$
$$-M(1-d_{17,2})+t_{17,2}+W_{17,2} \leqslant W_{18}$$

$$t_{11} + W_{11} \leqslant W_f$$

$$-M(1-d_{15,2})+t_{15,2} \leqslant W_f$$

$$t_{16} + W_{16} \leqslant W_f$$

$$t_{18} + W_{18} \leqslant W_f$$

Due Date $W_f - W_f^+ + W_f^- = D$

Interdependence

Job Alternatives

$$d_{6,1} + d_{6,2} + d_{6,3} = 1$$

$$d_{9,1} + d_{9,2} + d_{9,3} = 1$$

$$d_{12,1} + d_{12,2} = 1$$

$$d_{15,1} + d_{15,2} = 1$$

$$d_{17,2} + d_{17,2} = 1$$

Other Interdependency

$$d_{9,1} \leqslant d_{6,2}$$

$$d_{6,2} = d_{12,2}$$

$$0 \leqslant d_{ij} \leqslant 1$$

An alternate formulation suggested in [19] and the formulation to be extended in this chapter, constrains the length of the critical path by including one constraint for each possible path from S_S to S_F. A typical path might pass through S_S, S_1, ..., S_{ij}, ..., S_{mn} and S_F. The constraint for this path would be written

$$t_1 + \ldots -M(1-d_{ij}) + t_{ij} \ldots -M(1-d_{mn}) + t_{mn} \leqslant W_f$$

The precedence constraints on the length of the critical path for Figure III-1 will now be shown in "path" form.

Precedence (paths).

$$t_1+t_7+t_{10}+t_{11}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1+t_7+t_{10}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1+t_7+t_{10}-M(1-d_{12,2})+t_{12,2}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,1})+t_{6,1}+t_7+t_{10}+t_{11}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,1})+t_{6,1}+t_7+t_{10}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,1})+t_{6,1}+t_7+t_{10}-M(1-d_{12,2})+t_{12,2}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,2})+t_{6,2}+t_{10}+t_{11}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,2})+t_{6,2}+t_{10}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,2})+t_{6,2}+t_{10}-M(1-d_{12,2})+t_{12,2}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,3})+t_{6,3}+t_{10}+t_{11}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,3})+t_{6,3}+t_{10}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,3})+t_{6,3}+t_{10}-M(1-d_{12,2})+t_{12,2}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,3})+t_{6,3}+t_8+t_{10}+t_{11}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,3})+t_{6,3}+t_8+t_{10}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,3})+t_{6,3}+t_8+t_{10}-M(1-d_{12,2})+t_{12,2}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,3})+t_{6,3}+t_8-M(1-d_{9,1})+t_{9,1}+t_{13}+t_{14}+t_{16} \leqslant W_f$$

$$t_1-M(1-d_{6,3})+t_{6,3}+t_8+t_{13}+t_{14}+t_{16} \leqslant W_f$$

$$t_2+t_4+t_8+t_{10}+t_{11}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_2+t_4+t_8+t_{10}-M(1-d_{12,1})+t_{12,1}+t_{16} \leqslant W_f$$

$$t_2+t_4+t_8+t_{10}-M(1-d_{12,2})+t_{12,2}+t_{16} \leqslant W_f$$

$$t_2+t_4+t_8-M(1-d_{9,1})+t_{9,1}+t_{13}+t_{14}+t_{16} \leqslant W_f$$

$$t_2+t_4+t_8+t_{13}+t_{14} \leqslant W_f$$

$$-M(1-d_{15,2})+t_{15,2}-M(1-d_{17,1})+t_{17,1}+t_{18} \leqslant W_f$$

$$
\begin{aligned}
-M(1-d_{15,2})+t_{15,2}-M(1-d_{17,2})+t_{17,2}+t_{18} &\leq W_f \\
-M(1-d_{17,1})+t_{17,1}+t_{18} &\leq W_f \\
-M(1-d_{17,2})+t_{17,2}+t_{18} &\leq W_f \\
t_3+t_4+t_8+t_{10}+t_{11}-M(1-d_{12,1})+t_{12,1}+t_{16} &< W_f \\
t_3+t_4+t_8+t_{10}-M(1-d_{12,1})+t_{12,1}+t_{16} &\leq W_f \\
t_3+t_4+t_8+t_{10}-M(1-d_{12,2})+t_{12,2}+t_{16} &\leq W_f \\
t_3+t_4+t_8-M(1-d_{9,1})+t_{9,1}+t_{13}+t_{14}+t_{16} &< W_f \\
t_3+t_4+t_8+t_{13}+t_{14} &< W_f \\
t_3+t_5-M(1-d_{9,1})+t_{9,1}+t_{13}+t_{14}+t_{16} &< W_f \\
t_3+t_5-M(1-d_{9,2})+t_{9,2}+t_{14}+t_{16} &\leq W_f \\
t_3+t_5-M(1-d_{9,3})+t_{9,3}+t_{14}+t_{16} &< W_f \\
t_3+t_5-M(1-d_{9,3})+t_{9,3}-M(1-d_{15,2})+t_{15,2} &\leq W_f \\
t_3+t_5-M(1-d_{9,3})+t_{9,3}-M(1-d_{15,2})+t_{15,2} &\leq W_f \\
-M(1-d_{15,2})+t_{15,2} &< W_f \\
t_1+t_7+t_{10}+t_{11} &\leq W_f \\
t_1-M(1-d_{6,1})+t_{6,1}+t_7+t_{10}+t_{11} &\leq W_f \\
t_1-M(1-d_{6,2})+t_{6,2}+t_{10}+t_{11} &\leq W_f \\
t_1-M(1-d_{6,3})+t_{6,3}+t_{10}+t_{11} &< W_f \\
t_1-M(1-d_{6,3})+t_{6,3}+t_8+t_{10}+t_{11} &\leq W_f \\
t_2+t_4+t_8+t_{10}+t_{11} &\leq W_f \\
t_3+t_4+t_8+t_{10}+t_{11} &< W_f
\end{aligned}
$$

We suggest that typically in the second formulation of the precedence constraints (and it is also true of the first) many of the constraints will be redundant. This may be observed in the constraints representing the problem of this chapter. The next chapter will develop an algorithm which will select and eliminate the redundant constraints.

Chapter IV

Decision Network Reduction

The previous chapter formulated the DCPM problem as an integer programming problem that included either a constraint for every link in a precedence graph or a constraint for every path in the network. For the relatively small network of Figure III-1, either approach gives approximately forty precedence constraints. In this chapter we will show that in the "path" formulation of precedence constraints, many of the constraints are dominated and, therefore, that many of them may be eliminated. It is also shown that the "other" interdependency relations may make some precedence constraints infeasible and these, too, may be eliminated. Finally, an algorithm, is developed to reduce decision networks to equivalent networks containing only decision jobs and maximal distance between them. This reduced network could be used to generate a set of undominated paths to be included in the integer programming formulation as precedence constraints.

2. Dominance Tests for Constraint Elimination

Given any decision network, it is possible to order the jobs, including the decision jobs, from S_s to S_f so that no job appears on the list until all of its predecessors have appeared. We may now proceed down this technologically ordered list labelling the decision jobs $u(1)$, $u(2)$, ..., $u(k)$, ..., $u(h)$ where h is the total number of decision jobs in the network. Thus we may uniquely specify any decision job S_{mn} as $S_{u(i)}$ $i = 1, \ldots, h$

If we now define P as the set of all paths from S_s to S_f in the network, P may be partitioned into the following subsets,

$$P \quad \left\{ P^o, P^1, P^2, \ldots, P^j, \ldots, P^b \right\}$$

such that

$P^o = \{$0 order paths, those containing none of the decision jobs$\}$

$P^1 = \{$1st order paths, those containing one of the decision jobs$\}$

$P^j = \{$jth order paths, those containing exactly j of the decision jobs$\}$

$P^b = \{$bth order paths, those containing exactly b of the decision jobs$\}$

where b is the maximum number of decision jobs on any path from S_s to S_f

It is now possible to sub-partition P^1 into h subsets in the following way.

$$P^1 = \left\{ P^1_{u(1)}, P^1_{u(2)}, \ldots, P^1_{u(k)}, \ldots, P^1_{u(h)} \right\}$$

where $P^1_{u(k)}$ is that subset of paths containing only one decision job which is $S_{u(k)}$

Similarly, the 2nd order paths may be partitioned into

$$\frac{h!}{(h-2)!2!} \quad \text{subsets as follows.}$$

$$P^2 = \left\{ P^2_{u(i),u(j)} \right\} \qquad \begin{array}{l} i = 1, 2, \ldots, h-1 \\ j = i+1, \ldots, h \end{array}$$

and P^j may be partitioned into

$$\frac{h!}{(h-j)!j!} \qquad \text{subsets.}$$

In total, the maximum number of subsets, each containing a unique combination of decision jobs will be

$$1 + \frac{h!}{(h-2)!2!} \cdots + \frac{h!}{(h-j)!j!} \quad \cdots \quad + 1 = 2^h$$

The actual number of subsets required would be much less than this however. The set of interdependency relations from Chapter III

$$\sum_{j=i}^{k(i)} d_{ij} = 1$$

guarantees that many combinations of decision nodes are not feasible.

Now $\quad P^j_{u(m), \ldots, u(k), \ldots, u(n)}$

is a set of paths containing a particular combination of j of the h decision jobs. For any DCPM problem, any solution, that is any selection of decision jobs to perform that meets the interdependency constraints, will have one of the following mutually exclusive properties. Either all of the j decision jobs in $P^j_{u(m), \ldots, u(k), \ldots, u(n)}$ are performed or at least one of them is not performed.

If one of them is not performed, then the precedence constraints representing all of the paths in this set will not be binding on W_f, the early start of the Finish node.

That is if

$$d_{u(m)} + \ldots + d_{u(k)} + \ldots + d_{u(n)} < j$$

then the constraint representing any path in the subset will meet the condition

$$-M(1-d_{u(m)}) + t_{u(m)} - \dots -M(1-d_{u(k)}) + t_{u(k)} - \dots$$
$$-M(1-d_{u(n)}) + t_{u(n)} + T < 0 \leq W_f$$

where T represents the sum of the times of non-decision jobs on the path.

If all the jobs in the subset are performed, then the finish day of the project, W_f, will be constrained by equations representing all paths in the subset. Since the constraints are identical except that the value for T, the sum of the times of the non-decision jobs on a particular path, vary, it is only necessary to retain the constraint that has the maximum value for T. All constraints representing other paths in the subset will be dominated by the one chosen. Thus we require only one path for each possible combination of decision nodes. We will term this path an "undominated" path.

3. Implementation of Dominance Tests for Path Elimination

This section will develop an algorithm based on the repetitive application of the largest path calculation to generate an undominated set of paths in a decision network. As stated above, the undominated path will simply be the longest path in a subset of paths which all contain a particular combination of decision jobs. For example, in the subset $P^j_{u(i)}$ $i = 1, \dots, j$, if it is not empty, we must calculate the longest path through jobs

$$S_s,\ S_{u(1)},\ S_{u(2)},\ \dots,\ S_{u(k)},\ \dots,\ S_{u(j)},\ S_f$$

which goes through no other decision job. This is equivalent to finding the longest partial path connecting

S_s and $S_{u(1)}$, $S_{u(1)}$ and $S_{u(2)}$, etc.

and combining them to form the complete path. Again each partial path must not contain other decision jobs since this would result in the calculation of the longest path through a different combination of decision jobs.

A routine based on the usual longest path calculation (CPM) was written to generate the maximal distances between all decision nodes and S_s, S_f. The algorithm uses two time values for each job in the network. The "actual time" is the estimated completion time for the job which remains constant throughout while the "current time" is equal either to actual time or $-M$, a large negative number. As we shall see, if a job time is taken as $-M$, then no path through the job has a non-negative length and all such paths will be ignored. A flow chart for the algorithm will now be presented and reference will be made to Tables IV-1, 2 and 3 which show some steps of the algorithm for the decision network of Figure III-1.

1. List the jobs of the decision network in technological order
2. Form a matrix with a row for each job in the network. Column 1 will contain the job number; Column 2, the actual time; Column 3, the job's predecessors, Columns 4, 5 and 6, the current time, the early start time and the early finish time, respectively. (ie. Table IV-1)
3. Identify all decision jobs in Column 1. (In Tables IV-1 brackets are used)

4. Set the current time, Column 4, for all decision jobs at -M, for all other jobs set the current time equal to actual time. Set counter $II = S_s$. Go to Step 8.
5. Beginning at II, search down Column 1 for the first decision job below II, say this is $S_{u(n)}$. Set $II = S_{u(n)}$.
6. Set the early start, Column 5, for all jobs above II at -M.
7. Set the current time of II and the early start of II at 0.
8. Calculate the early start time of all jobs beneath II on the technologically ordered job list. If the early start value calculated is negative, enter -M in the early start column. To calculate the early start of the job directly beneath II, list all the predecessors of the job and for each predecessor add the predecessors early start time to its current time. The largest of these values, for all predecessors, is the early start time for the job being considered. Then proceed to the next job on the technologically ordered list for the next early start calculation.
9. For all decision jobs, add early start time to actual time to compute early finish time, Column 6.
10. For all decision jobs beneath II on the technological list and for S_f, record the early finish time if it is non-negative. These times will be the longest sub-path from the finish of decision job II to the finish of the decision job being examined (the longest path containing no other decision job).
11. If II is the last decision job in the technological list, go to Step 12. If not, go to Step 5.

12. HALT.

This algorithm has been applied to the problem of Figure III-1 with results shown in Table IV-3. Each non-blank entry in this matrix indicates the length of the maximal path between pairs of decision jobs or between the Start and Finish nodes. Note that the entries for line one, the maximal distances from S_s to all decision jobs, are taken directly from Column 6, Table IV-1. Similarily, the entries for row two, the distances from $S_{6,1}$ to all decision jobs, are taken from Column 6 of Table IV-2. Blank entries in the matrix indicate that no path not containing a decision job connects the to jobs. The output of a program which generates data for the matrix of IV-3, and estimates of computation times for various decision networks are given in Appendix B.

4. Feasibility Tests for Path Elimination

Consider a particular subset of paths

$$P^j_{u(i)} \qquad i = 1, 2, \ldots, j$$

For the constraints representing these paths to constrain w_f, the following condition must hold

$$\sum_{i=1}^{j} d_{u(i)} = j$$

This condition may be contradicted by the "other" interdependency constraints

$$\text{ie.} \quad d_{u(1)} + d_{u(2)} = 1$$

If the relation $\sum_{i=1}^{j} d_{u(i)} = j$ can be shown to be infeasible, then all paths in $P_{u(i)}$ $\quad i = 1, 2, \ldots, j$, that is all paths containing

1	2	3	4	5	6
Job	Actual Time	Predecessors	Current Time	Early Start	Early Finish
S_s	0	0	0	0	0
1	12	S_s	12	0	12
2	10	S_s	10	0	10
3	8	S_s	8	0	8
4	4	2,3	4	10	14
5	4	3	4	8	12
(6,1)	4	1	-M	12	16
(6,2)	6	1	-M	12	18
(6,3)	8	1	-M	12	20
7	2	1,(6,1)	2	12	14
8	3	4,(6,3)	3	14	12
(9,1)	5	5,8	-M	17	22
(9,2)	10	5	-M	12	22
(9,3)	15	5	-M	12	27
10	10	(6,3),(6,3),7,8	10	17	27
11	5	10	5	27	32
(12,1)	4	10,11	-M	32	36
(12,2)	5	10	-M	27	32
13	2	8,(9,1)	2	17	19
14	8	(9,2),(9,3),13	8	19	27
(15,1)	9	93,S_s	-M	0	9
(15,2)	13	93,S_s	-M	0	13
16	10	(12,1),(12,2),14,(15,1)	10	27	37
(17,1)	11	(15,2),S_s	-M	0	11
(17,2)	3	(15,2)S_s	-M	0	3
18	5	(17,1),(17,2),S_s	5	0	5
S_f	0	16,18,(15,2),11	0	37	37

Table IV-1

1	2	3	4	5	6
Job	Actual Time	Predecessors	Current Time	Early Start	Early Finish
S_s	0	0	0	-M	--
1	12	S_s	12	-M	--
2	10	S_s	10	-M	--
3	8	S_s	8	-M	--
4	4	2,3	4	-M	--
5	4	3	4	-M	--
(6,1)	4	1	0	0	--
(6,2)	6	1	-M	-M	-M
(6,3)	8	1	-M	-M	-M
7	2	1,(6,1)	2	0	
8	3	4,63	3	-M	-M
(9,1)	15	5,8	-M	-M	-M
(9,2)	10	5	-M	-M	-M
(9,3)	15	5	-M	-M	-M
10	10	(6,2),(6,3),7,8	10	2	12
11	5	10	5	12	17
(12,1)	4	10,11	-M	17	21
(12,2)	5	10	-M	12	17
13	2	8,(9,1)	2	-M	-M
14	8	(9,2),(9,3),13	8	-M	-M
(15,1)	9	(9,3),S_s	-M	-M	-M
(15,2)	13	(9,3),S_s	-M	-M	-M
16	10	(12,1),(12,2),14,(15,1)	10	-M	-M
(17,1)	11	(15,2),S_s	-M	-M	-M
(17,2)	3	(15,2),S_s	-M	-M	-M
18	5	(17,1),(17,2),S_s	5	-M	-M
S_f	0	16,13,(15,2),11	0	17	17

Table IV-2

Reduced Network Matrix

T O

FROM

	6,1	6,2	6,3	9,1	9,2	9,3	12,1	12,2	15,1	15,2	17,1	17,2	S_f
S_s	16	18	20	22	22	27	36	32	9	13	11	3	37
6,1							21	17					17
6,2							19	15					15
6,3				8			22	18					23
9,1													20
9,2													18
9,3									9	13			18
12,1													10
12,2													10
15,1													10
15,2											11	3	0
17,1													5
17,2													5

Table IV-3

the combination of decision jobs.

$$S_{u(i)} \qquad i = 1, 2, \ldots, j$$

may be eliminated from the integer programming formulation of the DCPM problem. Paths not eliminated by the interdependency constraints will be termed "feasible" paths.

5. Application of Dominance and Feasibility Tests to a DCPM Network

When the tests of the previous three sections are applied to precedence equations of the example Figure III-1, the reduced set of equations of Table IV-4 result. For example, the set $P^1_{(12,1)}$ contains the 6 paths

$$t_1 + t_7 + t_{10} - M(1-d_{12,1}) + t_{12,1} + t_{16} \leqslant W_f$$

$$t_1 + t_7 + t_{10} + t_{11} - M(1-d_{12,1}) + t_{12,1} + t_{16} \leqslant W_f$$

$$t_2 + t_4 + t_8 + t_{10} + t_{11} - M(1-d_{12,1}) + t_{12,1} + t_{16} \leqslant W_f$$

$$t_2 + t_4 + t_8 + t_{10} - M(1-d_{12,1}) + t_{12,1} + t_{16} \leqslant W_f$$

$$t_3 + t_4 + t_8 + t_{10} + t_{11} - M(1-d_{12,1}) + t_{12,1} + t_{16} \leqslant W_f$$

$$t_3 + t_4 + t_8 + t_{10} - M(1-d_{12,1}) + t_{12,1} + t_{16} \leqslant W_f$$

These may be rewritten

$$-M(1-d_{12,1}) + 38 \leqslant W_f$$

$$-M(1-d_{12,1}) + 43 \leqslant W_f$$

$$-M(1-d_{12,1}) + 46 \leqslant W_f$$

$$-M(1-d_{12,1}) + 41 \leqslant W_f$$

$$-M(1-d_{12,1}) + 44 \leqslant W_f$$

$$37 \leqslant W_f$$

$$-M(1-d_{6,3}) + 43 \leqslant W_f$$

$$-M(1-d_{9,1}) + 42 \leqslant W_f$$

$$-M(1-d_{9,2}) + 40 \leqslant W_f$$

$$-M(1-d_{9,3}) + 45 \leqslant W_f$$

$$-M(1-d_{12,1}) + 46 \leqslant W_f$$

$$-M(1-d_{12,2}) + 42 \leqslant W_f$$

$$-M(1-d_{6,1}) - M(1-d_{12,2}) + 43 \leqslant W_f$$

$$-M(1-d_{6,2}) - M(1-d_{12,1}) + 37 \leqslant W_f$$

$$-M(1-d_{6,3}) - M(1-d_{12,2}) + 48 \leqslant W_f$$

$$-M(1-d_{9,3}) - M(1-d_{15,1}) + 46 \leqslant W_f$$

$$-M(1-d_{9,3}) - M(1-d_{15,3}) - M(1-d_{17,2}) + 56 \leqslant W_f$$

$$-M(1-d_{15,2}) - M(1-d_{17,1}) + 29 \leqslant W_f$$

$$-M(1-d_{15,2}) - M(1-d_{17,2}) + 21 \leqslant W_f$$

$$-M(1-d_{17,1}) + 16 \leqslant W_f$$

$$-M(1-d_{17,2}) + 8 \leqslant W_f$$

$$-M(1-d_{9,3}) - M(1-d_{15,2}) - M(1-d_{17,1}) + 52 \leqslant W_f$$

Table IV-4

$$-M(1-d_{12,1}) + 39 \leqslant W_f$$

With $-M(1-d_{12,1}) + 46 \leqslant W_f$ in the problem, the other 5 constraints are redundant.

We will now give an example of constraint elimination by feasibility tests.

The equation

$-M(1-d_{6,3}) - M(1-d_{9,1}) + 48 \leqslant W_f$ will hold only if

$$d_{6,3} + d_{9,1} = 2$$

However, the interdependency constraints

$$d_{9,1} \leqslant d_{6,2}$$

and

$$d_{6,1} + d_{6,2} + d_{6,3} = 1$$

violate this condition. Therefore, the equation may be dropped.

6. Lower Bound Calculation

The set of path constraints has now been reduced from one per path in the original decision graph to one per combination of decision jobs lying in a path from S_s to S_f. The number may be even further reduced because of feasibility tests on the interdependency relations.

We will now define set

$M = \{$all feasible, undominated paths$\}$

and set

$M_i = \{$ the subset of paths in M that contain any of the decision jobs in decision set S_i, a multi-job set.$\}$ Each path in M_i contains exactly one of S_{ij}, $j = 1, 2, \ldots k(i)$ since we are assuming that only one job from each multi-job set may be selected, but they may also contain other decision jobs.

We will now define a "feasible solution" to a DCPM problem as a selection of a set of decision jobs that will satisfy all "alternative" and "other" interdependency constraints on the problem. This set of decision jobs will, with the non-decision jobs, form a usual project graph of the critical path method. Given the project graph resulting from a "feasible solution", we state that one of the paths from M_i will be included in that graph. If no such path exists, then no path through S_i exists in the network. This would violate our assumption that S_s is a predecessor and S_f is a successor to all jobs in the project graph.

Each constraint representing a path from M_i will be of the general form

$$-M(1-d_{ij}) \ldots\ldots\ldots\ldots\ldots -M(1-d_{mn}) + T_{i,e} \leq W_f$$

where $T_{i,e}$ is the length of path e in M_i.

If one of the paths from M_i must be performed, then a lower bound on W_f is the lowest value of $T_{i,e}$ for all paths in M_i. Furthermore, if we calculate a lower bound for every decision node, we may select the highest of the lower bounds as a lower bound on W_f. Any undominated, feasible path which is shorter than this bound may be eliminated since it is redundant, given the bound on W_f.

An Application of Lower Bound Calculation

The reduced set of equations from Table IV-4 will be used to illustrate the dominance tests described above. The only feasible paths through S_6, that is the set M_6, are the following paths,

$$-M(1-d_{6,3}) + 43 \leqslant W_f$$

$$-M(1-d_{6,1}) - M(1-d_{12,1}) + 47 \leqslant W_f$$

$$-M(1-d_{6,2}) - M(1-d_{12,2}) + 43 \leqslant W_f$$

$$-M(1-d_{6,3}) - M(1-d_{12,1}) + 52 \leqslant W_f$$

The minimum of the $T_{i,e}$ $e=1,\ldots,4$, is 43 and therefore 43 is a lower bound on the length of the critical path. Similarly, the lower bound provided by M_9 is 40, M_{12} is 42, M_{15} is 21 and M_{17} is 8. These calculations are shown in Table IV-5. The highest of the bounds is then 43 and all paths shorter than or equal to this may be removed. For this particular problem this reduction technique does not give any further improvement. The final set of predecessor constraints may now be written

(1)			$+\ 43$	$\leqslant$	W_f
(2)	$-M(1-d_{9,3})$		$+\ 45$	$\leqslant$	W_f
(3)	$-M(1-d_{12,1})$		$+\ 46$	$\leqslant$	W_f
(4)	$-M(1-d_{9,3})$	$-M(1-d_{15,1})$	$+\ 46$	$\leqslant$	W_f
(5)	$-M(1-d_{6,1})$	$-M(1-d_{12,1})$	$+\ 47$	$\leqslant$	W_f

(6) $-M(1-d_{9,3})-M(1-d_{15,2})\ -M(1-d_{17,2})\ +\ 48\ \leqslant\ W_f$

(7) $-M(1-d_{6,3})\ -M(1-d_{12,1})\ +\ 52\ \leqslant\ W_f$

(8) $-M(1-d_{9,3})-M(1-d_{15,2})\ -M(1-d_{17,1})\ +\ 56\ \leqslant\ W_f$

By selecting appropriate values for M and combining, these may be rewritten

(1) (3) $3d_{12,1} + 43 \leqslant W_f$

(2) (4) $15d_{9,3} + d_{15,1} + 39 \leqslant W_f$

(5) (7) $4d_{6,1} + 9d_{6,3} + 4d_{12,1} + 39 \leqslant W_f$

(6) (8) $15d_{9,3} + 13d_{15,2} + 11d_{17,1} + 3d_{17,2} + 17 \leqslant W_f$

The final problem, now in reduced path form, may be solved by any standard integer programming technique.

Computation of Lower Bound

	6,1	6,2	6,3	9,1	9,2	9,3	12,1	12,2	15,1	15,2	17,1	17,2	
P^0													37
P^1											1		16
												1	8
			1										43
				1									42
					1								40
						1							45
							1						46
								1					42
P^2	1						1						47
		1						1					43
			1				1						52
						1			1				46
										1	1		29
										1		1	21
P^3						1				1	1		56
						1				1		1	48
Lower Bound		43			40		42		21			8	

Table IV-5

Chapter V

A Network Algorithm for Restricted Types of Integer Programs

In the literature review of Chapter II, we referred briefly to several integer programming routines for problems containing 0,1 variables. In our terms, these techniques could optimize planning problems with "job alternative" and "other" types of interdependency. We will now develop an algorithm based on the calculations of the critical path method which will solve certain restricted types of integer programs. A numerical example of the algorithm will be given and then the algorithm will be applied to a problem taken from the decision jobs and interdependency constraints of a decision network.

2. The Network Algorithm

Consider the integer programming problem

(1) Maximize $wb + b_o$

Subject to $wA^o \leq c$

$w \geq o$, w integer

where b is an $m \times 1$ column vector, b_o is a scalar, c is a $1 \times n$ row vector and o and w are $1 \times m$ row vectors with values assigned to w so as to maximize the objective function subject to the constraints of (1). A^o is an $m \times n$ matrix which meets the following assumption.

Assumption 1: All rows of A^o contain a maximum of two non-zero entries and these are found in adjacent columns. The programming problem may now be interpreted as a network problem in the following way. For each variable w_i $i = 1,2, \ldots, m$ define a set of 0-1 variables, one for each possible integer level of w_i. These would be

$$w_i = \left\{ w_{i,o}, \; w_{i,1}, \ldots, \; w_{i,j}, \ldots \; w_{i,k(i)} \right\}$$

where $w_{i,j} = \begin{cases} 1 & \text{if variable } w_i = j \\ 0 & \text{otherwise} \end{cases}$

and $k(i)$ is the maximum possible value of w_i

Therefore $$\sum_{j\ o}^{k(i)} w_{ij} = 1$$

and $$w_i = \sum_{j=o}^{k(i)} j w_{ij}$$

A feasible solution to the programming problem will then consist of a choice of exactly one variable from each of the following sets such that the constraints of (1) are met.

$$w_1 = \left\{ w_{1,o}, \; w_{1,1}, \; \ldots\ldots\ldots\ldots\ldots, \; w_{1,k(1)} \right\}$$

$$w_2 = \left\{ w_{2,o}, \; w_{2,1}, \; \ldots\ldots\ldots\ldots\ldots, \; w_{2,k(2)} \right\}$$

$$w_m = \left\{ w_{m,o}, \; w_{m,1}, \; \ldots\ldots\ldots\ldots\ldots, \; w_{m,k(m)} \right\}$$

Now let the w_{ij} be nodes in a network formed as follows. Link each $w_{1,j}$, $j = 1,2, \ldots .k(1)$ to each $w_{2,j}$, $j = 1,2, \ldots .k(2)$ by a directed line segment. Then link each

$d_{t,j}$, $j = 1,2, \ldots ., k(t)$ to each

$d_{t+1,j}$, $j = 1, 2, \ldots, k(t+1)$

where $t = 2, 3, \ldots, m-1$

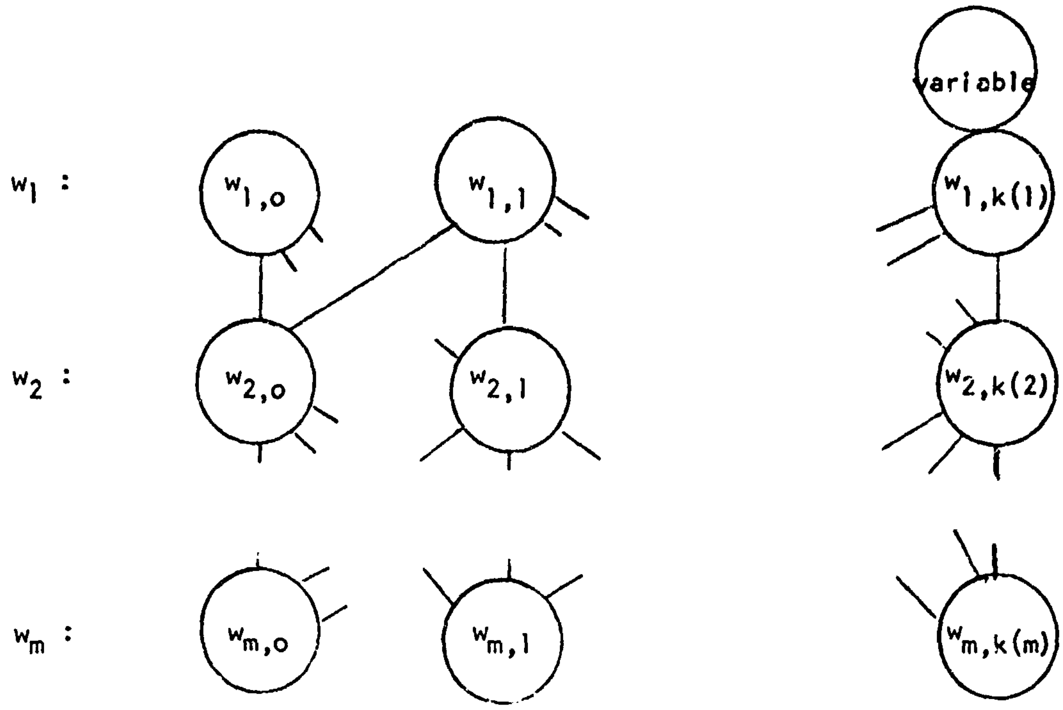
Since every variable in each set is linked to all variables in adjacent sets, the resulting network contains a path for all combinations of integer values of $w_1, w_2, \ldots, w_m$. Since there are no links within a variable set, the restriction $\sum_{j=o}^{k(i)} w_{ij} = 1$

is maintained. Thus every feasible and non-feasible solution is represented by a path in the network and every path is a feasible or non-feasible solution. Here a feasible solution is defined as the selection of a level for each of the w_i variables that meets the constraints of (1). The network may now be modified to eliminate paths representing non-feasible solutions.

First we will show how to calculate $k(1)$, the maximum integer value of variable w_1. Assume all variables w_i, $i = 2, 3, \ldots, m$ have the value o, that is $w_{i,o} = 1$.

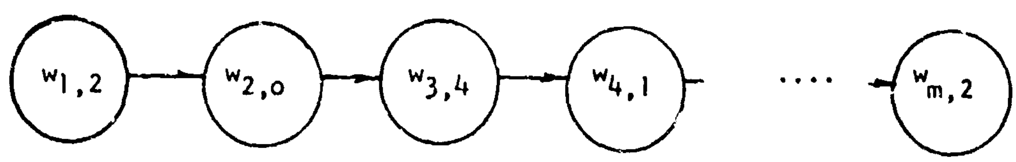
Then from the constraints in $wA^o \leqslant c$ of (1) that have only positive entries, it is possible to calculate a maximum value for w_1, that is $k(1)$. Similarly, we may calculate maximum values for all other variables.

We now have the graph



where at each level we have nodes representing all the feasible integer values of one variable, and each of these nodes is connected to all nodes of the suceeding level (or variable).

The constraints of (1), $wA^o \leqslant c$ may now be introduced exactly into the network. From our earlier discussion, it is known that each path in the network represents a particular combination of integer values of the variables, w_{ij} and each path is therefore a potential solution to the problem. For example, one path might be

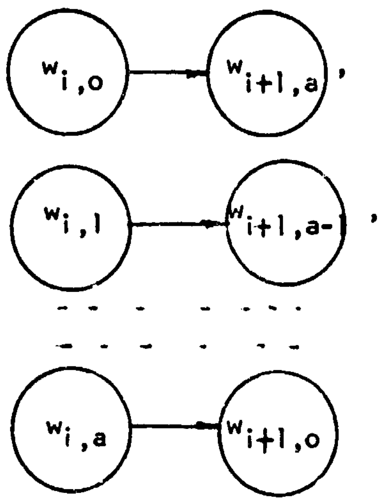


If such a solution (path) violates one of the constraints of (1), we remove the path by eliminating particular directed line segments that connect the integer levels of the two variables contained in the constraint. Assumption i guarantees that only two variables will be involved in the constraint and that they will be adjacent in the graph.

Graphical interpretations of certain constraints follow.

(i) $w_i + w_{i+1} = a$

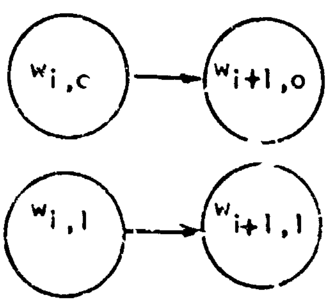
All links between variables w_i and w_{i+1} will be removed with the exception of those now listed.

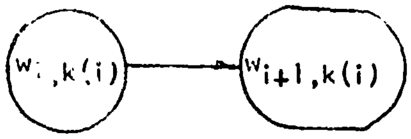


Where the sum of the values represented by the connected variables equals "a"

(ii) $w_i = w_{i+1}$

all links between variables w_i and w_{i+1} will be removed with the exception of those now listed.





where $k(i) \leq k(i+1)$

When all the constraints of (1) have been introduced into the network (by link elimination), we will term the resulting graph a "feasible" network.

Theorem 1: Every path in the feasible network is a feasible solution to the problem and every feasible solution may be represented by a path in that network.

PROOF: Between each level of the graph, non-feasible links are removed. If the constraining equations have been interpreted correctly, remaining links connect feasible pairs of variables.

Now assume a path exists in the reduced network from the set of variables w_1 to the set w_m and variables $w_{1,a}$, $w_{2,b}$, ..., $w_{m,c}$ are on that path. Variable set w_1 may only be combined in constraint equations with variable set w_2 if considering <u>all</u> the constraints on w_1, w_2 the combination of $w_{1,a}$, $w_{2,b}$ is feasible then $w_{1,a}$ may appear in any solution that contains $w_{2,b}$. The argument may be extended to w_3, w_4 and finally to w_m.

Therefore any path in the reduced network is a feasible solution.

Now assume a feasible solution exists for which a path does not exist. This implies that between two adjacent variable sets w_i, w_{i+1} there is no connecting link in the network between the variables from these sets which are in the feasible solution, that is between

$w_{i,a}$, $w_{i+1,b}$. Since links are only removed if in combination they are infeasible under the constraints, then a solution containing $w_{i,a}$ and $w_{i+1,b}$ is not feasible. Therefore every feasible solution must be represented by a path.

Theorem 2: If each node, w_{ij}, of the network graph is evaluated at jb_i, the solution to maximization problem is the longest path in the network.

PROOF: By Theorem 1 every path is a feasible solution to the problem. Since w_{ij} represent the original variable w_i at integer level j, the value of w_{ij} in the functional will be jb_i where b_i is the contribution of a unit of w_i to the functional. The path length for any solution is simply the value of the functional. The variables on the longest path are those that give an optimum solution to the maximization problem. Similarly the solution to a minimization problem may be found by determining the shortest path.

The familiar rules of Critical Path Scheduling will calculate the longest path in network. In this context the early start (E.S.) for a variable w_{ij} is interpreted to be the maximum value of the criterion function for variables w_1 to w_{i-1} that may be feasibly combined with w_{ij} (maximization). Late start (L.S.) is the maximum feasible value of the criterion function less the maximum value of the criterion function for variables w_{ij} to w_m that may be feasibly combined with w_{ij}.

In addition the "slack" values calculated by that technique have meaning in this context. The slack values of a job indicates the difference in length between the longest path on which the job is found and the longest path of the network. Since in the integer programming problem, the path length is equivalent to the value of the functional, and a solution is any path in the reduced network, the slack value indicates the minimum change in the functional which will be realized if w_{ij} is brought into the solution. Thus the "slack" values are exact evaluations for every w_{ij}, that is, for every variable w_i at every feasible level. These evaluations may be calculated for either maximization or minimization problems. As we would expect, evaluators for variables in the solution will be 0.

Theorem 3: If the variables in a problem which meets Assumption 1 may be listed so that no constraint exists between w_i, w_{i+1} $i = 1, \ldots,$ m-1 then the optimum solution to a problem made up of the variables $w_1 \ldots w_i$ and the constraints relating to them and the optimum of problem containing the variables $w_{i+1} \ldots w_m$ will together give an optimum solution to the combined problem and the problems are independent.

PROOF: Let the optimum solution to the first problem be a path $w_{1,a}$, $w_{2,b} \ldots w_{i,c}$ and the solution to the second be $w_{i+1,a} \ldots w_{m,b}$. Since no constraint exists between w_i, w_{i+1}, all nodes are connected. Therefore the longest path in the complete network will consist of the longest path above the level w_{i+1} plus the longest path below w_i. This will be the sum of the individual solutions to the two problems.

For purposes of the algorithm it is possible to have any set of mutually exclusive variables, rather than a series of mutually exclusive levels of the same variable in each variable set of the graph. Assumption 1 must hold for the set of variables in their relation to other sets of variables however. We also note that the contribution of the variable to the functional need not be a linear function.

3. A Numerical Example

Max. $6x_1 + 7x_2 + 8x_3 + 2x_4$

St. $x_1 < 4$

$3x_1 + 4x_2 \leqslant 10$

$x_2 + x_4 \leq 1$

$x_1 + x_3 = 6$

$x_i \geqslant 0$, integer

The maximum values for the variables, determined from the constraints, are

$$x_1 = 3, \quad x_2 = 1, \quad x_3 = 6, \quad x_4 = 1$$

Figure V-1 illustrates the beginning graph and Figure V-2 the graph with links removed to satisfy the constraints. In these networks, the variables when listed in the order x_3, x_1, x_2, and x_4, satisfy Assumption1. The optimum solution consists of those variables for which E.S. = L.S. It is possible that several such solutions would exist. In Figure V-2 these variables are seen to be $w_{1,o}$, $w_{2,1}$, $w_{3,6}$ and $w_{4,o}$ that is $w_1 = o$, $w_2 = 1$, $w_3 = 6$ and $w_4 = o$. The value of the functional for this solution is $(6)(o) + (7)(1) + (8)(6)$

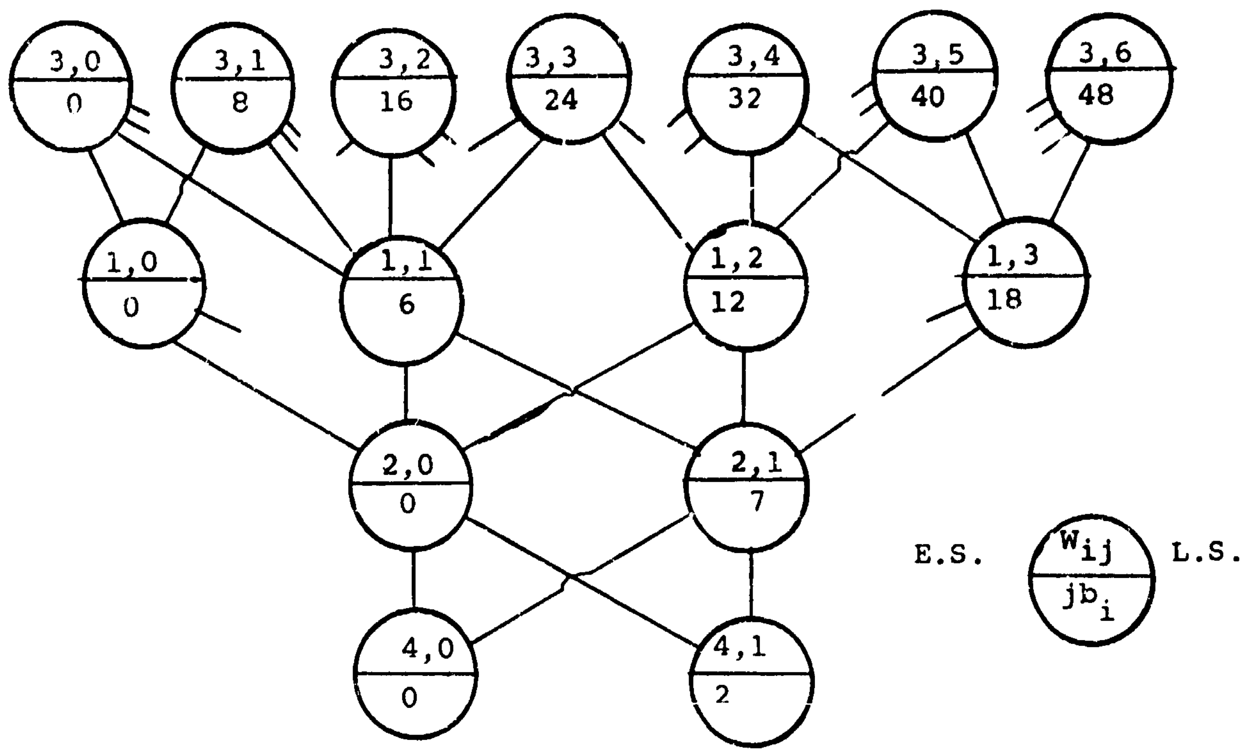


FIGURE V - 1

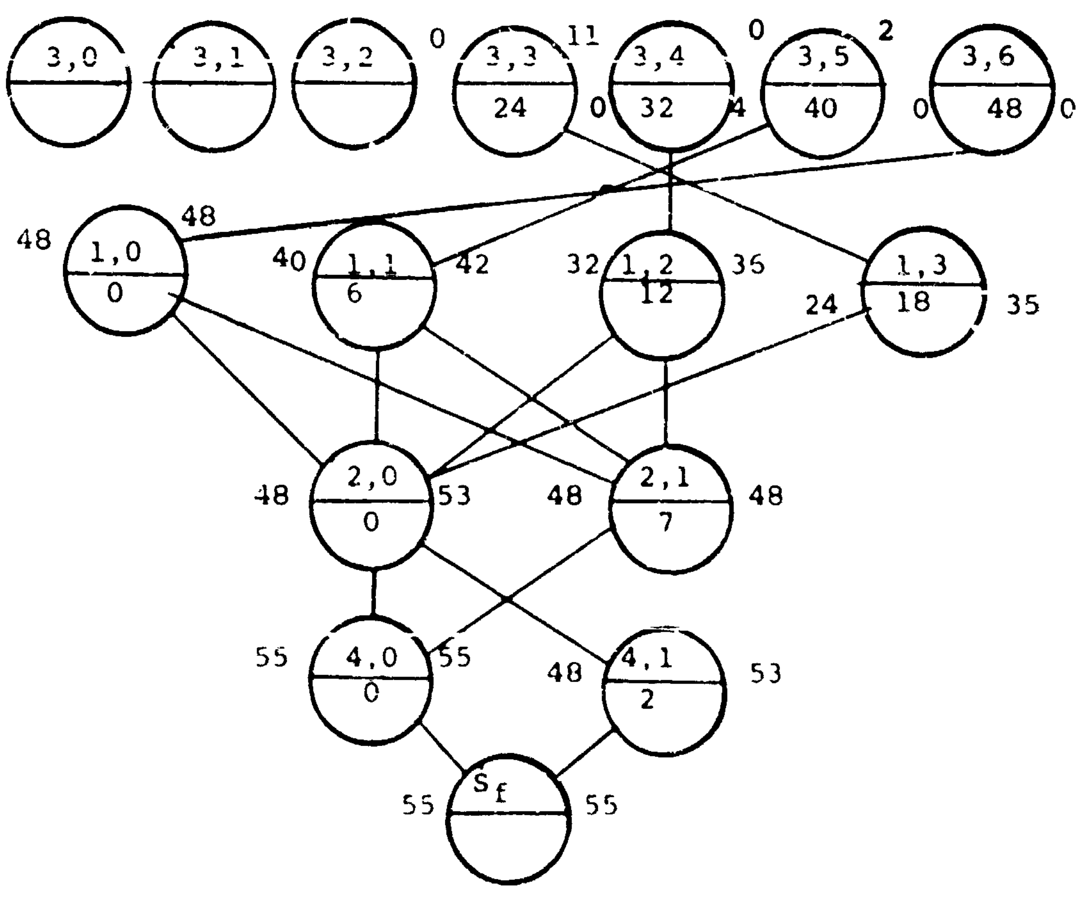


FIGURE V - 2

+ (2)(o)= 55. In addition, we see that variable x_3 at levels o, 1, and 2 is infeasible since variables $w_{3,o}$, $w_{3,1}$, $w_{3,2}$ lie on no complete path. Finally, we can calculate that the best solution containing variable x_2 at level o ($w_{2,o}$) would have a functional value, (48-53)= -5, five units lower than the optimal solution of 55.

4. Violations of Assumption 1

If Assumption 1 is violated, it may be possible to adapt the algorithm to the resulting problem. If to the problem of Section 3, we add the constraint $x_4 < x_1$, then Assumption 1 is violated. Since x_1 is involved in constraints with the three other variables, it is not possible to list the variables so that all constraints involve adjacent variables. The problem may still be solved in several ways.

(i) Define a new variable x_4^2 representing all possible combinations of variables x_2 and x_4. These are $x_{4,o}^{2,o}$, $x_{4,o}^{2,1}$, $x_{4,1}^{2,o}$. Constraint x_2 x_4 1 rules out $x_{4,1}^{2,1}$. The resulting network and its solution is shown in Figure V-3. The functional is now valued at 53 with

$$x_1 = 1, \quad x_2 = 1, \quad x_3 = 5, \quad x_4 = 0$$

(ii) The technique of truncated enumeration may be used on that part of the problem which cannot be introduced into the graphical network. A straightforward branching strategy would be to select in turn the variables that (a) appear in constraints and (b)

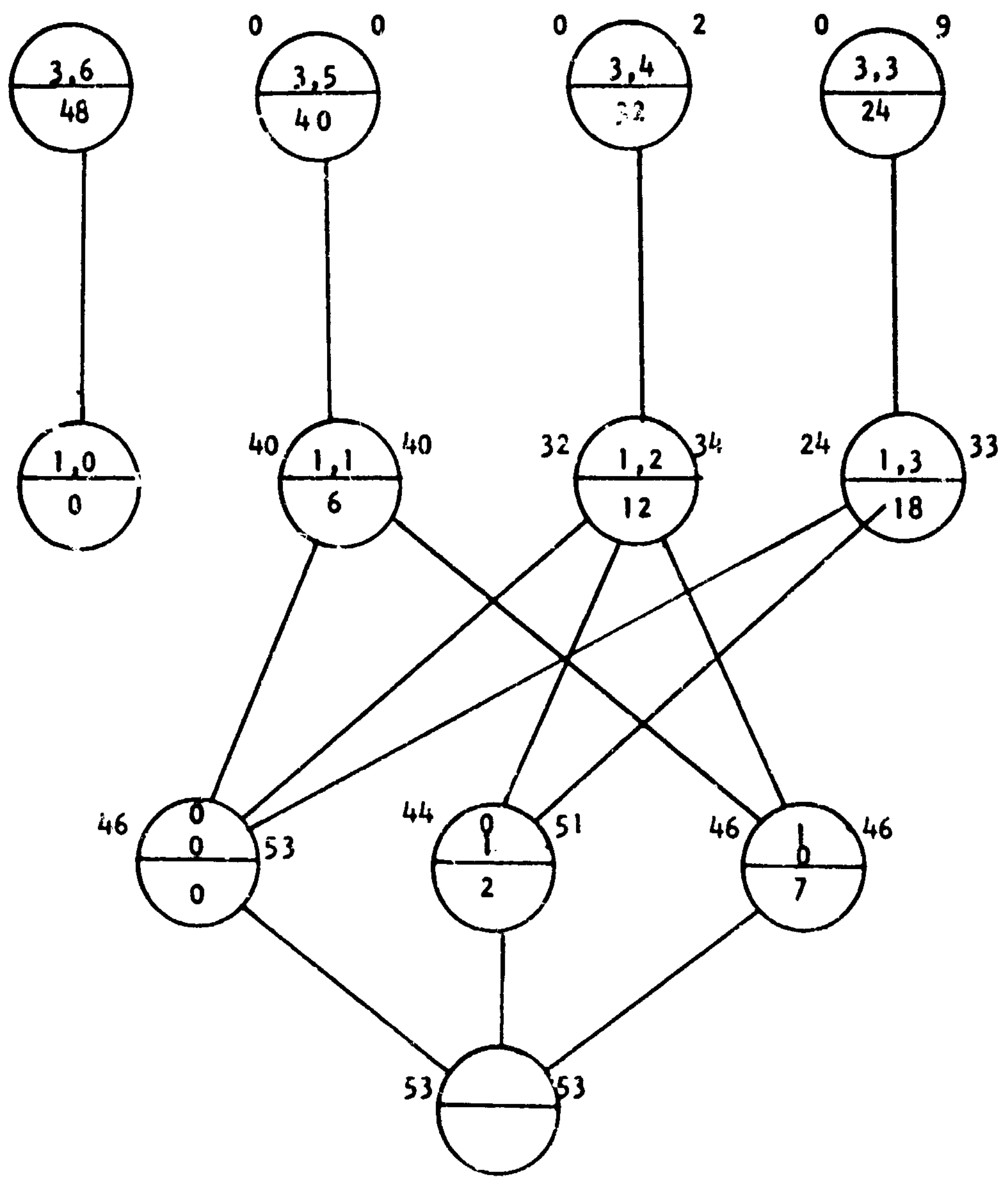
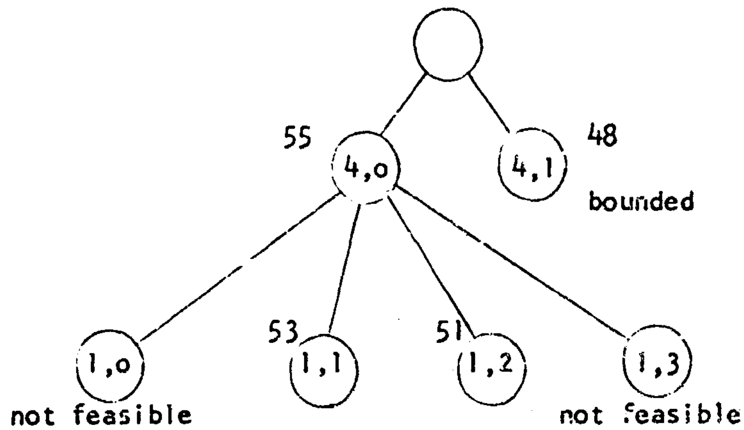

3,6
48
0
3,5
0
40
0
3,4
2
32
0
3,3
9
24
1,0
0
40
1,1
40
6
32
1,2
34
12
24
1,3
33
18
46
0
0
53
0
44
0
1
51
2
46
1
0
46
7
53
53

Figure V-3

cannot be listed in adjacent levels in our graph. At each level we would develop the node with the maximum evaluation (Maximization). At each node the bounds would be calculated using our problem network with the network modified to represent conditions on the path of the enumeration tree, above the node in question. The problem of this section will now be solved with this procedure (that is x_4 x_1 is added to the problem of Figure IV-2).



Therefore the selection of $x_4 = o$, $x_1 = 1$ gives an optimal solution of 53.

5. Application to a DCPM Problem

The method of this chapter has been developed to handle problems arising in Decision CPM networks. A project having six decision sets might give the following problem.

Minimize $\quad ox_1 + 10x_2 + 20x_3 + ox_4 + 7x_5 + 9x_6$

$$ox_7 + x_8 + 12x_{x9} + ox_{10} + 14x_{11} + 15x_{12}$$

$$ox_{13} + 6x_{14} + ox_{15} + 13x_{16} + k$$

Subject to "job alternative" interdependency

$$\text{Job 1} \quad x_1 + x_2 + x_3 = 1$$

$$\text{Job 2} \quad x_4 + x_5 + x_6 = 1$$

$$\text{Job 3} \quad x_7 + x_8 + x_9 = 1$$

$$\text{Job 4} \quad x_{10} + x_{11} + x_{12} = 1$$

$$\text{Job 5} \quad x_{13} + x_{14} = 1$$

$$\text{Job 6} \quad x_{15} + x_{16} = 1$$

and "other" interdependency.

$$x_3 = x_6$$

$$x_7 = x_{10}$$

$$x_5 = x_8$$

$$x_2 + x_5 \leqslant 1$$

$$x_{12} + x_{13} \leqslant 1$$

$$x_8 \leq x_{11}$$

$$x_1 \leqslant x_{16}$$

If the job sets are ordered 6, 1, 2, 3, 4, 5, then all constraints may be included in the first network. Figure V-4 illustrates the reduced graph and the evaluations of the variables. The minimum cost solution to this problem is $x_{15} = 1$, $x_2 = 1$, $x_4 = 1$, $x_7 = 1$, $x_{10} = 1$ and $x_{13} = 1$ for a total cost of 10.

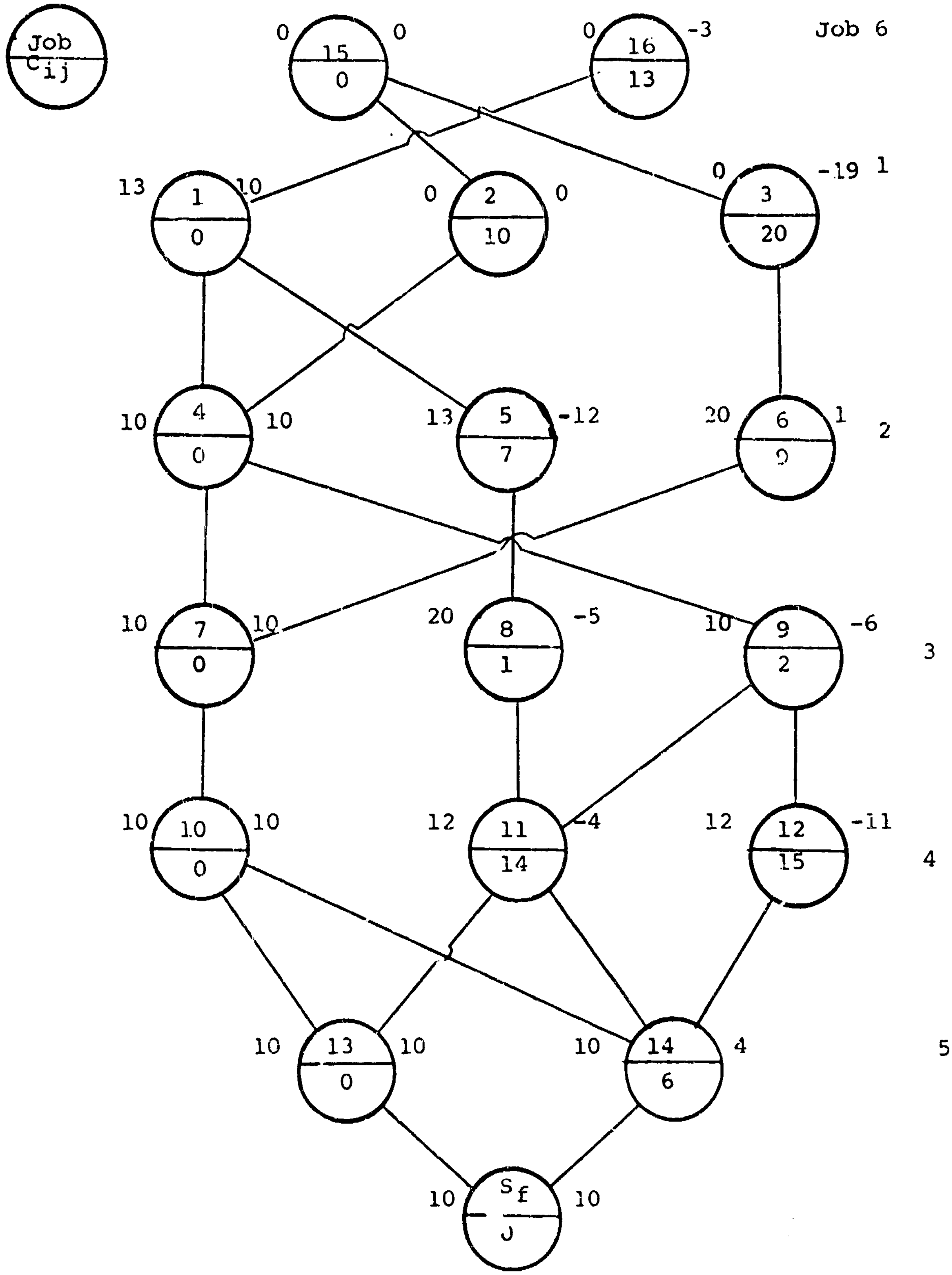

Job
C_{ij}
Job 6
15
0
16
13
1
0
2
10
3
20
4
0
5
7
6
9
7
0
8
1
9
2
10
0
11
14
12
15
13
0
14
6
S_f
0
1
2
3
4
5

FIGURE V-4

Chapter VI

Branch and Bound Algorithms for the DCPM Problem

In Chapter III the DCPM problem was formulated as an integer programming problem but for large problems this approach would require substantial computation time. For this reason, it was decided to examine the applicability of restricted enumeration solution techniques. These methods, under the general name "branch and bound" have been used for solving a series of rather difficult cominatorial problems. The applications include the travelling salesman problem [27, 51], a version of the plant location problem [78], truck routing [62] and production sequencing problems [42, 47, 61]. This work has demonstrated that for a wide range of problems these techniques are efficient.

The work "branch" in this context refers to a specific decision rule for enumerating a tree of all possible solutions to the problem. The "bound" term implies that at each node of the tree a maximum value (for maximization problems) of the criterion function is calculated. This bound on the value of the criterion function for a partial solution may indicate that no optimal solution will be found that contains the particular partial solution and therefore search on the path may be terminated. Alternately, the calculation may show that the partial solution could be contained in an optimal solution to the problem and that the path (solution) should be developed further.

Two truncated enumeration schemes for the DCPM problem discussed in Chapter III will be presented here and experimental results for one of the schemes will be given. The first approach, the "reduced constraint" algorithm, assumes that the set of precedence constraints are reduced by the dominance, feasibility and lower bound tests of Chapter IV. The second approach, the "fixed order" algorithm assumes only that project information such as job times and precedence relations are available.

2. Reduced Constraint Algorithm

This algorithm will be based on the assumption that a reduced set of path constraints has been determined from the problem to be solved. For the numerical example of Chapter IV the resulting constraints are shown by Table VI-1. If, for example, iob $S_{9,3}$ is performed, then the minimum length of the critical path would be 45 days. Since the lower bound on project length calculated in Chapter IV was 43 days, the reduced constraint matrix assumes that W_f will be at least 43 days.

The problem we must solve then is

(1) Minimize $\sum_{i=1}^{m} \sum_{j=1}^{k(i)} c_{ij} d_{ij} - pW_f^- + rW_f^+$

Subject to

Precedence: As given in the reduced constraint matrix

Due Date: $W_f - W_f^+ + W_f^- = D$

$$\text{Interdependency:} \quad \sum_{j\ 1}^{k(i)} d_{ij} = 1 \qquad\qquad i = 1,\ldots,h$$

plus "other" interdependency constraints.

Node	6,1	6,2	6,3	9,1	9,2	9,3	12,1	12,2	15,1	15,2	17,1	17,2	W_f
Cost	400	200	0	200	100	0	50	0	100	0	0	150	V_1
													43
						1							45
							1						46
						1			1				46
	1						1						47
						1				1		1	48
			1				1						52
						1				1	1		56

Reduced Constraint Matrix

Table VI-1

The bounding process involves a separation of the full problem into two serially related subproblems. We first solve to find the minimum total job cost at a particular point in the tree, considering only decision jobs accepted, $d_{ij} = 1$, and decision jobs excluded, $d_{mn} = 0$, to that point in the tree. If S_A is the set of jobs accepted and S_E the set included

(2) $$C_J = \sum_{i=1}^{h} \sum_{j=1}^{k(i)} c_{ij} \; d_{ij}$$

Subject to

ACCEPTANCE $d_{ij} = 1$ $\qquad S_{ij} = \{S_A\}$

EXCLUSION $d_{mn} = o$ $\qquad S_{mn} = \{S_E\}$

Interdependency $\displaystyle\sum_{j=1}^{k(i)} d_{ij} = 1$ $\qquad i = 1, \ldots, h$

plus "other" interdependency constraints.

An optimal solution to this problem is the selection of a set of the decision jobs which includes all of the jobs in S_A, none of the jobs in S_E meets all interdependency constraints and minimizes job cost C_J. If no such solution exists, then that point of the tree may be labelled as infeasible and no further search is required. If a solution is found, then we can state that no set of decision jobs can be selected, given existing constraints, that will have a lower total job cost than C_J.

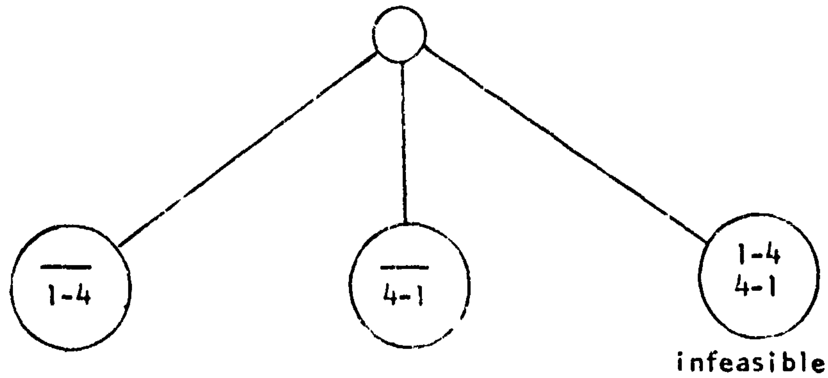
We have now calculated minimum job cost at a particular point in the tree and we now show that it is possible to add to it C_p, which is a lower bound on penalty cost for the date of project completion. Given the constraints implied by the jobs in S_A and S_E and given the information from Table VI-1, it is possible to calculate a minimum length for the critical path of the project. For example, if S_A contains job $S_{9,3}$ and S_E contains job $S_{15,2}$, then we must do jobs

$S_{9,3}$ and $S_{15,1}$ ($\sum_{j=1}^{2} d_{15,j} = 1$). From Table IV-1 we see that t· se two jobs introduce a path 46 days long into the network. Thus we can calculate a lower bound, C_p, of due date penalty or premium. Thus the lower bound on total cost, C_T, for the partial solution is $C_J + C_p$. In addition, if it is found that the complete solution to the job cost minimization problem also gives a path length resulting in cost C_p, we may terminate further search down the p·th. The optimum solution containing the partial path has been found since we have a complete solution with minimum job cost and path length, given our previous selections.

The general branching strategy to be used here is as follows. From the set of all decision paths, the set of paths with the maximum number of accepted and excluded jobs, P_p, will be taken. For each of these path P_p a minimum bound will be calculated and the path with the minimum lower bound will be selected for further elaboration. Once the path has been chosen, a decision rule is required to select decision jobs that will be examined at the next stage. The general approach to be used here is similar to that used by Eastman [27] and later by Shapiro [67] in the travelling salesman problem. At each node of the problem they solve an assignment problem on the cost matrix. The solution that results may imply several sub-tours, rather than a complete all-city tour. For example in a five city problem the solution

1-4, 4-1, 2-3, 3-5, 5-2

contains two subtours, 1-4-1 and 2-3-5-2. Shapiro takes the smallest of the tours, i.e. 1-4, 4-1 and branches on these variables



In this notation (1-4) implies that the route selected will include a trip from city 1 to city 4 and ($\overline{1\text{-}4}$) implies that such a trip will not be made. Since the combination of 1-4, 4-1 constitutes a subtour, by problem definition they both cannot appear in a feasible solution. Thus either the condition "not 1-4" or "not 4-1" (or both) must hold. If it is possible to close all nodes leading from $\overline{1\text{-}4}$ and $\overline{4\text{-}1}$, then all solutions have been discovered and bounded.

This suggests the following branching strategy for out problem. At a particular node of the tree, solve the following integer programming problem as in (2)

$$\text{Minimize} \quad C_j = \sum_{i=1}^{h} \sum_{j=1}^{k(i)} c_{ij}\, d_{ij}$$

Subject to

Acceptance $\quad d_{ij} = 1 \qquad S_{ij} = \{ S_A \}$

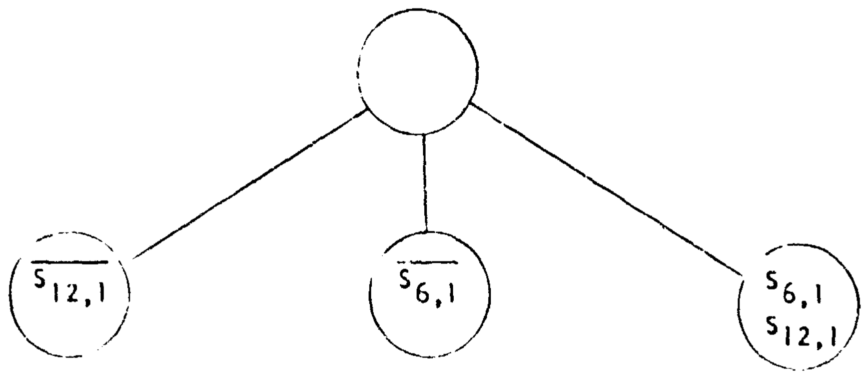
Exclusion $\quad d_{mn} = o \qquad S_{mn} = \{ S_E \}$

Interdependency

$$\sum_{j\ 1}^{k(i)} d_{ij} = 1$$

plus "other" interdependency constraints.

A solution to this problem will consist of the cheapest feasible selection of the decision jobs. It is then possible to test in the original decision graph for the length of the critical path associated with this selection of decision jobs. Alternately, if the reduced set of path constraints exist, for example see Table VI-1, the length of the critical path may be determined directly from that set of equations. In addition, we may identify those variables on the critical path of the network. These critical path variables then form one branch from the new node, and each variable, individually excluded, forms the alternate paths from the node. Any solution to the problem must contain either the set of all decision nodes on the original critical path, or have at least one of them excluded. If, for example, at a particular node on the tree, the cheapest feasible set of decisions contained $S_{6,1}$ and $S_{12,1}$ and both of these jobs were on a critical path of 47 days, we would branch as shown below



In the application of the idea to the DCPM problem, the combination of jobs on the critical path, say $S_{6,1}$, $S_{12,1}$ is feasible rather than infeasible as in Shapiro's problem. However, although it is feasible, it is never necessary to develop that particular node further. Given the exclusions and acceptances to that point on the path, we solved a programming problem to minimize the total job cost, C_J. The minimum cost feasible set of jobs is then tested in the matrix of Table VI-1 to determine project length, project length cost C_p, and the set of jobs on the critical path. Given that we do the set of jobs on the critical path and that prior acceptances and exclusions be enforced, the total cost can not be less than $C_p + C_j$ and the solution that gives C_p C_j is a feasible solution to the problem. Thus the node is completely developed.

If at any stage of the development of the tree, all new paths generated are bounded by a previous complete solution or are infeasible or if a complete solution has been obtained, it is necessary to move backwards up the tree and pick new paths to develop. Again we find the unbounded paths that have the maximum number of job acceptances and exclusions and of these select the one with the minimum bound. This implies that we move up the tree to the first open node. The algorithm will now be presented in flow chart form, then applied to the sample problem of Figure III-1.

Step 1. Reduce the network by the dominance feasibility and lower bound tests of Chapter IV to obtain the reduced constraint matrix.

Step 2a. Solve an integer programming problem to find the minimum cost set of decision jobs that will meet all interdependency constraints. Set the value of C_p to equal the cost of this solution. Go to Step 3.

Step 2b. Select the complete solution and the job cost C_J calculated at Step 7 for the path to be elaborated.

Step 3. Calculate the length of the critical path, W_f, associated with the decision jobs selected at Step 2. This may be determined either from the original project graph or from the reduced set of constraints. Let $C_p = -rW_f^- + pW_f$ given $W_f - W_f + W_f^- = D$. Thus C_p is either the early premium or late penalty associated with finish day W_f and due date D. Now establish one branch from the first node which includes the complete solution of Step 2, that is a particular set of decision jobs, with a total project cost of $C_T + C_J = C_p$.

Step 4. Establish alternate branches from the current node by adding nodes which specifically exclude each of the decision jobs on the critical path determined at Step 3.

Step 5. For each excluded decision job, S_{ij}, solve an integer programming problem, (2), which minimizes the total sum of job cost C_j, considering all jobs excluded to that point on the path. The problem may have no solution and, if so, the path may be immediately terminated.

Step 6. For each excluded decision job, S_{ij}, use the job alternative interdependency constraints $\sum_{i=1}^{k(j)} d_{ij} = 1$ to see if the series of exclusions have forced the "acceptance" of some decision jobs. Given the "acceptances", the original project graph on the reduced set of path constraints will enable us to calculate a minimum bound on the length of the critical path, W_f. Use W_f to calculate a lower bound on completion date cost, C_p. For each decision job excluded, add the job cost of Step 5, C_j, to the minimum completion date cost calculated above, C_p, to determine a minimum bound on total cost $C_T = C_J + C_p$.

Step 7 Record the full solution to the programming problem of Step 5 and calculate the critical path length associated with each solution.

Step 8. Test all paths from the last node for feasibility, complete solution or bounded solution.

(a) If the program of Step 5 has no solution, then the path being tested is infeasible and no further search on that path is required.

(b) If for any path the bound of Step 6 is equal to the complete solution cost of Step 7, then we have a complete solution to the problem and no lower cost will be found down this particular path. If the total cost of the full solution is less than the cost of the existing best solution, update the existing best with the new value.

(c) If the lower bound calculated in Step 6 is higher than an existing complete solution to the problem, the path is bounded and need not be considered further.

Step 9. Choose from the feasible, unbounded partial paths at the current node the path with the minimum cost bound. If a tie exists, choose the path with the maximum number of forced acceptances. Go to Step 2b.

If no feasible, unbounded partial path exists, then the node is closed and we backtrack one level in the graph. Go to Step 8.

If all nodes in the network are closed, then the problem is solved and the current best solution is the optimal solution.

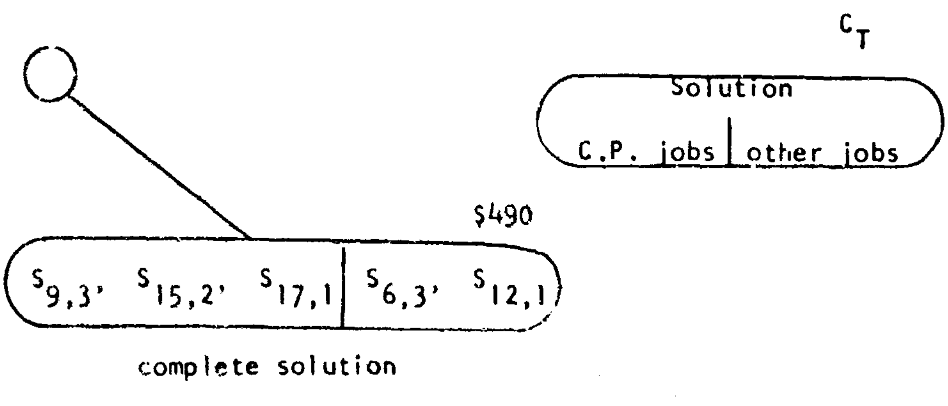
3. Application of the Reduced Constraint Algorithm

The steps of the algorithm will now be illustrated with the DCPM problem of Figure III-1 with $D = 45$ days, $r = \$20$ and $p = \$40$. The complete solution tree for the problem is shown in Figure VI-2.

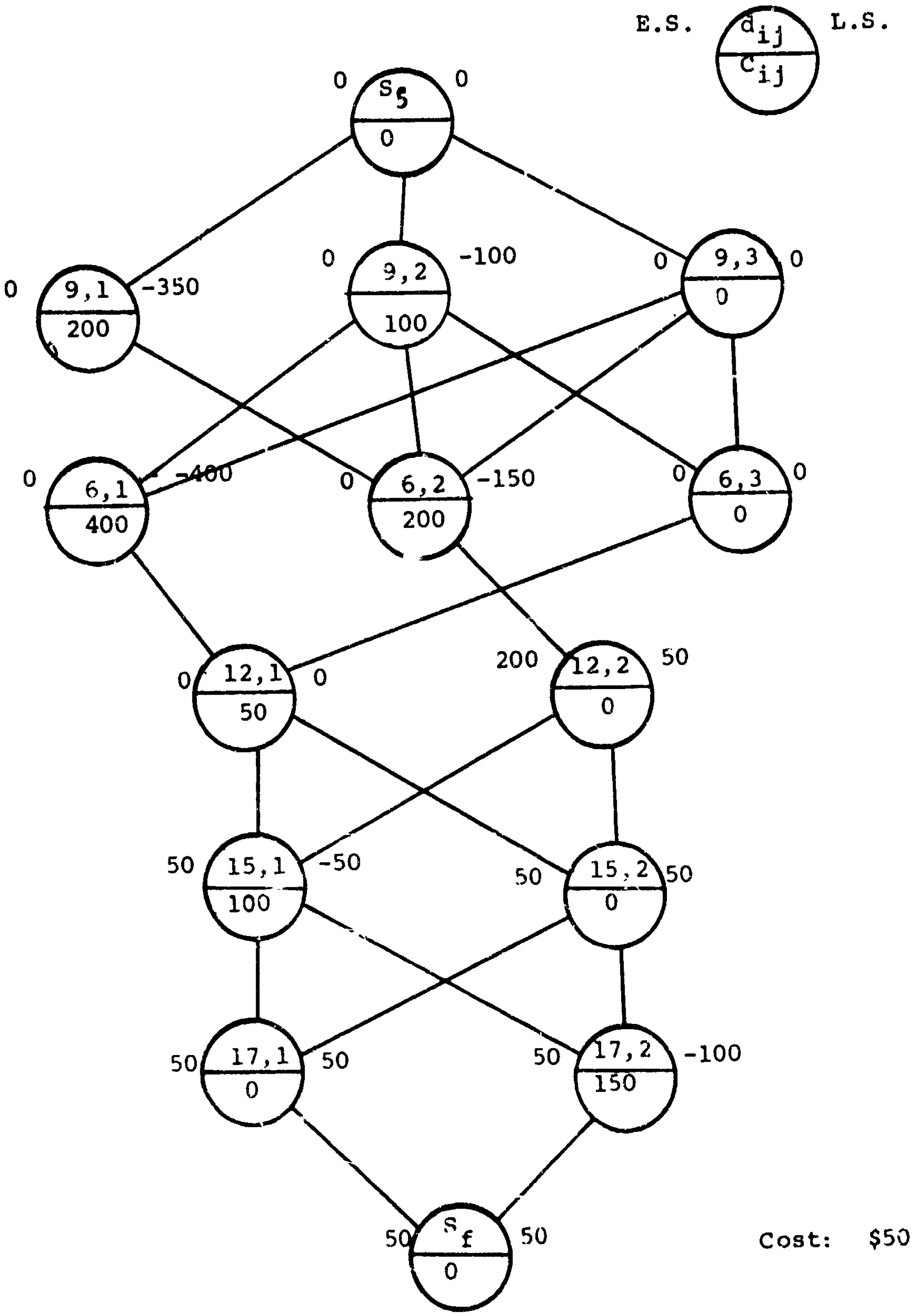
Step 1. Table VI-1 illustrates the reduced set of constraints for the sample problem.

Step 2. The integer programming algorithm of Chapter V will be used to solve the problem of selecting the minimum cost, feasible set of decision jobs. Figure VI-1 shows the initial programming network modified to include the "other" interdependency constraints. The minimum cost solution is $S_{6,2}$, $S_{9,3}$, $S_{12,1}$, $S_{15,2}$ and $S_{17,1}$ with a cost of $C_J = \$50$.

Step 3. With the jobs given in Step 2, we find from Table VI-1 that the critical path is 56 days long and it contains decision jobs $S_{9,3}$, $S_{15,2}$, and $S_{17,1}$. $C_p = (56 - 45)(40) = \$440$. The total cost for the solution $C_T = 50 + 440 = \$490$.



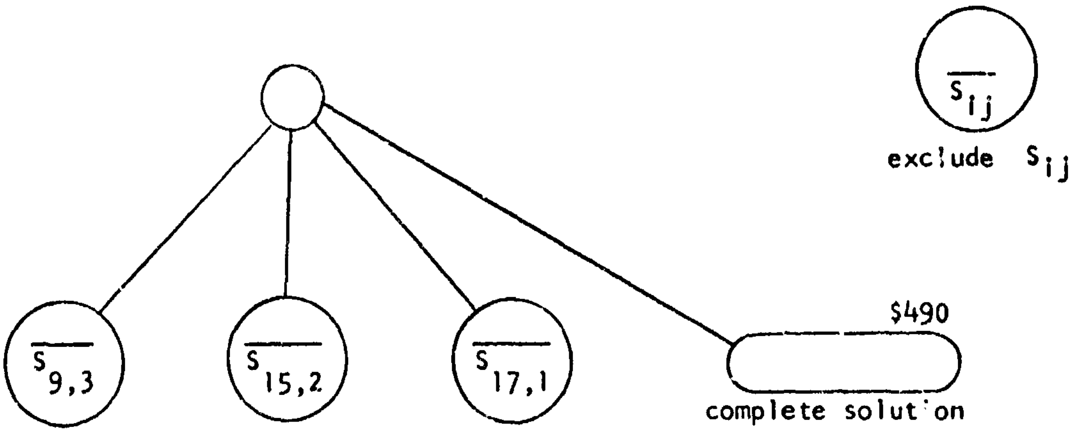
Step 4. We now establish alternate branches excluding all jobs on the critical path of Step 4.



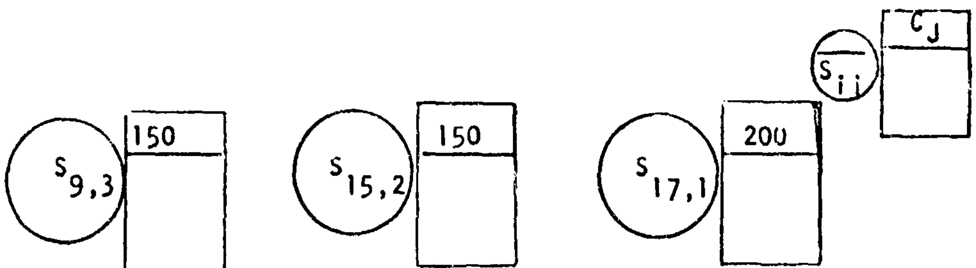
Solution: $S_{9,3}$ $S_{6,3}$

$S_{12,1}$ $S_{15,2}$ $S_{17,1}$

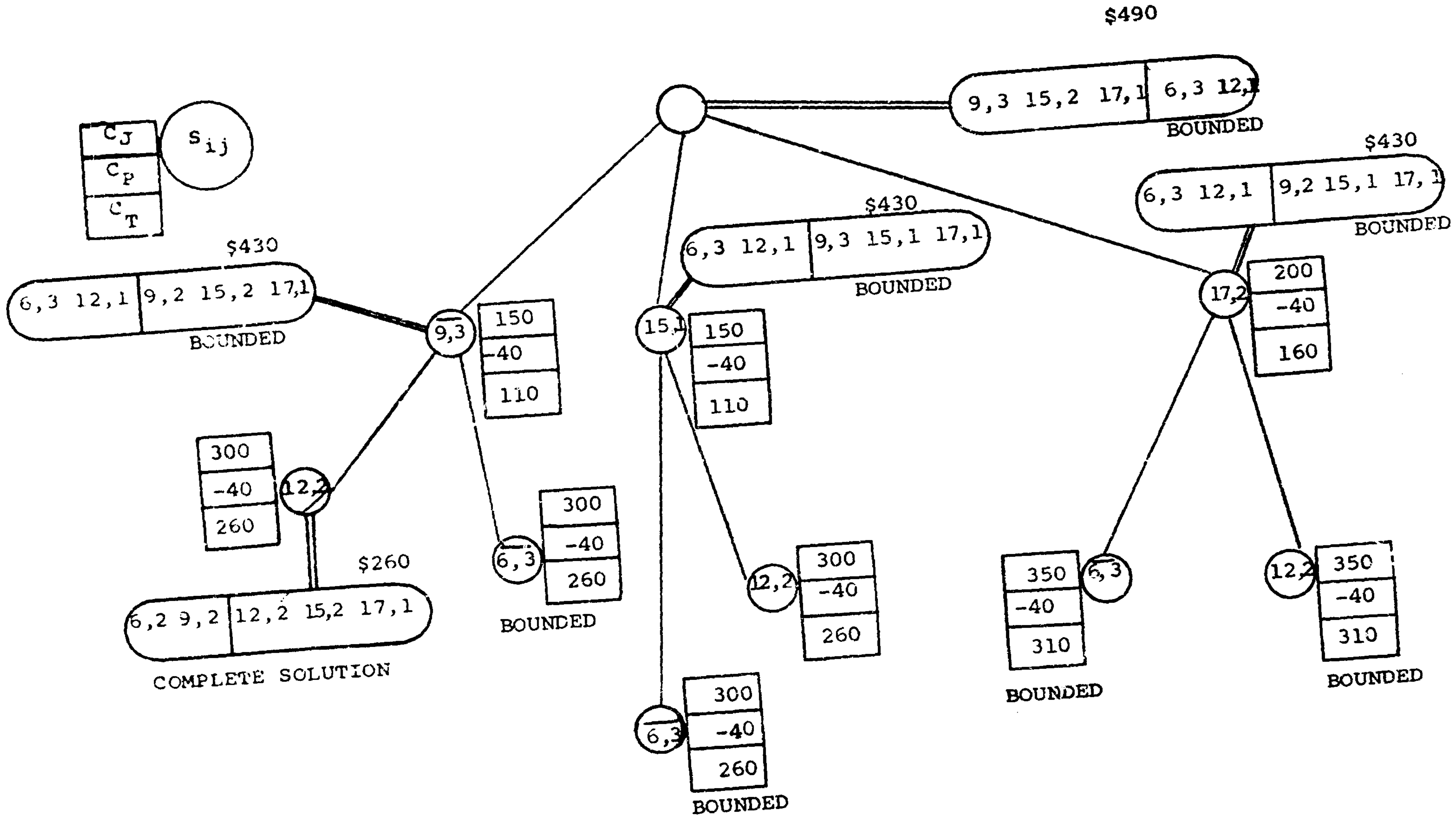
FIGURE VI-1



Step 5. For each excluded job, $\overline{S_{ij}}$, we determine the remaining minimum cost, C_J, feasible solution. The values would be obtained by removing the node $\overline{S_{ij}}$ from the graph of Figure VI-1 and recalculating the shortest path.

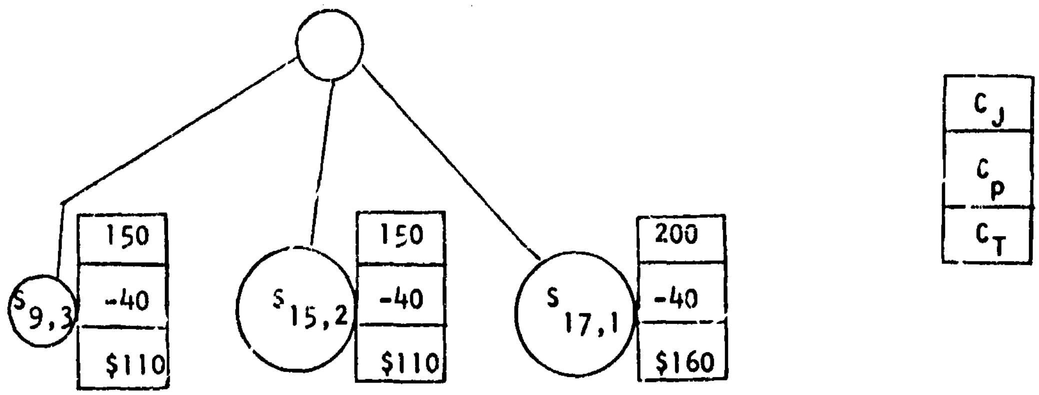


Step 6. For each exclusion calculate the minimum bound for the length of the critical path. If we exclude $S_{9,3}$, then we force the acceptance of no job and therefore the minimum length of the critical path would be the absolute minimum, 43 days. This gives a reward of $(45 - 43)(20) = -\$40$. If we reject $S_{15,2}$, then we are forced to accept $S_{15,1}$. However, no path that includes only decision job $S_{15,1}$ is longer than 43 days. Therefore the minimum completion cost is again -\$40.



$490
9,3 15,2 17,1 | 6,3 12,1
BOUNDED
$430
6,3 12,1 | 9,2 15,1 17,1
BOUNDED
$430
6,3 12,1 | 9,3 15,1 17,1
BOUNDED
$430
6,3 12,1 | 9,2 15,2 17,1
BOUNDED
$260
6,2 9,2 | 12,2 15,2 17,1
COMPLETE SOLUTION
s_{ij}
C_J
C_P
C_T
9,3
150
-40
110
12,2
300
-40
260
6,3
300
-40
260
BOUNDED
15,1
150
-40
110
12,2
300
-40
260
6,3
300
-40
260
BOUNDED
17,2
200
-40
160
6,3
350
-40
310
BOUNDED
12,2
350
-40
310
BOUNDED

FIGURE VI-2



Step 7. Compute the complete cost for each node developed in Step 5.

$\overline{S_{ij}}$	Solution	C_j	W_f	C_p	C_T
$\overline{S_{9,3}}$	$S_{6,3}$, $S_{9,2}$, $S_{12,1}$, $S_{15,2}$, $S_{17,1}$	150	52	280	430
$\overline{S_{15,2}}$ $(S_{15,1})$	$S_{6,3}$, $S_{9,3}$, $S_{12,1}$, $S_{15,7}$, $S_{17,1}$	150	52	280	430
$\overline{S_{17,1}}$ $(S_{17,2})$	$S_{6,3}$, $S_{9,3}$, $S_{12,1}$, $S_{15,2}$, $S_{17,2}$	200	52	280	480

Step 8. All three paths $\overline{S_{9,3}}$, $\overline{S_{15,2}}$, $\overline{S_{17,1}}$ are feasible and all but $\overline{S_{17,1}}$ are unbounded. The minimum cost solution available costs $430.

Step 9. Select node $S_{9,3}$ for elaboration since it is tie with $S_{15,2}$ for minimum bound.

The final solution as shown in Figure VI-2 contains jobs $S_{6,2}$, $S_{9,2}$, $S_{12,2}$, $S_{15,2}$ and $S_{17,1}$ with costs, $C_j + C_p = C_T$, $300 - 40 = \$260$.

4. Fixed Order Algorithm

The second scheme proposed works directly from the matrix of Table IV-3, the reduced network matrix, which shows maximal distances between decision jobs in a decision graph. For convenience this table will be reproduced as VI-2.

Reduced Network Matrix

	6,1	6,2	6,3	9,1	9,2	9,3	12,1	12,2	15,1	15,2	17,1	17,2	S_f
S_s	16	18	20	22	22	27	36	32	9	13	11	3	37
6,1							21	17					17
6,2							19	15					15
6,3				8			22	18					23
9,1													20
9,2													18
9,3									9	13			18
12,1													10
12,2													10
15,1													10
15,2											11	3	0
17,1													5
17,2													5

Table VI-2

It contains implicitly a reduced set of project paths that exist in the original network Figure III-1 . In this algorithm the order of selection of decision sets is strictly determined, that is for our problem an ordering of decision set S_6, S_9, S_{12}, S_{15} and S_{17} might be chosen. In elaborating a particular decision set, the paths to be considered are simply all possible decision jobs in the set under consideration. At S_i we consider branching on all S_{ij} , $j = 1,2, \ldots, k(i)$. This involves a feasibility test to see if the new job S_{ij} is consistent with jobs previously chosen on the path given the "other" interdependency constraints. A feasible path is then selected for elaboration based on a calculation of a total cost bound for each feasible path. As in the first algorithm, the path selected will always be from the most fully developed paths so that we always push directly down to a complete solution or halt because a path is bounded, then backtrack up the tree closing open nodes as we find them.

At any node in the tree it is possible to calculate directly the total cost of decision jobs assigned on the path to that point. If we add to this the total cost of minimum cost jobs in decision sets from which no assignment has been made, we have a lower bound on total job cost C_J. In our problems the costs for jobs in each decision set have been normalized, that is the smallest cost is subtracted from all costs, so that the total minimum cost for unassigned sets will always be zero.

At the node under consideration, it is also possible to calculate a minimum length for the decision path under consideration. A project graph can be constructed from the information of Table VI-2 as we have done in Figure VI-3. The reduced graph is broken at all unassigned decision nodes, and the longest path algorithm is applied to find the shortest possible length of the network given previous assignments. This value is a lower bound on project length and may be used to calculate C_p -- minimum completion cost. $C_T = C_J + C_p$ gives a lower bound for the particular path under consideration.

The steps of the algorithm are as follows.

Step 1. Normalize the cost of all jobs in decision sets by subtracting the lowest cost for any mob in a set from all jobs in that set.

Step 2. Apply the routine of Chapter IV, Section 3, to find a reduced network containing only decision jobs and maximal length paths between decision jobs.

Step 3. Sequence the decision sets in a fixed order.

Step 4. Elaborate the first decision set in the fixed order list, for which no assignment has been made and calculate total job cost, C_J, for each resulting path. The integer programming algorithm of Chapter V will find the minimum cost solution, given interdependency constraints and decision job acceptances. If no feasible solution exists, then search on this path may be terminated.

Step 5. For each newly developed path, find a minimum bound for the length of the critical path, W_f, and evaluate the cost of this path length as C_p. The reduced network from Step 2 may be used to compute minimum project length. As in the algorithm of Chapter IV, Section 3, we break the reduced network at all decision jobs, then introduce only those decision jobs accepted on the partial path which is under consideration. A critical path calculation which ignores all unaccepted decision jobs then gives a lower bound on project length. A lower bound on the cost of the partial path will then be $C_T = C_J + C_p$.

Step 6. Record the full solution to the programming problems of Step 4 and calculate the critical path length associated with each full solution using the methods of Step 5. Compute the total cost of each solution.

Step 7. Test each newly developed path for feasibility, complete solution and bounded partial solution.

(a) A path may be excluded based on a feasibility test of Step 4.

(b) If for any path the cost of a complete solution from Step 6 is equal to the lower bound calculated in Step 5, no further development is necessary. No other solution on this path can have a lower total cost. If the total cost of the full solution is less than the cost of the existing best solution, update the existing "best" with the new value.

(c) If the lower bound of Step 5 is higher than an existing complete solution to the problem, the path is bounded and need not be considered further.

Step 8. (a) Choose from the feasible unbounded partial paths at the current node, the path with the minimum cost bound. Break ties with random selection. Go to Step 3.

(b) If no feasible, unbounded, partial path exists, then we backtrack one level in the graph. Go to Step 8(a).

(c) If no open nodes are found, HALT. The optimal solution is the current best solution.

5. Application of the Fixed Order Algorithm

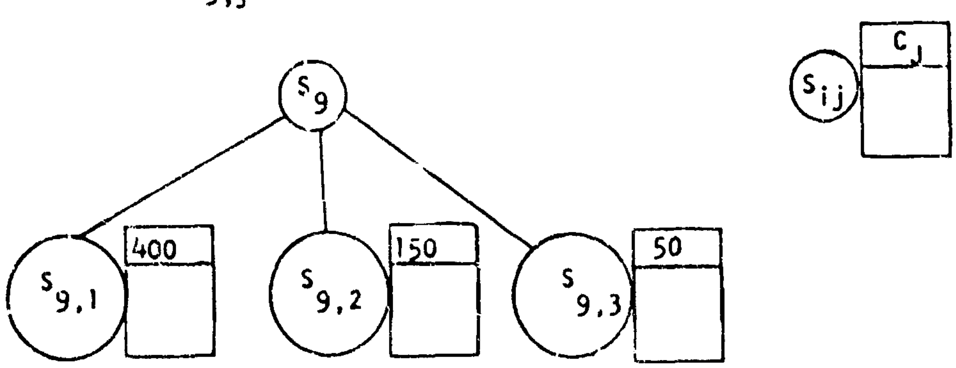
The steps of the algorithm will now be illustrated with the DCPM problem of Figure III-1 with $D = 45$ days, $r = \$20$ and $p = \$40$. The complete solution tree for the problem is shown in Figures VI-4, 5.

Step 1. The job costs in this problem are normalized.

Step 2. Table VI-2 and Figure VI-3 illustrate the reduced network.

Step 3. Choose sequence S_9, S_6, S_{12}, S_{15}, S_{17} although any sequence is permissible.

Step 4. Elaborate the three alternatives for decision job S_9 and for each calculate C_j the minimum job cost for a feasible solution containing $S_{9,j}$.



Step 5. Calculate the minimum path length associated with each partial path. From Figure VI-3, we can determine the longest path through $S_{9,1}$ and only $S_{9,1}$ is 42 days long. Similarly, bounds on $S_{9,2}$ and $S_{9,3}$ may be calculated as 40 and 45 respectively. The associated values for C_p would be -\$60, -\$100 and 0. Therefore, the total

REDUCED NETWORK

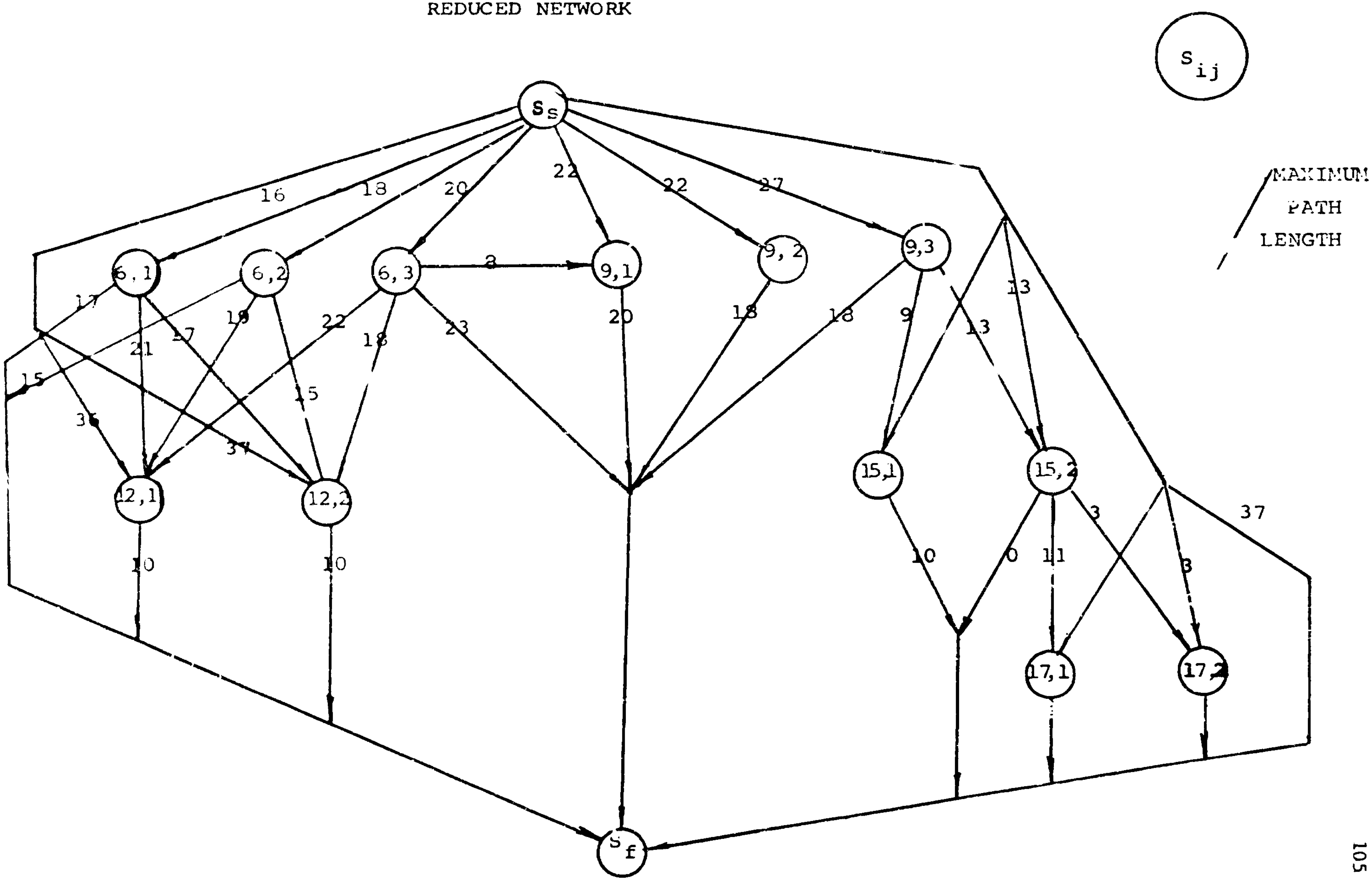
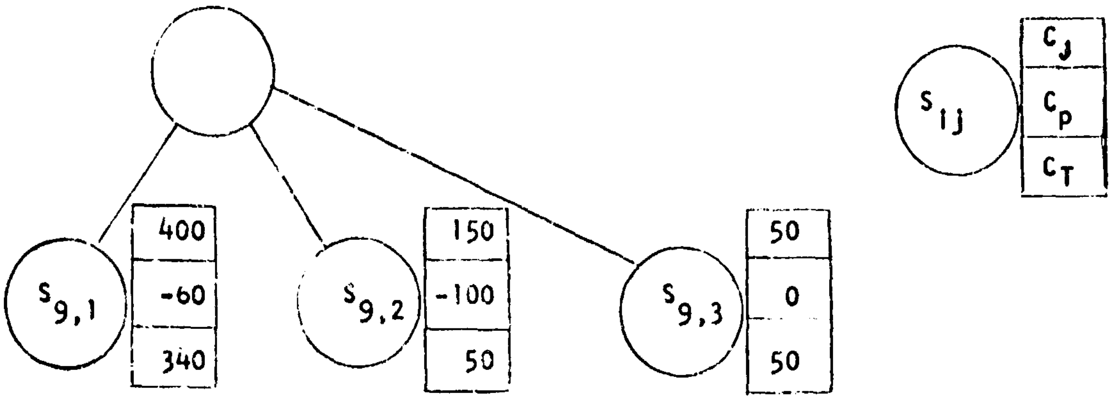


FIGURE VI-3

lower bounds, C_T, are \$340, \$50 and \$50.



Step 6. The complete solutions of Step 4 are

Acceptance	Solution	C_J	W_f	C_p	C_T
$S_{9,1}$	$S_{6,2}$, $S_{9,1}$, $S_{12,2}$, $S_{15,2}$, $S_{17,1}$	400	43	-40	360
$S_{9,2}$	$S_{6,3}$, $S_{9,2}$, $S_{12,1}$, $S_{15,2}$, $S_{17,1}$	150	52	280	430
$S_{9,3}$	$S_{6,3}$, $S_{9,3}$, $S_{12,1}$, $S_{15,2}$, $S_{17,1}$	50	56	0	490

Step 7. Choose either $S_{9,2}$ or $S_{9,3}$ for further elaboration since they have identical lower bounds of \$50.

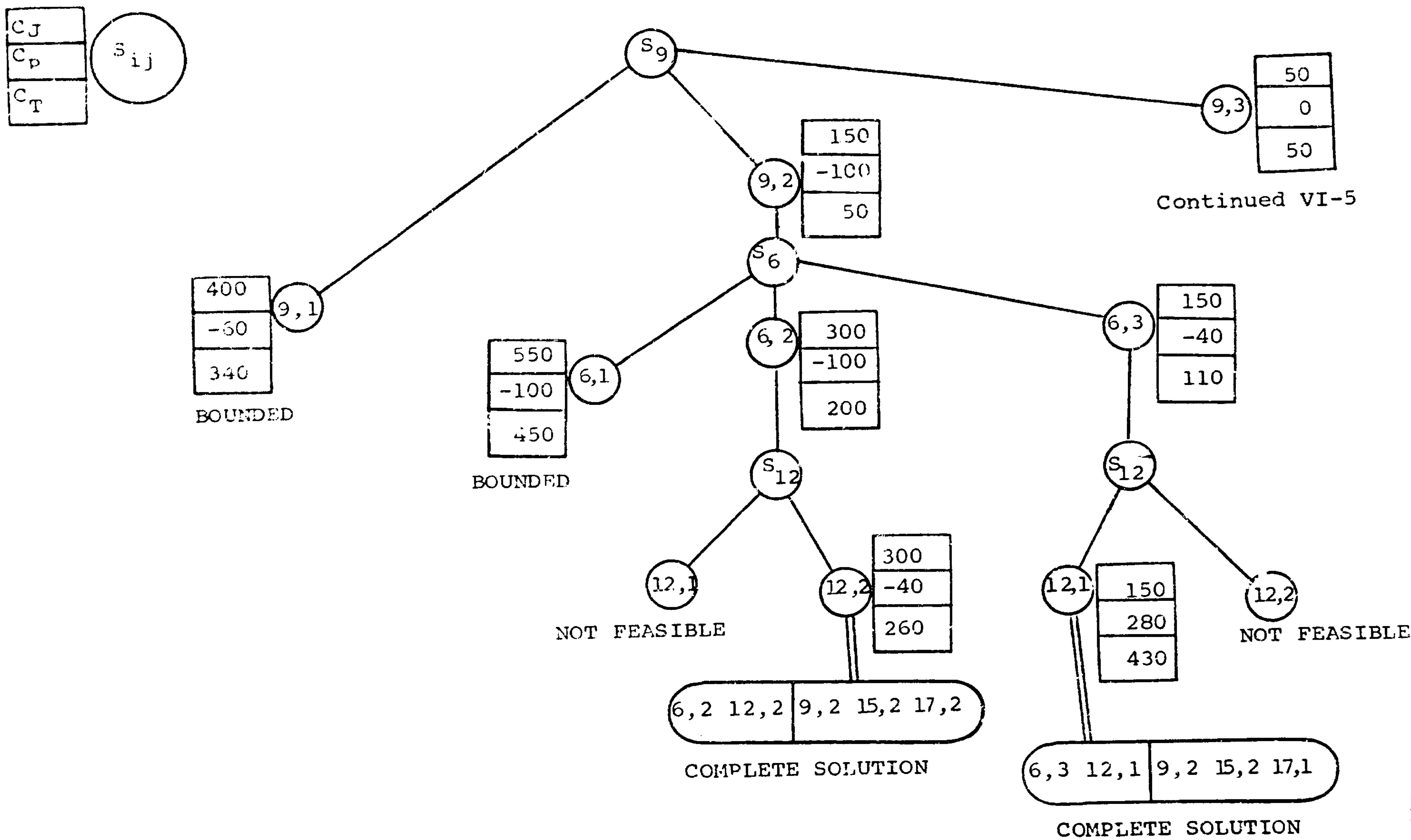
The optimal solution is shown in Figure VI-4 to be $S_{6,2}$, $S_{9,2}$, $S_{12,2}$, $S_{15,2}$ and $S_{17,1}$ with a total cost of \$260.

6. Computational Results with the Fixed Order Algorithm*

The efficiency of this algorithm for any given problem will depend on the fixed order established for the decision sets. In our

---

* The computational results reported in this section were obtained with the collaboration of M. Wagner. The details of four competitive algorithms, along with computional results, are given in [23]. The programs used are listed in Appendix E.



C_J
C_p
C_T
S_{ij}
S_9
9,1
400
-60
340
BOUNDED
9,2
150
-100
50
9,3
50
0
50
Continued VI-5
S_6
6,1
550
-100
450
BOUNDED
6,2
300
-100
200
6,3
150
-40
110
S_{12}
12,1
NOT FEASIBLE
12,2
300
-40
260
6,2 12,2 | 9,2 15,2 17,2
COMPLETE SOLUTION
S_{12}
12,1
150
280
430
12,2
NOT FEASIBLE
6,3 12,1 | 9,2 15,2 17,1
COMPLETE SOLUTION

FIGURE VI-4

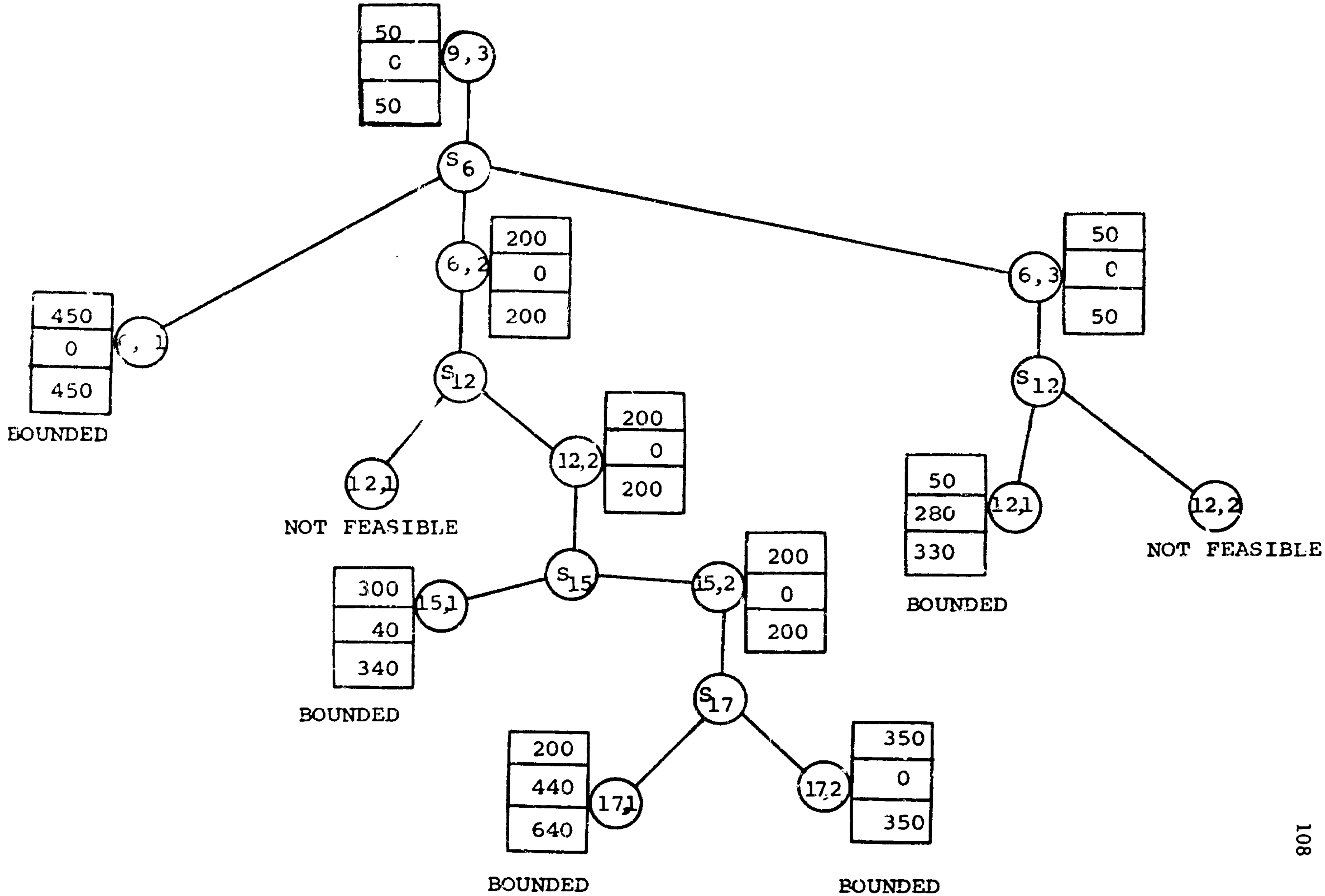

50
C
50
9,3
S6
200
0
200
6,2
450
0
450
,1
BOUNDED
S12
12,1
NOT FEASIBLE
12,2
200
0
200
S15
15,1
300
40
340
BOUNDED
15,2
200
0
200
S17
17,1
200
440
640
BOUNDED
17,2
350
0
350
BOUNDED
6,3
50
C
50
S12
12,1
50
280
330
BOUNDED
12,2
NOT FEASIBLE

FIGURE VI-5

example, the fact that S_9, S_6, and S_{12} appeared at the head of the list coupled with the interdependency constraints on these job sets, allowed substantial truncation based on feasibility tests. We may also observe from this example that a combination of $S_{6,3}$ and $S_{12,1}$ always gave a good project completion cost bound since they allowed a long path to form in the network (52 days). This, of course, could be observed directly in Table VI-1. This suggests that we might cut search time if decision sets that appeared on long paths were placed at the front of the fixed order list. This would certainly be in the spirit of the first algorithm presented. In order to measure criticality of the decision sets, the following procedure was used.

Step 1. At each decision set, the cheapest decision job was chosen.

Step 2. The resulting CPM problem is solved and the slack for each job calculated.

Step 3. Sequence the decision sets in increasing order of slack measured for the member of that set in the CPM problem.

The algorithm was tested on 10 problems of approximately 210 jobs and 15 three-job decision sets each. The total number of combinations of decision jobs would therefore be $3^{15} \approx 1.5 \times 10^6$. Characteristics of the 10 jobs are given in Appendix A.

Each problem was solved twice, the first time with a simple technological ordering of decision sets, and the second time with the slack ordering described above. Computational results are given in Table VI-3. The algorithm was programmed in Fortran IV for an IBM 7094. The program was run under a time-sharing system -- the time reported derived from the system interval timer and do not include swap times imposed by the time-sharing system. Times required to obtain the reduced network matrix are not included here, but reported separately in Appendix B. In addition to the times reported, these problems required 2-6 seconds for reading data and performing certain initializing functions in preparation for the application of the branch and bound procedure. The computation times reported are exactly repeatable.

Computation results show an impressive superiority for choosing decisions based on initial slack. Ratios of computation time as high as 50:1 were found. In addition, the absolute amount of time required by the algorithm using a slack ordered list suggests that the algorithm is efficient for these problems.

Problem Number (Appendix A)	Ordering Heuristic	Optimal Solution Found	Optimal Solution Proven
		Computation time (sec.)	
78	Technological	330-T	330-T
	Slack	6.4	9.8
79	Technological	73.7	213-T
	Slack	16.4	36.8
80	Technological	51.0	157-T
	Slack	13.8	23.5
81	Technological	139.4	261-T
	Slack	101.3	130.5
82	Technological	7.7	141-T
	Slack	12.8	26.6
83	Technological	14.3	18.5
	Slack	1.8	2.2
84	Technological	40.8	73.2
	Slack	5.8	9.4
85	Technological	2.2	5.6
	Slack	2.0	2.4
86	Technological	98-T	98-T
	Slack	2.2	2.4
87	Technological	81.0	95-T
	Slack	1.6	1.6

T - computation terminated before completion

Table VI-3

Chapter VII

RESOURCE CONSTRAINED DECISION NETWORKS

The decision networks we have considered in previous chapters have included precedence and interdependency constraints. To evaluate a particular design, that is a particular solution to the problem, it was simply necessary to calculate the length of the critical path for the resulting network. Then the project cost could be determined exactly. If resource constraints are added to this problem, it is no longer a simple matter to find the minimum length of the project. Branch and bound techniques have been proposed to solve the problem exactly, but for problems of reasonable size, the computation time is excessive. Johnson [42] reports that a 100 task, single resource problem ran 79 minutes without proving an optimal solution. In a five decision set problem, we would have $3^5 = 243$ possible solutions to evaluate in this manner.

For this reason it was decided to evaluate proposed solutions to the decision problem by loading them under the limited resource with a "good" heuristic rule. The heuristic to be used should be efficient so as to keep computation time within reasonable limits, but it should provide tight schedules. In this chapter we will examine nine possible heuristic loading rules and choose the best of them for use in solving resource constrained decision models. We then examine three techniques for choosing an optimal set of decisions. These include complete enumeration, pairwise switching and multiple pairs switching.

2. Project Scheduling Heuristics

A graphical technique for project scheduling was first proposed by Gantt [36]. In this technique each resource is shown as a bar on a bar chart where the horizontal dimension is time. Each task can then be identified as a rectangle of resource use continuing for a specific length of time. Precedence relations between tasks and a loading heuristics determine the relative position of the tasks in time and the completion date of the project. Given this visual display, it is possible, for small projects, to experiment with various sequences that satisfy the precedence constraints, so as to determine the optimal, that is minimal length, schedule.

For large projects this graphical technique is inefficient. Not only is it difficult to attempt many sequences, but it would require much time to keep job information up to date. Therefore, in most current applications, a computer model of the Gantt chart is maintained. With a computer system, it is relatively easy to up-date project information as jobs are completed, calculate job slacks and test various loading heuristics. As we have discussed in Chapter II, various heuristics have been proposed for the loading problem [?, 25, 43, 44], but there has been little comparison made of the effectiveness of these rules. The rules to be tested here are those that can be implemented quickly and that have been well regarded in previous work. They are all basically serial loading techniques. The jobs in the project are ordered first in technological order, then within technological order by a secondary measure. The loading routines take the first job on the list, schedule it at its early start date, then proceed down the ordered list scheduling each job in turn. The particular sequence of jobs in the list will, therefore, strongly influence the completion date of the project. Two routines which will be discussed below modify the serial loading slightly to allow a previously scheduled job to be shifted forward.

Nine rules are to be tested here. These consist of three basic job sequences within the technological order, each sequence loaded by three heuristic programs. The three orders to be tested are

1. random
2. increasing early start
3. increasing late start

The three serial routines, LOADC, LOADN and LOAD will now be presented.

LOADC takes each job as it appears on the technologically ordered list and places it in the schedule at the earliest possible time. The start time of any job will be constrained by the finish time of its predecessors and availability of resources. The flow chart for the routine is as follows.

1. Technologically order the jobs.

2. Solve for the critical path of the network. Find Early Start, Late Start and slack for each job.

3. Select the job sequence to be tested (random, E.S., L.S. within the technological order) and reorder the jobs.

4. Set $i = 1$ where i denotes a position on the ordered job list.

5. Calculate ES_i, the Early Start day of the job in the i_{th} position on the list. Since the list is technologically ordered, all predecessors of the job will be scheduled and Early Start can be calculated in the usual way.

6. Attempt to schedule the job in the i_{th} position on day ES_i. If the job cannot be scheduled because of insufficient resources, go to Step 8. If it can be scheduled, do so and set $i = i + 1$.

7. If all jobs have been scheduled, go to Step 9. If not, go to Step 5.

8. $ES_i = ES_i + 1$. Go to Step 6.

9. Halt.

Routines LOADN and LOAD are similar to LOADC except that jobs previously scheduled may be shifted forward to reduce the demand for resources on a given day. A flow chart for LOADN will now be presented.

1. Technologically order the jobs.

2. Solve for the critical path of the network. Find Early Start, Late Start and Slack, (SL), for each job.

3. Select the job sequence to be tested and reorder the jobs.

4. Set $i = 1$ where i denotes a position on the ordered job list.

5. Test to see if the job in the i_{th} position is scheduled. If it is scheduled, set $i = i + 1$ and go to Step 5. If it is not scheduled, go to Step 6.

6. Calculate ES_i, the Early Start day of the i_{th} job.

7. Schedule the job in the i_{th} position of the ordered list to begin on day ES_i. If more than the available resource are required, measure the excess resource required, then go to Step 9. If sufficient resources were avialable, set $i = i + 1$.

8. If all jobs have been scheduled, go to Step 15. If not, go to Step 5.

9. List all jobs scheduled to operate on day ES_i which use at least as much resource as the excess resource measured in Step 7. Of these jobs, select the one with maximum job slack, SL. Assume this job holds position j on the ordered list.

10. Remove from the schedule any successor of job j which has been scheduled.

11. Set $ES_j = ES_i + 1$.

12. Attempt to schedule job j on day ES_j. If the job cannot be scheduled because of insufficient resources, go to Step 13. If it can be scheduled, do so and set $i = j + 1$. Go to Step 8.

13. $SL_j = SL_j - 1$.

14. $ES_j = ES_j + 1$. Go to Step 12.

15. Halt.

In the LOADC routine discussed earlier, all resource conflicts were resolved by shifting the job currently being scheduled forward. LOADN modifies this so that given excessive demands for resource on a given day, the job with the greatest amount of slack was shifted forward. The routine LOAD which was tested is identical to LOADN in the flow chart above, except that Step 13 is omitted. This means that the slack of a job is not reduced as it is shifted forward.

The nine heuristic loading techniques tested are illustrated by the following matrix.

LOADING HEURISTIC

		LOADC	LOADN	LOAD
JOB ORDER	Random			
	L.S.			
	E.S.			

3. Experimentation with Project Scheduling Heuristics

Sixty-five projects were generated for this series of tests. These projects varied in size from 40 to 230 jobs, in length of the critical path from 54 to over 200 days, in scheduled length, given resource constraints, from 60 to over 600 days. Specific characteristics of individual projects are given in Appendix A. The operating results of the heuristics programs are given by problem number in Appendix C.

The results may be summarized briefly as follows. Of the nine rules tested, the LOADC code operating on a LS ordered list was clearly superior to all other heuristics. Tables VII-1 and VII-2 show that in 56 of the 65 problems tested this routine gave the shortest project completion time and that in 47 of the 56 cases no other routine found the minimum length schedule. Furthermore, Table VII-3 shows that on no occasion was the worst schedule generated by this particular combination of job ordering and loading routine. An examination of particular project results in Appendix C is interesting. The difference between the best and worst solution in many cases is as high as 30 percent to 50 percent of the best schedule achieved.

In general, the late start ranking is superior to either random with a technologically ordered list, or an E.S. ordered list. The LOADC loading method is superior to LOADN and LOAD for L.S. orderings, but the evidence is not so clear for either a random or an E.S. order. The test results clearly show that the use of LOADC loading routine on a list of jobs ordered by L.S. will provide good solutions, relative to the other heuristics tested. This method will now be used to evaluate alternate solutions to the resource constrained decision network problem.

	LOADC	LOADN	LOAD
RANDOM	4	4	4
L.S.	56	10	2
E.S.	4	3	2

Results of 65 projects

Number of "best" schedules

Table VII-1

	LOADC	LOADN	LOAD
RANDOM	0	1	0
L.S.	47	3	0
E.S.	2	0	1

Results of 65 projects

Number of unique best scheduling

Table VII-2

	LOADC	LOADN	LOAD
RANDOM	7	2	16
L.S.	0	1	20
E.S.	4	1	21

Results of 65 projects

Number of worst schedules

Table VII-3

4. Total Enumeration

The technique may be explained as the evaluation of all feasible combinations of the decision variables in a combinatorial problem. In general, the method is not useful because of the large amount of computer time required to solve problems. We use the method here on a series of 10 small problems in order to have some means of evaluating the heuristics we propose to use on the resource constrained decision network problem.

The problems to be tested have three active resource categories, 40-60 jobs, 5 decision sets of 3 jobs each. Thus, there are $3^5 = 243$ possible combinations of decision jobs to be tested in each problem. The routine iteratively activates a combination of five decision jobs, one from each job set, then calculates the critical path in the resulting project network, orders the jobs by late start and loads them under specified resource limits. The cost of the decision jobs in a particular combination and the resulting length of the project are recorded.

Table VII-4 reports the computation times for the complete enumeration routines programmed in Fortran IV and run under a time-sharing system on an IBM 7094. The times reported are derived from the system interval times and do not include swap times imposed by the time-sharing system. The best solution found is also reported.

As explained above, the schedule lengths are heuristically determined so that for any combination of decision jobs, we almost certainly are not reporting the optimal schedule. It is our point, however, that large problems of the type we are attempting to solve cannot practically be solved by existing algorithms. To illustrate this point, we note that since the running time for our 3^5 problem is approximately 129 seconds, a problem with 15 decision sets would require at least

$$\frac{3^{15}}{3^5} \cdot 129 \times \frac{1}{60\times 60} = 212 \text{ hours}$$

if we were to enumerate all possible solutions and solve them heuristically. This number can be multiplied by 30 to 100 if the project has 60-100 tasks and we wish an optimal loading for each design. Appendix D reports all undominated solutions and a distribution of critical path lengths and resource constrained schedule lengths for the projects.

We now report a technique that will substantially reduce the search required -- but, as will be explained, will not guarantee that the best set of decision jobs are selected.

5. Pairwise Interchange

The pairwise interchange technique is a method of partially enumerating combinations of variables in a problem. We will illustrate the method in terms of a simple DCPM problem. If the decision network contains three decision nodes S_1, S_2, S_3 with respectively three, two and two job alternatives, then Figure VII-1 shows an enumerated tree of all possible combinations of the variables.

A pairwise enumeration scheme would begin with one decision job from decision set S_1, one from S_2 and one from S_3. In our routine the job with the lowest cost in each set is selected as a starting solution. Assume the initial solution is $S_{1,1}$, $S_{2,1}$ and $S_{3,1}$. The pairwise interchange routine begins with one of the decision jobs, say S_1, and iterates it through all possible alternatives,

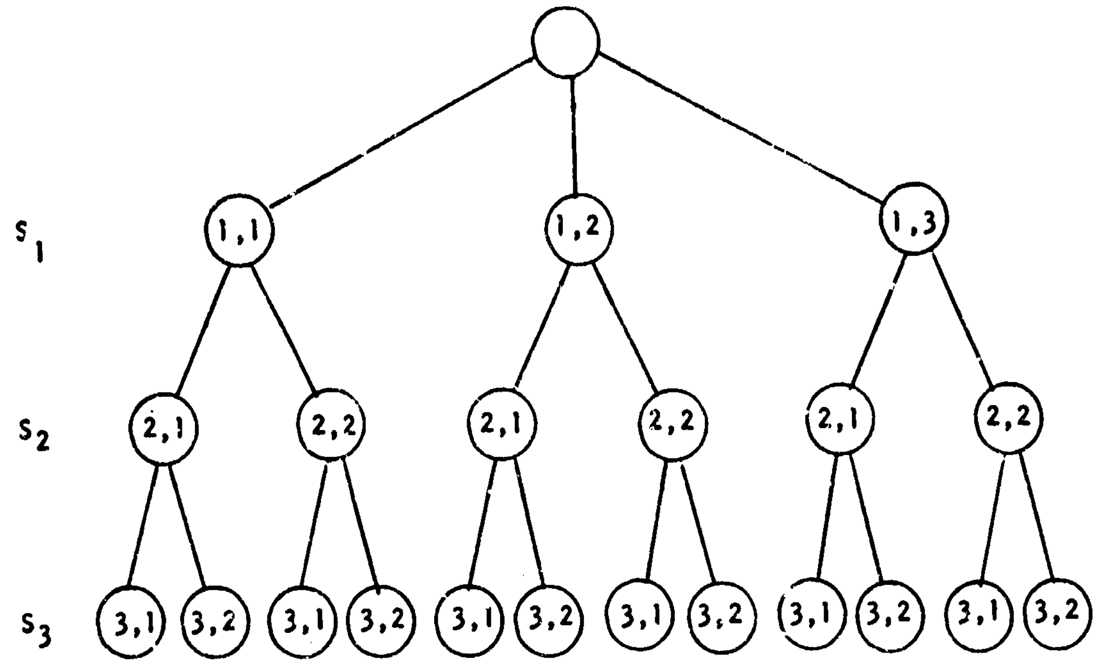


Figure VII-1

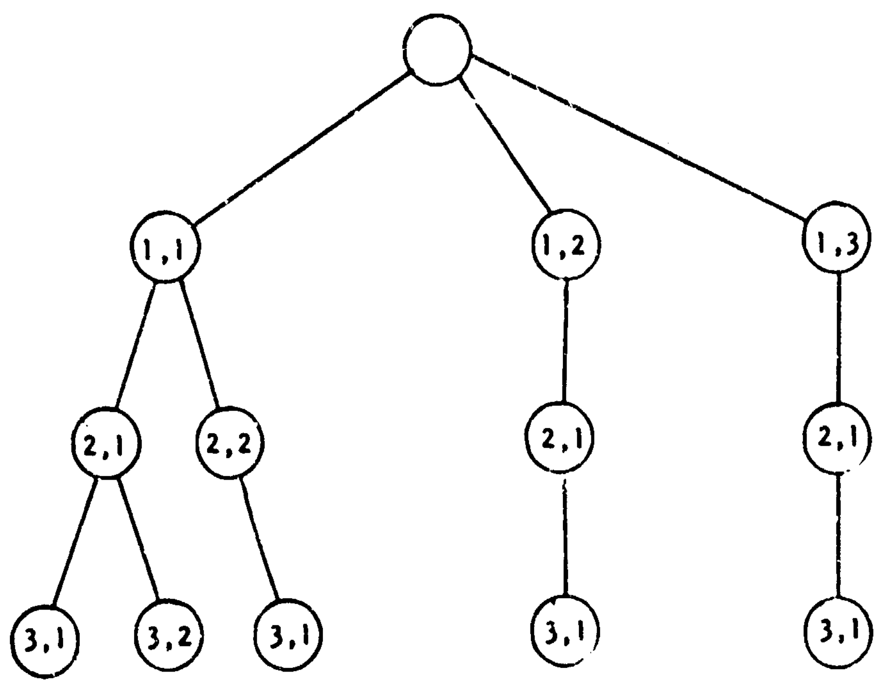


Figure VII-2

$S_{1,1}$, $S_{1,2}$ and $S_{1,3}$ while holding all other jobs as they were in the initial solution $(S_{2,1}, S_{3,1})$. Then the first decision job is returned to its original setting $S_{1,1}$ and then job S_2, then S_3 are iterated through their possible alternatives. The combinations of jobs examined are shown in Figure VII-2. Note that each solution contains only one job that is different than the set of jobs in the initial solution and in total we examine only four solutions in addition to the initial solution.

For each combination of decision jobs, the initial solution plus the set of four combinations generated by the pairwise interchange routine, the project is scheduled using the heuristic discussed in Section 2 of this chapter. For each of the four schedules, the sum of job cost and completion date cost is calculated. The combination of jobs with lowest total cost is compared to the cost of the initial solution if it is lower, this combination becomes the new "initial" solution and jobs in that solution are exchanged. The process continues until no improvement is found.

The ten problems tested by complete enumeration were tested again with the pairwise interchange routine. The cost of the best solutions generated, along with computation times, are given in Table VII-4. Complete details of the solutions are given in Appendix D. A review of results shows that for each of the ten projects, the optimal solutions were found and that, in every case, the solutions stepped from one undominated solution to another.

For comparison with a later heuristic two, fifteen decision set projects were attempted. The solutions obtained to these problems are shown in Table VII-4. It will be noted that these problems ran 268 and 465 seconds. These times suggest that a more efficient technique is required for large problems.

6. Multiple Pairs Switching

The pairwise interchange routine when applied to a five decision set problem, three jobs per set, examines ten alternative solutions, then accepts the best of these for further exploration. In some instances, several good exchanges are discovered on the first pass, but only the best exchange is accepted. Then, in the second set of pairwise exchanges, a previously discovered change is found to be good and at the end of the second pass, it is accepted. This suggests that a more efficient routine should, at each stage, accept all exchanges that appear to be beneficial.

The routine developed proceeds as follows. It begins by applying a simple pairwise switching routine to all decision variables. The base solution used is simply that which contains the cheapest job from each decision set. For each new job brought into the solution, we calculate a project length based on the loading heuristic described earlier in this chapter. The total cost of this solution is compared to the previous lowest cost solution and the difference is defined as a price for the decision job.

When all decision jobs not in the starting solution have been evaluated in this way, we have "prices" for all decision jobs. The implicit price of jobs in the original solution is, of course, zero. Now, using the algorithm of Chapter IV, with job prices as defined here, we can solve for the best set of jobs to perform. A solution might imply that one or more variables are to be switched. This design can then be evaluated by our loading heuristic and a total cost can be calculated. If this cost is lower than our previous optimum, we recalculate job "prices" with a pairwise interchange routine, otherwise, we stop. Results of this algorithm for our twelve problems are shown in Table VII-4 and in more detail in Appendix D.

7. Discussion of Results

Table VII-4 compares the computation times and quality of solutions for the three routines applied to the full problem. For small problems (five decision sets), the complete enumeration routine takes approximately 6-9 times as long as pairwise interchange methods. Pairwise interchange techniques take approximately the same time as the multiple pairs approach. In eight of the ten problems examined, both the pairwise interchange and the multiple pairs exchange method found the optimal solution as proved by complete enumeration. In one of the remaining two problems, the pairwise interchange routine found a better solution than multiple pairs.

For large problems (fifteen decision sets), the pairwise interchange routine took longer but found superior solutions to the multiple pairs method. For these problems, it was not practical to enumerate all possible designs and so the optimal solution is not known as computation times for these problems of 6 to 11 minutes suggests that even the simple routines would not be practical for large problems.

PROBLEM NO.	TOTAL ENUMERATION		PAIRWISE		MULTIPLE PAIRS	
	COMPUTER TIME (seconds)	MINIMUM COST	COMPUTER TIME	COST	COMPUTER TIME	COST
66	259	$450	36.8	550	40.9	600
67	310	650	56	650	38.6	650
68	345	500	68.5	500	52.8	500
69	345	510	59.2	510	59.7	510
70	354	450	70	450	60.4	450
71	304	590	48	590	48.6	590
72	228	2400	42	2400	43.3	2400
72	364	190	34.4	190	50.4	190
74	129	450	18	850	22.8	850
75	228	400	54.5	400	37.5	400
76	-	-	686	836	468	867
77	-	-	441	710	380	914

Table VII-4

Chapter VIII

APPLICATIONS OF THE DCPM MODEL

The decision nodes of the DCPM model have been primarily used throughout this thesis to represent job alternatives in the discrete time-cost tradeoff problem. This chapter will show that the decision nodes may also represent any given job performed at different points in time, or at different physical locations and with these interpretations, the model may be used to formulate the resource constrained project scheduling problem and the single product assembly line balancing problems as integer programming problems. In addition, the application to project time cost trade-off problems will be extended to projects under incentive contracts with non-linear criterion functions.

2. The mxn Job-Shop Scheduling Problem

This formulation assumes that the job-shop problem has been solved heuristically [20, 28] and that a feasible finish date W_f, the early start of artificial finish job S_f has been determined. For each job, S_i, it is then possible to calculate an early start ES_i and an early and a late finish time, EF_i, LF_i, using the usual rules of the critical path method.

If the criterion function is to minimize the make span for the fixed job file under consideration, then job S_i must begin on day ES_i or later and must finish on or before LS_i. A start before ES_i

is not possible given precedence constraints and similarly a finish later than LF_i would delay the completion of the project beyond W_f. Since W_f is a feasible solution, a schedule that delays W_f cannot be optimal.

Subscripts:

f machines	$f = 1,2,\ldots,m$
t day	$t = 1,2,\ldots,D$
i job	$i = 1,2,\ldots,f$

$P_i = \{$set of immediate predecessors of job i$\}$

$L_{tr} = \{$set of all jobs performed on day t on machine r$\}$

Variables:

d_{ij} job i beginning on day j $\quad j = ES_{ij},\ldots,LS_i$

$$d_{ij} = \begin{cases} 1 & \text{if the job is performed} \\ 0 & \text{otherwise} \end{cases}$$

Constraints:

A_{fd} number of machines of type f available on day d

t_i time length of job i. It is assumed that each job requires only one machine.

ES_{ij} the start time associated with each job alternative S_{ij}

Constraints:

1) Interdependence

$$\sum_{j=ES_i}^{LS_i} d_{ij} = 1 \qquad\qquad i = 1,\ldots,m$$

each job will be performed once and only once.

2) Resource Limits

$$\sum_{S_{ij} \varepsilon L_{tr}} d_{ij} \leqslant a_{ft} \qquad\qquad \begin{array}{l} f = 1,\ldots,m \\ t = 1,\ldots,W_f \end{array}$$

Machine capacity will not be exceeded on any day

3) Precedence Constraints

$$\sum_{j=ES_p}^{LS_p} (ES_{pi} \quad t_p) d_{pj} - \sum_{j=ES_i}^{LS_i} (ES_{ij}) d_{ij} \quad \text{all} \quad p \varepsilon P_i \qquad i = 1,\ldots,S_f$$

A job cannot be started until all its predecessors are completed.

Criterion:

Minimize

$$\sum_{j=ES_f}^{LS_f} ES_{fj}$$

This criterion function attempts to minimize the day on which the artificial finish job, S_f, begins. This effectively minimizes the day on which all jobs are finished on all machines.

This formulation is related to that of Bowman [8] and may be extended to the resource constrained project in the same way that Wiest [80] has extended Bowman's model. The project formulation may also be extended to the resource constrained discrete time-cost trade-off problem by expanding the decision set for each job to include several (k(i)) job alternatives.

Wiest [80] has estimated that a project with 55 jobs in 4 shops with a time span of 30 days would require 5225 equations and 1650 variables. If job splits were not allowed, the number of equations would rise to 6870. The formulation suggested here requires an inter-dependency constraint for each job (55), a resource constraint for each resource for each day (4 x 30 120) and a constraint for each precedence relation (100 - 500). We can then estimate that this formulation would require 300 - 700 constraints -- approximately one-tenth of those with Bowman-Wiest formulation. In this formulation there would be at least one variable for each day of slack in the original heuristic schedule. This number would be substantially below the figure 1650, estimated by Wiest. An estimate in the range 50-200 would not seem unreasonable here. The exact number of constraints and variables will, of course, depend on the specific problem.

It is difficult to make an exact comparison with Manne's formulation [54] since his approach is not suitable for the resource constrained project problem. He implicitly assumes a resource level of one by his use of non-interference constraints. He does estimate that a job file of ten tasks to be performed on five machines would require 250 variables. There would be a constraint for each precedence relation and two constraints for <u>each pair</u> of jobs which must use every machine. This would involve approximately 500 constraints for his problem.

3. The Single-Product Assembly-Line Balancing Problem

This formulation will again assume that the combinatorial problem has been solved heuristically [72, 77] and an upper limit M_{max} has been set on the number of stations to be used.

Subscripts:

j - stations $\quad j = 1,\ldots,M_{max}$

i - job $\quad i = 1,\ldots,S_f$

S_{ij} - job i performed at station j

$P_i = \{$set of immediate predecessions of job $i\}$

$L_j = \{$set of all job alternatives (S_{ij}) that may be performed at station $j.\}$

Variables:

d_{ij} job i performed at station $j \quad j = 1,\ldots,M_{max}$

$$d_{ij} = \begin{cases} 1 & \text{if job is performed at station } j \\ 0 & \text{otherwise} \end{cases}$$

Constraints:

C cycle time -- maximum amount of work a station may perform

t_i time length of job i

Constraints:

1) Interdependence

$$\sum_{j=1}^{M_{max}} d_{ij} = 1 \qquad i = 1,\ldots,S_f$$

each job must be performed once and only once.

2) Resource limits

$$\sum_{S_{ij} \in L_j} t_i d_{ij} \leqslant C \qquad j = 1,\ldots,M_{max}$$

only C units of time can be performed at any station

3) Precedence

$$\sum_{j=1}^{M_{max}} j d_{pj} \leqslant \sum_{j=1}^{M_{max}} j d_{ij} \qquad p \in P_i \quad i = 1,2,\ldots,S_r$$

A job cannot be allocated to a station unless all its predecessors are assigned to that station or an earlier one.

Criterion:

Minimize $$\sum_{j=1}^{M_{max}} j d_{f,j}$$

Here we attempt to minimize the number of the station in which the final job, S_f, appears. This effectively minimizes the total number of stations used.

For the 11 job problem of Jackson [42] with C = 10 and an initial heuristic solution of six stations, this formulation would require eleven interdependency constraints, six resource constraints and fourteen precedence constraints for a total of 31. The maximum number of variables would be 66, but this could be reduced by an ES-LS argument to approximately 39. This is a substantially smaller problem than competitive formulations.

4. An Application of Decision CPM to Incentive Contracts

Recently several government agencies have changed their contracting procedures from predominantly cost-plus-fixed fee to incentive fee contracts. The incentive contracts are written so that the maximum fee obtainable decreases as cost of the project increases [59] and the per cent of the maximum fee actually paid decreases with decreasing performance. Performance points may be awarded for successfull performance or quality tests, and for meeting a series of specified due dates (or mile stones) within the project network. A sample contract fee structure is shown in Figure VIII-1.

A manufacturer faced with a time-cost trade-off problem within an incentive contract has an especially difficult problem. If a job is "crashed", it is possible that extra points will be earned as a result of meeting a particular due date. At the same time, the increase in project cost will cause all "points" to be slightly devalued. The following integer programming* problem is solved iteratively to determine the optimal selection of jobs to be performed. The exact procedure will be described after the model is presented.

Subscripts:

i	job sets	$i = 1,\ldots,m$
j	decision jobs in a job set	$j = 1,2,\ldots, k(i)$

Variables:

d_{ij} $\quad j = 1,2,\ldots,k(i)$

The alternative ways of performing job S_i

$$d_{ij} = \begin{cases} 1 & \text{if job } i \text{ is performed by alternative } S_{ij} \\ 0 & \text{otherwise} \end{cases}$$

W_{ij} the early starting time of job S_{ij}

$S_m \in F_m = \{\text{the set of all jobs that are given due dates}\}$

$S_e \in P_{ij} = \{\text{the set of all jobs that precede job } S_{ij}\}$

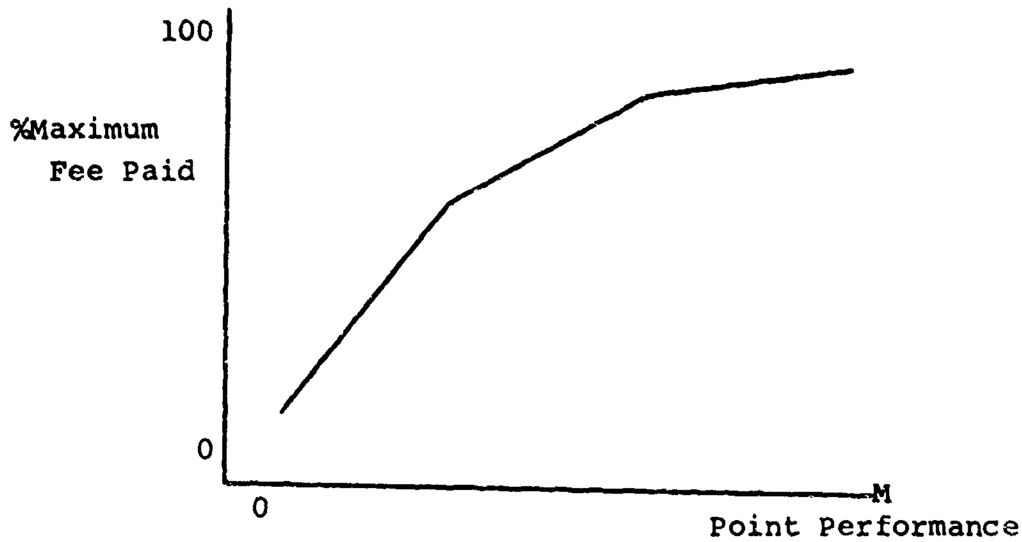
W_m^+ the number of days after D_m that S_m is completed

W_m^- the number of days before D_m that S_m is completed

---

* This problem was originally formulated by the author, then applied to an actual problem by E. Smylie [69].

RELATION BETWEEN POINT PERFORMANCE
AND % MAXIMUM FEE OBTAINED



RELATION BETWEEN MAXIMUM OBTAINABLE FEE
AND COST PERFORMANCE

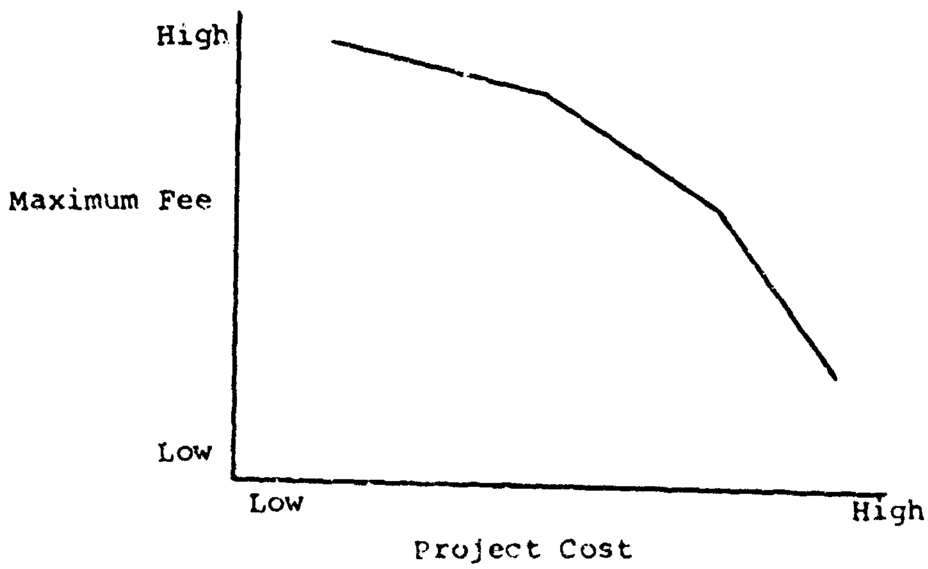


FIGURE VIII-1

RELATION BETWEEN FEE PAID AND
PERFORMANCE ON POINTS AND COST

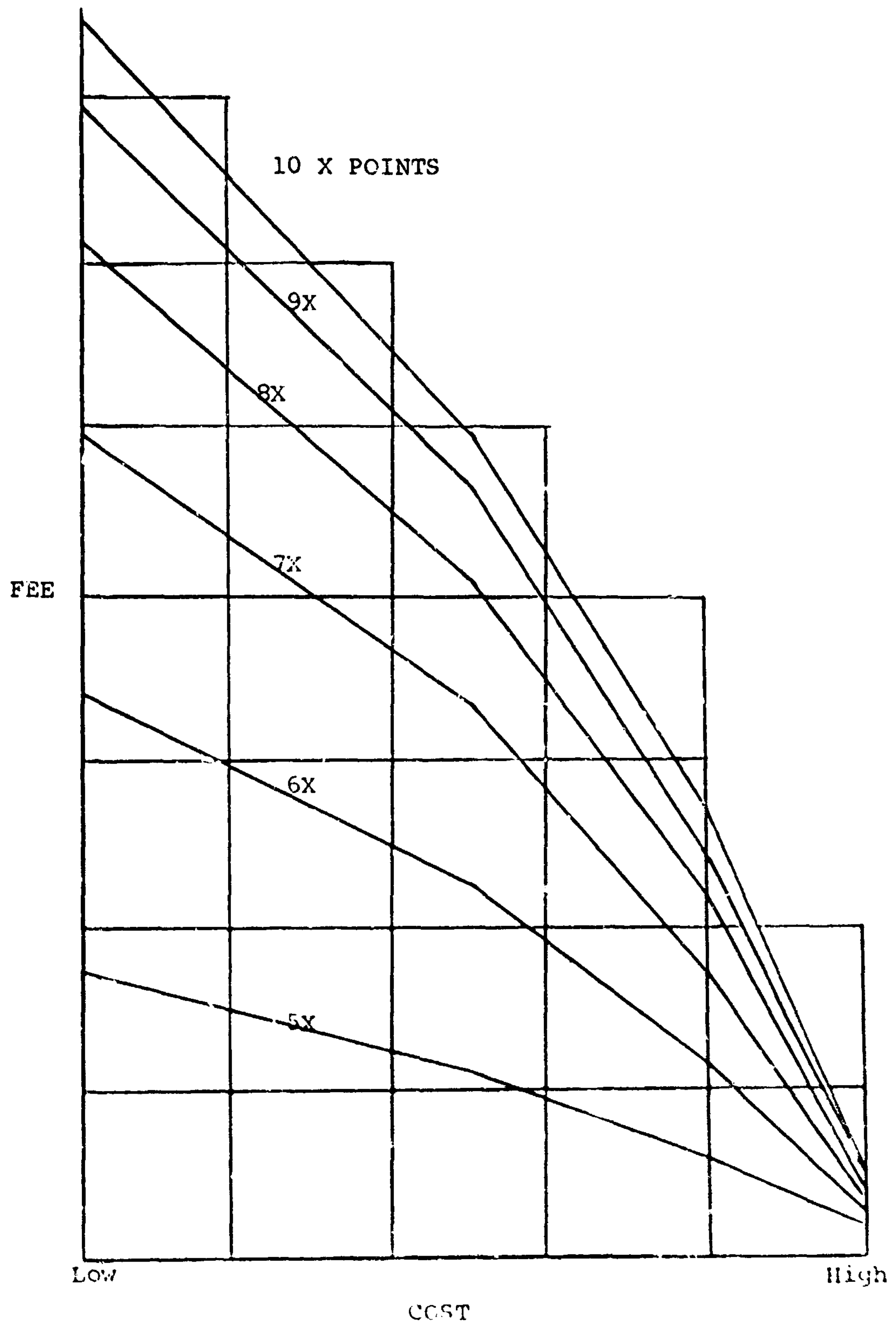


FIGURE VIII-2

Constants:

t_{ij} the time required to perform S_{ij}

C_{ij} the cost of S_{ij}

D_m due date given to job S_m

B maximum project cost

P_m point loss for each day after D_m job S_m is completed

r_m daily point reward for early completion of job S_m

Constraints:

Interdependence

$$\sum_{j=1}^{k(i)} d_{ij} = 1$$

Precedence

As in [19] if S_i precedes S_m and S_i is a unit set job, the precedence relation is shown

$$W_i + t_i \leq W_m$$

and if S_{ij} precedes S_m and it is a multi-set job

$$-M(1-d_{ij}) + t_{ij} + W_{ij} \leq W_m$$

Resource (Budget) Constraint:

$$\sum_{i=1}^{n} \sum_{j=1}^{k_i} C_{ij} d_{ij} \leq B$$

Due Date Constraints:

$$W_m - W_m^{+} + W_m^{-} = D_m \quad m \in F_m$$

Criterion:

$$\text{Maximize} \sum_{m \in S_m} r_m W_m^{-} - p_m W_m^{+}$$

This formulation maximizes the number of performance points obtained subject to cost limits on the complete project. Initially, the problem is solved without a budget constraint to determine the maximum available points and the cost associated with this solution. B is then set one unit below the cost found above and the problem is resolved.

The routine is applied iteratively until the minimum cost point is reached. At each solution the combination of performance points obtained and budget cost will allow total fee to be calculated. Figure VII-3 shows a series of such points with an optimal solution marked. This approach has been applied to an actual oroject by Smylie [69] and is reported by Crowston and Smylie [22].

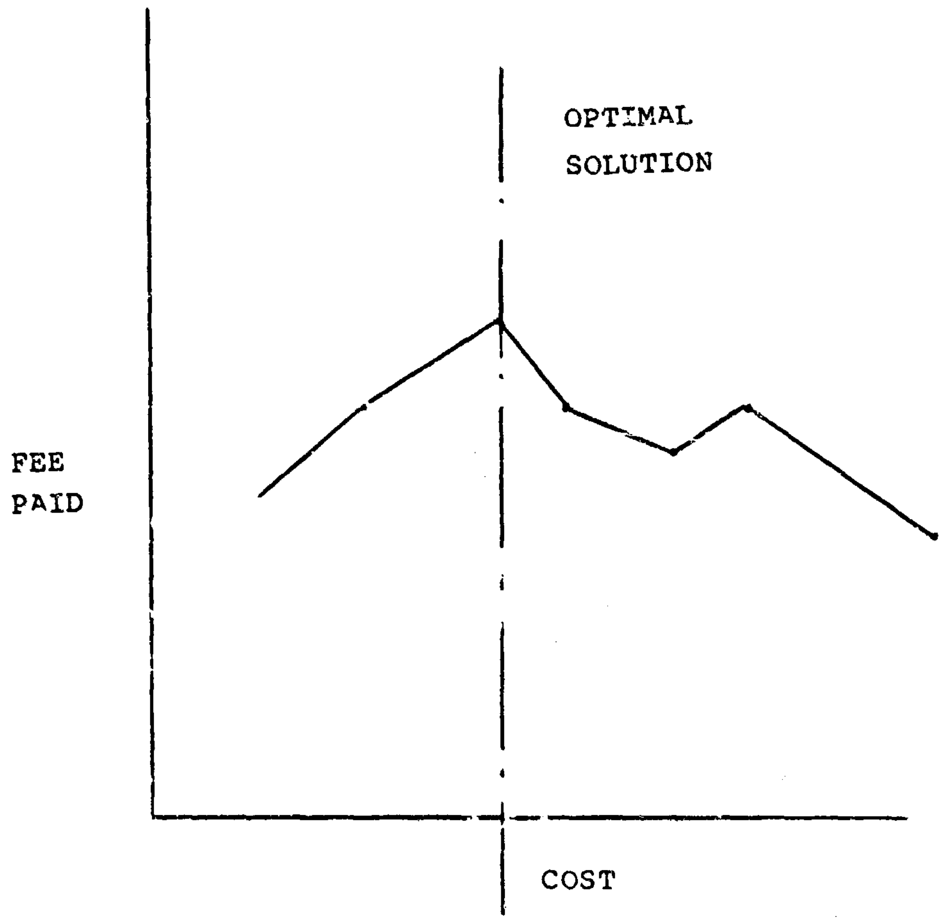


FIGURE VIII-3

Chapter IX

Summary and Conclusions

Our original goal was to determine efficient procedures for the solution of design problems in situations where the design could only be evaluated in terms of optimal operating decisions for the system. The study was confined to the area of project scheduling although the models we used were related to many other planning problems. The relationship between these problems was shown in the literature review of Chapter II.

The articles reviewed there were categorized by the type of constraint found in the model. These included time or precedence constraints, showing the sequence in which the jobs were to be performed, or putting time constraints on the start or finish time of a particular job. The use of resource constraints implied that the jobs required resource inputs to be performed and that the availability of the resource was limited in each time period. Finally, interdependency constraints between individual tasks were introduced. One particular type of interdependency, that is, the mutually exclusive relation between sets of jobs, was developed in some detail throughout the thesis since the job alternatives could be used to represent design alternatives in a planning problem.

Models with simple time constraints, the usual Critical Path problem, could be solved optimally with longest path algorithm. Problems that involve simple interdependency relations form a special class of 0 - 1 integer programming problem. Tree search schemes have been developed for this class of problem. With time and interdependency constraints in a model, as we have in DCPM, the literature suggest a standard integer programming routine is required.

When resource demands and constraints are added to a model, it becomes a difficult problem to solve it optimally. For example, given time and resource constraints on a set of tasks, it is a difficult combinational problem to find the sequence of jobs that minimize project length. Branch and bound techniques have been developed for small problems, but large problems must be solved heuristically. Only a few models include all three types of constraints that we have discussed, and these are solved heuristically.

The design problem that is the central concern of the thesis is the problem of selecting which jobs to perform, from a decision set of mutually exclusive alternatives. The optimal set minimizes the total of job cost and project completion date cost. This problem is formulated in Chapter III as an integer programming problem. It is shown in this chapter that the number of precedence constraints required for any network would be one for each precedence link, or one for each path in the network. This number could be so large that the problem could not be solved by current integer programming routines.

In Chapter IV dominance, feasibility and lower bound tests are developed to eliminate many non-binding precedence constraints from the integer programming model. This is equivalent to the elimination of many paths, which can under no condition become critical, from the decision network. The remaining set of paths which may become the critical path are termed the "reduced constraint" set. An algorithm is developed to implement the dominance tests referred to above. In a decision graph, the algorithm determines the longest path between all pairs of decision jobs, if any path exists, and eliminates all paths but the longest one. The result is a "reduced network" containing only decision jobs and maximal length paths between decision jobs.

If we consider the set of tasks in any DCPM problem and the set of interdependence constraints defined on that set, we observe large numbers of mutually exclusive relations, "job alternative" interdependencies, and some number of constraints between individual members of job sets, "other" interdependencies. In Chapter V an integer programming network algorithm is developed to efficiently solve the problem of selecting the minimum cost set of decision jobs given these types of interdependency constraints. The optimization technique used is the "longest path" calculation of the critical path method applied to a network in which each path through the network is a feasible solution to the integer programming problem. The length of each path is exactly equivalent to the value of the criterion function for the solution the path represents. In this algorithm, the usual "slack" measure of the critical path method may be interpreted as the dual evaluation of variables which are not in the solution. When the structure of interdependence in a problem does not allow the algorithm to be directly applied, it is shown that it may be coupled with a branch and bound algorithm to solve the problem.

In Chapter VI two branch and brand routines are developed to solve the DCPM problem. Both of these use the algorithm of the previous chapter to handle the interdependency constraints. In the first routine at each node, we calculate the minimum cost job selections, given job selections and rejections to that point on the tree and determine the critical path given these job selections. The branching decisions are made so as to progressively break all critical paths, by prohibiting the use of decision jobs on the path. The second algorithm sets a fixed order in which the decision sets will be considered, and at each node, evaluates all alternatives within that set. Given any selection of alternatives, a lower bound on total project cost is given by the sum of job cost, for those jobs previously committed, and a project completion cost, based on a finish date determined from the reduced network, with only the jobs that were previously committed active in the network.

The second routine was coded and tested with two different orderings of the job sets. The first fixed order was simply a technological order taken from the position of the decision sets in the original network. The second fixed order was arranged so that the decision sets were listed in order of increasing slack. This slack was determined by choosing from each decision set, the cheapest decision job and solving the resulting critical path problem. The slack on these jobs was then determined and its value used to rank the decision sets. In every case, the slack order was superior to a technological order and the superiority ranged from 5:1 to over 50:1. Using the slack order problems of fifteen decision nodes or 1.5×10^6, possible designs were solved in 1.6 to 130 seconds. This suggests that large decision networks may be solved efficiently with this routine.

In Chapter VII, decision networks with resource limits were introduced. Before the combined "design", "operating" problem was attempted, it was necessary to determine a good "operating" rule, or, in this case, a heuristic loading rule for sequencing project jobs under limited resources. Several rules from the literature were tested and of these a serial loading rule, operating on a job list ordered by late start, with no job bumping, proved superior.

This rule was then used in combination with three tree search techniques. These were complete enumeration, to provide proof of the optimal solution, pairwise interchange and multiple pairs exchange. The pairwise interchange was superior to multiple pairs routine in that the computation times were approximately the same but the solutions were superior. In ten problems the pairwise routine solved eight optimally. It was concluded that none of the techniques were efficient for large problems.

In the last chapter, the decision network integer programming formulation was applied to the m x n job shop scheduling problem, the single product assembly-line balancing problem and the DCPM incentive contrast problem. For the first applications, the formulation was markedly more efficient than the one existing integer programming formulations used as a basis of comparison. The last formulation was interesting in that the model was applied to a problem with a non-linear criterion function.

Suggestions for Future Research

The research questions that have arisen in this study fall into three broad categories. First we will suggest specific areas for development of algorithmic and heuristic reoutines for the solution of related problems. Then we will discuss the need for additional tests of the models developed here on actual problems and, finally, we will discuss the extension of these models to other planning problems.

It is clear that much more work remains to be done on the development of truncated enumeration schemes for the DCPM problem. We have suggested two here and tested one of these. It would be desirable to develop other approaches to this problem, specifically one that could operate on the original network and save the time required by our network reduction scheme. All models could then be tested against a range of DCPM problems. In Chapter V we developed an integer programming routine for restricted types of problems, and suggested that it could be coupled with a tree search scheme for more complex problems. It would be useful to develop a program for such a combined model and test it on a series of problems.

The area that requires the most work would be the solution of resource constrained decision networks. We have attempted several methods that give reasonable solutions, but use excessive amounts of time. A routine patterned after the first branch and bound scheme of Chapter VI would have some hope of success. In this application, the critical sequence as defined by Wiest would appear to be useful in the branching rule, rather than the critical path used here.

One application of the DCPM model [69] has been attempted and, in that instance, the network reduction rules suggested here were extremely powerful in reducing the size of the network that had to be considered. This experience suggests that many real networks have large areas with substantial slack, so that large reductions is possible. If this is true in general, then the network reduction notions of this thesis may be extremely useful to managers even though they do not use full range of optimization techniques suggested here. If a large problem is reduced to a small one, then the manager's heuristics may perform better.

Finally, we believe that these models will be useful at the detail level of job-planning, that is, to the optimization of man-machine and related process charts and to the most aggregate level of policy-making within a firm. Wherever there is a variety of things to be done, connected with the types of constraints we have discussed throughout this thesis, our models, or some version of them, should be relevant. We hope, then, that this research will influence the development of a wide range of planning models.

APPENDIX A

APPENDIX A

Problem Generation

The problems used throughout this thesis were generated from six basic precedence networks. We shall label these Type A, B, C, D, E and F. All problems of type A contained an identical tree structure (precedence ordering) and identical decision jobs. The job lengths, job costs and resource usages for all decision jobs in all projects were predetermined. All these nodes contained exactly three alternatives. All other job times and resource usages were randomly determined for all non-decision jobs in all problems. Finally, the cost of all non-decision jobs was set at zero since these costs were constant.

The subroutine, HYPO, used to generate the final project from the basic networks is shown at the end of this Appendix. The following table gives some data on the precedence networks. This is number of unit jobs, number of decision sets, number of precedence links.

Network	Jobs	Decision Sets	Precedence Links
A	34	5	72
B	46	5	87
C	49	5	92
D	49	5	92
E	171	15	364
F	202	15	343

Table A-2 will present details on projects used throughout the thesis. The information includes network type, critical path (given cheapest decision jobs), resource limit, total resource usage (cheapest decision job solution) and project deadline information where relevant.

PROJECTS

Problem No.	Type	Critical Path	Resource Limit	Total Resource Usage Res. 1	Res. 2	Res. 3
1	A	61	12	899	514	579
2	A	50	12	715	403	463
3	A	57	18	747	332	266
4	A	66	18	919	85	81
5	A	70	18	815	217	68
6	A	63	12	891	580	654
7	A	53	15	526	286	265
8	A	64	12	1,031	225	263
9	A	62	15	875	93	56
10	A	62	12	765	523	580
11	A	58	12	769	67	18
12	A	54	12	713	549	410
13	A	67	12	736	206	102
14	A	62	12	519	199	142
15	A	62	12	750	58	48
16	A	60	12	942	674	678
17	A	58	12	723	208	2[illegible]6
18	A	58	12	709	83	36
19	A	63	12	928	749	638
20	A	64	12	538	162	403
21	A	56	12	785	0	94
22	A	55	15	766	468	717
23	B	75	12	1,254	713	952
24	B	74	15	1,027	775	811
25	B	63	18	999	770	700
26	B	74	12	861	296	0
27	B	70	15	732	346	199
28	B	88	18	934	307	364
29	B	67	12	851	106	143
30	B	79	15	973	164	117

Problem No.	Type	Critical Path	Resource Limit	Total Resource Usage Res. 1	Res. 2	Res. 3
31	B	63	18	1,279	78	114
32	C	67	12	1,355	988	974
33	C	61	12	696	302	351
34	C	63	12	1,051	130	121
35	C	67	15	914	601	738
36	C	69	15	1,044	0	232
37	C	61	15	999	136	121
38	C	58	18	1,075	690	782
39	C	63	18	600	427	412
40	C	63	18	985	185	122
41	D	76	12	1,392	785	1,034
42	D	63	12	1,082	432	510
43	E	218	12	4,723	4,230	4,063
44	E	194	10	4,700	578	579
45	E	195	18	4,561	298	357
46	E	192	12	4,164	3,954	3,265
47	E	195	18	2,456	1,666	1.229
48	E	202	12	2,333	1,695	1,726
49	E	207	12	4,268	341	342
50	E	201	15	4,704	4,048	4,054
51	E	221	15	2,185	1,397	1,996
52	E	207	15	4,350	560	467
53	E	200	18	5,101	4,445	4,043
54	E	218	12	5,849	3,022	2,465
55	E	218	14	3,936	4,142	3,846
56	E	210	12	3,645	3,258	3,158
57	F	102	12	4,397	3,610	3,536
58	F	95	18	5,227	661	684
59	F	109	12	2,322	1,627	1,808
60	F	118	12	4,467	456	491

Problem No.	Type	Critical Path	Resource Limit	Total Resource Usage Res. 1	Res. 2	Res. 3	Reward	Due-Date	Penalty
61	F	100	15	4,383	3,616	3,826			
62	F	95	15	2,320	1,429	1,594			
63	F	100	15	4,401	485	421			
64	F	113	18	4,638	4,086	3,953			
65	F	103	12	2,226	1,223	1,565			
66	D	69	18	1,155	466	732	40	70	50
67	B	87	10	1,079	522	543	50	135	50
68	D	67	10	1,212	298	269	30	145	100
69	D	76	12	1,392	785	1,034	30	142	60
70	C	76	10	1,111	494	621	50	134	35
71	B	78	15	1,191	1,054	895	50	105	80
72	B	63	18	999	770	700	10	50	100
73	C	67	12	1,354	988	974	10	160	20
74	A	53	10	523	118	0	10	66	200
75	B	63	18	999	770	700	30	70	60
76	F	102	12	4,397	3,610	3,536	30	470	70
77	E	202	12	2,333	1,695	1,726	30	280	50
78	E	-	-	-	-	-	150	136	50
79	E	-	-	-	-	-	150	186	50
80	E	-	-	-	-	-	150	187	50
81	E	-	-	-	-	-	150	195	50
82	E	-	-	-	-	-	150	185	50
83	F	-	-	-	-	-	150	150	50
84	F	-	-	-	-	-	150	90	50
85	F	-	-	-	-	-	150	97	50
86	F	-	-	-	-	-	150	92	50
87	F	-	-	-	-	-	150	100	50

```
      SUBROUTINE HYPO (CP,IP,K,TIM)
      DIMENSION CP(260,25),IP(50,15),K(30),TIM(15)
      PRINT 4
    4 FORMAT (35H PUNCH ISEED I5,VRES,VM1,TIM5-7F5.3)
      READ 1,ISEED,VRES,VM1,(TIM(I),I=11,15)
    1 FORMAT(I5,7F5.3)
      PRINT 37
   37 FORMAT(13H PUNCH K14-I5)
      READ 18,(K(I),I=1,14)
   18 FORMAT(14I5)
      PRINT 3
    3 FORMAT(15H PUNCH IFILE I5)
      READ 5,IFILE
    5 FORMAT (I5)
      DO 7 I1=1,300
      READ(IFILE,6)(TIM(I2),I2=1,10)
    6 FORMAT (10F5.0)
      IF(TIM(1).EQ.0.) GO TO 9
       CP(I1,12)=1
      IF (TIM(2).EQ.0.) GO TO 16
      CP(I1,12)=3
   16 K(1)=K(1)+1
      DO 8 I3=1,10
      CP(I1,I3)=TIM(I3)
    8 CONTINUE
    7 CONTINUE
    9 CONTINUE
      REWIND IFILE
      CALL SETUF(ISEED,USEED)
C     PICK UP DJ INFO
      I2=1
      K1=K(1)
      IP(I2,1)=1
      DO 10 I1=1,K1
      CP(I1,18)=1
      CP(I1,22)=CP(I1,1)
      IF(CP(I1,12).EQ.1.) GO TO 10
      I2=I2+1
      IP(I2,2)=CP(I1,1)
      IP(I2,15)=CP(I1,2)
      IP(I2,3)=CP(I1,2)
      IP(I2,1)=I2
      IP(I2,14)=CP(I1,12)
      CP(I1,23)=I2
   10 CONTINUE
      K(5)=I2-1
C     NOW PRED RELATION-ASSUME ALL 3
      IP(2,4)=1
      IP(3,4)=1
      IP(4,4)=1
      IS=I2-5
      DO 13 I3=2,IS,3
      I6=I3+3
      I8=I3+5
      DO 12 I4=I6,I8
      DO 11 I5=1,3
      I7=I5+3
      IP(I4,I7)=I3+I5-1
   11 CONTINUE
```

```
   12 CONTINUE
   13 CONTINUE
      IF=I2+1
      DO 14 I1=1,3
      I2=I1+3
      IP(IF,I2)=IF-4+I1
   14 CONTINUE
      IP(IF,1)=IF
       TIM(1)=.07
       TIM(2)=.21
       TIM(3)=.34
       TIM(4)=.67
       TIM(5)=.79
       TIM(6)=.86
       TIM(7)=.92
       TIM(8)=.97
       TIM(9)=.99
       TIM(10)=1.0
      K1L1=K1-1
      DO 330 I=2,K1L1
      IF(CP(I,12).NE.1.) GO TO 330
       CALL RANNOF(USEED,VRAN)
       DO 320 ITM=1,15
       T=TIM(ITM)
       IF(VRAN-T) 321,321,320
  320 CONTINUE
  321 CONTINUE
       CP(I,3)=ITM
  330 CONTINUE
       TIM(1)=.04
       TIM(2)=.12
       TIM(3)=.22
       TIM(4)=.34
       TIM(5)=.48
       TIM(6)=.64
       TIM(7)=.78
       TIM(8)=.88
       TIM(9)=.96
       TIM(10)=1.0
       TIM(15)=1.0
      DO 380 I=2,K1L1
       IF(CP(I,12).NE.1.) GO TO 380
       CALL RANNOF(USEED,VRAN)
       IF(VRAN-VRES) 370,450,450
  370 CONTINUE
       CALL RANNOF(USEED,VRAN)
       DO 390 IG2=11,15
       T=TIM(IG2)
       IF(VRAN-T) 400,400,390
  390 CONTINUE
  400 CONTINUE
       IG2=IG2-4
  410 CONTINUE
       CALL RANNOF(USEED,VRAN)
       DO 420 ITM=1,10
       T=TIM(ITM)
       IF(VRAN-T) 430,430,420
  420 CONTINUE
  430 CONTINUE
```

```
              CP(I,IG2)=ITM
              CALL RANNOF(USEED,VRAN)
              IF(VRAN-VM1) 370,380,380
         450 CONTINUE
         380 CONTINUE
              PRINT 17,K1
          17 FORMAT (I5)
      C         DO 15 I=1,K1
      C         PRINT 600,(CP(I,J),J=1,15)
         600 FORMAT(15F5.0)
          15 CONTINUE
              RETURN
              END
R 5.283+1.566
```

APPENDIX B

COMPUTATIONAL RESULTS - REDUCED NETWORK ALGORITHM

The iterative application of the longest path algorithm as described in Chapter IV-3 gives maximal distances between all decision jobs. The time required for this routine on six baxic networks is shown below.

NETWORK	COMPUTATION TIME Sec.
A	7.7
B	12.5
C	15.3
D	14.7
E	38.5
F	35.2

A sample output from the program for network A follows:

Decision Jobs From	To	Maximum Path Length (days)
$1(S_S)$	3	1
1	4	1
1	5	1
1	9	11
1	10	11
1	11	11
1	12	8
1	13	8
1	14	8
1	26	14
1	27	14
1	30	14
1	S_F	33

Decision Jobs From	To	Maximum Path Length (days)
3	26	11
3	27	11
3	30	11
3	S_F	30
4	26	16
4	27	16
4	30	16
4	S_F	35
5	26	26
5	27	26
5	30	26
5	38	45
9	S_F	26
10	S_F	29
11	S_F	31
12	22	16
12	23	16
12	24	16
12	S_F	35
13	22	17
13	23	17
13	24	17
13	S_F	36

Decision Jobs From	Decision Jobs To	Maximum Path Length (days)
14	22	18
14	23	18
14	24	18
14	S_F	37
22	S_F	11
23	S_F	13
24	S_F	21
26	S_F	15
27	S_F	17
30	S_F	20

APPENDIX C

APPENDIX C

PROJECT LOADING HEURISTIC RESULTS

Problem No.	Order	Load-C	Load-N	Load	Problem No.	Order	Load-C	Load-N	Load
1	Random	104	104	124	12	Random	59	62	61
	L.S.	97	113	113		L.S.	59	61	61
	E.S.	109	112	110		E.S.	62	61	62
2		87	95	105	13		67	67	67
		79	82	97			67	67	67
		78	92	92			68	67	67
3		66	64	64	14		77	74	69
		60	76	79			63	63	71
		60	66	62			68	63	80
4		94	83	83	15		78	90	94
		78	83	87			62	79	67
		93	95	99			74	74	78
5		84	70	70	16		94	89	89
		70	81	74			76	79	86
		73	71	70			73	93	84
6		111	111	115	17		66	62	62
		102	112	128			60	60	62
		107	119	130			63	66	66
7		63	61	61	18		62	58	58
		57	58	61			58	63	60
		63	58	61			59	58	58
8		92	86	93	19		117	138	144
		84	84	116			116	120	129
		86	102	146			118	120	150
9		111	104	110	20		75	78	80
		96	99	101			73	73	92
		92	99	83			74	83	88
10		87	86	93	21		96	99	99
		71	84	97			87	90	133
		78	80	105			92	89	89
11		76	88	83	22		86	86	91
		72	73	73			76	82	112
		69	73	103			78	95	107

Problem No.	Order	Load-C	Load-N	Load	Problem No.	Order	Load-C	Load-N	Load
23	Random	181	178	184	35	Random	95	90	92
	L.S.	148	167	167		L.S.	84	89	89
	E.S.	157	162	172		E.S.	89	98	98
24		128	113	131	36		89	91	91
		98	112	140			87	90	89
		115	109	145			96	98	122
25		101	96	119	37		82	88	87
		77	83	97			81	84	85
		88	95	110			88	94	127
26		138	122	122	38		79	90	94
		102	98	123			75	83	84
		116	123	177			83	96	94
27		87	87	97	39		64	63	63
		72	74	74			63	63	63
		87	84	85			64	63	63
28		99	92	92	40		68	68	68
		92	92	92			68	71	73
		99	99	99			77	81	77
29		113	122	141	41		167	160	174
		96	103	122			153	160	189
		97	97	97			170	171	232
30		107	108	108	42		131	131	133
		80	84	102			111	119	156
		100	106	112			132	143	138
31		123	112	114	43		603	600	632
		83	97	99			568	595	911
		100	104	109			614	613	817
32		170	206	209	44		627	623	729
		155	162	159			601	619	673
		167	169	170			611	640	764
33		89	94	94	45		337	370	351
		72	81	83			298	329	431
		81	96	100			327	344	355
34		120	117	137	46		545	513	827
		109	115	175			500	564	723
		112	107	130			534	549	639

Problem No.	Order	Load-C	Load-N	Load	Problem No.	Order	Load-C	Load-N	Load
47	Random	233	223	255	59	Random	261	267	321
	L.S.	211	203	214		L.S.	237	265	324
	E.S.	230	241	242		E.S.	250	298	321
48		307	308	341	60		466	445	651
		294	309	382			432	428	489
		313	330	368			430	456	559
49		451	476	552	61		379	410	493
		417	454	607			363	358	454
		433	490	644			369	379	486
50		435	447	522	62		197	245	230
		416	448	520			184	199	235
		440	497	540			190	193	271
51		262	256	258	63		333	371	378
		245	247	253			321	359	441
		271	276	288			325	348	388
52		353	385	403	64		323	306	410
		338	347	562			318	343	383
		353	379	493			333	357	382
53		398	399	482	65		153	183	195
		349	409	490			147	164	176
		368	415	478			149	188	188
54		543	563	625					
		531	583	762					
		556	592	713					
55		453	488	663					
		410	471	573					
		464	509	574					
56		488	470	565					
		447	473	588					
		491	578	584					
57		490	551	607					
		482	482	638					
		502	529	618					
58		277	325	334					
		261	294	310					
		263	278	354					

APPENDIX D

APPENDIX D

COMPUTATIONAL RESULTS - RESOURCE CONSTRAINED DECISION NETWORK PROBLEMS

This Appendix reports experimental results for three routines, total enumeration, pairwise exchange and multiple pairs exchange applied to resource constrained decision network problems.

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
69	79	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
62	76	300	$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
62	73	350	$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,2}$	$S_{5,3}$
50	71	500	$S_{1,1}$	$S_{2,2}$	$S_{3,3}$	$S_{4,3}$	$S_{5,2}$
50	70	550	$S_{1,1}$	$S_{2,2}$	$S_{3,3}$	$S_{4,2}$	$S_{5,2}$
49	69	850	$S_{1,1}$	$S_{2,1}$	$S_{3,1}$	$S_{4,3}$	$S_{5,1}$

Critical Path		Project Length	
Length	Number	Length	Number
49	24	69	4
50	24	70	9
52	60	71	21
58	18	72	44
60	18	73	25
62	18	74	30
69	54	75	26
72	27	76	13
		77	24
		78	39
		79	8

Time 259 sec.

Total Enumeration Problem 66

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
87	153	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
87	149	300	$S_{1,3}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
87	145	350	$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
87	143	450	$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
87	137	550	$S_{1,1}$	$S_{2,3}$	$S_{3,1}$	$S_{4,3}$	$S_{5,3}$

Critical Path Length	Critical Path Number	Project Length Length	Project Length Number	Project Length Length	Project Length Number
68	162	137	1	153	15
87	81	138	1	154	18
		139	4	155	13
		140	2	156	14
		141	4	157	6
		142	2	158	11
		143	5	159	5
		144	4	160	10
		145	10	161	6
		146	10	162	8
		147	15	163	3
		148	12	164	3
		149	13	165	1
		150	12	166	2
		151	13	167	1
		152	15	168	2
				169	1
				170	1

Time [illegible]10 sec.

<u>Total Enumeration Problem 67</u>

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
67	169	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
57	169	300	$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
57	152	350	$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,2}$	$S_{5,3}$
55	150	400	$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$
57	147	450	$S_{1,1}$	$S_{2,3}$	$S_{3,3}$	$S_{4,2}$	$S_{5,3}$
55	145	500	$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$
53	143	600	$S_{1,1}$	$S_{2,3}$	$S_{3,1}$	$S_{4,2}$	$S_{5,3}$
53	142	750	$S_{1,1}$	$S_{2,1}$	$S_{3,1}$	$S_{4,2}$	$S_{5,3}$

Critical Path Length	Number
47	72
40	36
53	18
55	18
57	18
67	54
68	27

Project Length	Number	Length	Number
142	2		
143	2	156	6
144	5	157	14
145	5	158	5
146	6	159	15
147	8	160	8
148	8	161	9
149	11	162	9
150	13	163	6
151	10	164	9
152	15	165	8
153	9	166	5
154	13	167	9
155	11	168	3
		169	7
		170	3
		171	2
		172	3
		174	3
		176	1

Time 345 sec.

Total Enumeration Problem 68

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
76	153	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
76	149	300	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,2}$
76	147	350	$S_{1,3}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,2}$
64	145	400	$S_{1,1}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
62	143	450	$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
60	141	550	$S_{1,1}$	$S_{2,3}$	$S_{3,1}$	$S_{4,3}$	$S_{5,3}$

Critical Path Length	Critical Path Number	Project Length Length	Project Length Number
56	72	141	18
58	36	142	9
60	18	143	18
62	18	144	12
64	18	145	20
76	54	146	45
78	27	147	14
		148	36
		149	18
		150	33
		151	16
		153	4

Time 345 sec.

<u>Total Enumeration Problem 69</u>

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
76	141	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
76	139	300	$S_{1,3}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
76	137	350	$S_{1,3}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$
56	136	400	$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$
56	135	450	$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
56	133	500	$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$
56	131	600	$S_{1,1}$	$S_{2,3}$	$S_{3,1}$	$S_{4,2}$	$S_{5,3}$

Critical Path Length	Critical Path Number
53	48
54	24
55	36
56	30
61	18
76	81

Project Length Length	Project Length Number
131	1
133	3
134	3
135	5
136	8
137	10
138	11
139	17
140	13
14[illegible]	18
14[illegible]	19
143	15
144	19
145	18
146	14
157	15
148	13
149	8
150	12
151	5
152	5
153	5
154	4
156	2

Time 354 sec.

Total Enumeration Problem [illegible]0

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
78	115	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
78	109	300	$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
78	108	350	$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
78	107	450	$S_{1,2}$	$S_{2,3}$	$S_{3,1}$	$S_{4,3}$	$S_{5,3}$
78	106	500	$S_{1,2}$	$S_{2,3}$	$S_{3,1}$	$S_{4,2}$	$S_{5,3}$

Critical Path Length	Critical Path Number	Project Length Length	Project Length Number
59	108	106	3
70	36	107	3
74	18	108	21
78	81	109	12
		110	11
		111	9
		112	25
		113	20
		114	48
		115	18
		116	22
		117	11
		118	14
		119	6
		120	7
		121	6
		122	4
		123	2
		124	1

Time 304 sec.

Total Enumeration Problem 71

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
63	77	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
63	72	300	$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
63	71	350	$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
63	70	400	$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$

Critical Path		Project Length	
Length	Number	Length	Number
47	108	70	8
52	36	71	17
53	18	72	34
63	81	73	42
		74	37
		75	27
		76	25
		77	24
		78	18
		79	9
		80	2

Time 228 sec.

<u>Total Enumeration Problem 72</u>

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
67	155	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
67	149	300	$S_{1,3}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
67	147	400	$S_{1,3}$	$S_{2,3}$	$S_{3,1}$	$S_{4,3}$	$S_{5,3}$

Central Path		Project Length	
Length	Number	Length	Number
46	4	147	2
47	4	143	2
48	16	149	13
51	24	150	13
53	24	151	29
54	36	152	34
56	18	153	34
58	18	154	38
64	18	155	25
67	54	156	18
68	27	157	17
		158	7
		159	7
		160	3
		161	1

Time 364 sec.

Total Enumeration Problem 73

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
53	69	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
43	68	350	$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
41	63	450	$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
40	62	550	$S_{1,1}$	$S_{2,3}$	$S_{3,1}$	$S_{4,3}$	$S_{5,3}$

Critical Path Length	Critical Path Number	Project Length Length	Project Length Number
38	54	62	3
40	9	63	3
41	9	65	3
42	9	66	9
43	81	67	12
53	81	68	6
		69	12
		70	15
		71	21
		72	15
		73	21
		74	18
		75	21
		76	15
		77	18
		78	18
		79	15
		80	6
		81	3
		82	6
		83	3

Time 129 sec.

Total Enumeration Problem 74

Undominated Solutions

Critical Path	Project Length	Cost	Decision Jobs				
67	77	250	$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
63	72	300	$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$
63	71	350	$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$
63	70	400	$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$

Critical Path Length	Critical Path Number	Project Length Length	Project Length Number
47	108	70	8
52	36	71	17
53	18	72	34
63	81	73	42
		74	37
		75	27
		76	25
		77	24
		78	18
		79	9
		80	2

Time 228 sec.

Total Enumeration Problem 75

Pairwise Interchange Results

Problem 66

Due Date 70 | Premium 40 | Penalty 50

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	79	250	700
$S_{1,1}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	73	400	550

OPTIMUM SOLUTION - NOT FOUND, TIME 36,8 sec.

$S_{1,1}$	$S_{2,1}$	$S_{3,1}$	$S_{4,3}$	$S_{5,1}$	63	850	450

Problem 67

Due Date 135 | Premium 50 | Penalty 50

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	152	250	1,150
$S_{1,1}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	145	400	900
$S_{1,1}$	$S_{2,3}$	$S_{4,1}$	$S_{4,3}$	$S_{5,3}$	147	550	650

OPTIMUM SOLUTION - FOUND TIME 56 sec.

Problem 68

Due Date 145 | Premium 30 | Penalty 100

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	169	250	2,650
$S_{1,1}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	154	400	1,300
$S_{1,1}$	$S_{2,3}$	$S_{3,3}$	$S_{4,2}$	$S_{5,3}$	147	450	650
$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	145	500	500

OPTIMUM SOLUTION - FOUND TIME 68.5 sec.

Pairwise Interchange Results

Problem 69

Due Date 142 | Premium 30 | Penalty 60

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	153	250	910
$S_{1,1}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	145	400	580
$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$	143	450	510

OPTIMUM SOLUTION - FOUND TIME 59.2 sec.

Problem 70

Due Date 134 | Premium 50 | Penalty 35

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	141	250	495
$S_{1,3}$	$S_{2,3}$	$S_{4,2}$	$S_{4,3}$	$S_{5,3}$	139	[illegible]	475
$S_{1,3}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	137	350	455
$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	133	500	450

OPTIMUM SOLUTION - FOUND TIME 70 sec.

Problem 71

Due Date 105 | Premium 50 | Penalty 80

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	115	250	1,050
$S_{1,2}$	$S_{2,3}$	[illegible]	$S_{4,3}$	$S_{5,3}$	109	300	620
$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$	10[illegible]	350	590

OPTIMUM SOLUTION - FOUND TIME 48 sec.

Problem 72

Due Date [illegible]0 | Premium 10 | Penalty 100

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	77	250	2,950
$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	72	300	2,500
$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$	71	350	2,450
$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	70	400	2,400

OPTIMUM SOLUTION - FOUND TIME 42 sec.

Pairwise Interchange Results

Problem 73

Due Date 160					Premium 10		Penalty 20
Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	155	250	200
$S_{1,3}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$	149	300	190

OPTIMUM SOLUTION - FOUND TIME 34.4 sec.

Problem 74

Due Date 66					Premium 10		Penalty 200
Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	69	250	850

OPTIMUM SOLUTION - NOT FOUND TIME 18 sec.

$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$	63	450	420

Problem 75

Due Date 70					Premium 30		Penalty 60
Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	77	250	670
$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	72	300	420
$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$	71	350	410
$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	70	400	400

OPTIMUM SOLUTION - FOUND TIME 54.5 sec.

Pairwise Interchange Results

Problem 76

Due Date 70

Premium 30 Penalty 70

Schedule Length	Job Cost	Total Cost
482	836	1,676
473	932	1,142
465	936	836

OPTIMUM SOLUTION - UNKNOWN, TIME ?*? sec.

Problem 77

Due Date 280

Premium 30 Penalty 50

Schedule Length	Job Cost	Total Cost
294	836	1,536
284	878	1,078
268	1,070	710

OPTIMUM SOLUTION - UNKNOWN, TIME 441.0 sec.

Multiplc Pairs Interchange Results

Problem 66

Due Date 70 | Premium 40 | Penalty 50

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,2}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	79	250	700
$S_{1,1}$	$S_{2,2}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	71	550	600

OPTIMUM SOLUTION - NOT FOUND TIME 40.9 sec.

$S_{1,1}$	$S_{2,1}$	$S_{3,1}$	$S_{4,3}$	$S_{5,1}$	63	850	450

Problem 67

Due Date 135 | Premium 50 | Penalty 50

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	152	250	1,150
$S_{1,1}$	$S_{2,3}$	$S_{3,1}$	$S_{4,3}$	$S_{5,3}$	137	550	650

OPTIMUM SOLUTION - FOUND TIME 38.6 sec.

Problem 68

Due Date 145 | Premium 30 | Penalty 100

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	169	250	2,650
$S_{1,1}$	$S_{2,3}$	$S_{3,1}$	$S_{4,2}$	$S_{5,3}$	143	600	540
$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	145	500	500

OPTIMUM SOLUTION - FOUND TIME 52.8 sec.

Multiple Pairs Interchange Results

Problem 69

Due Date 142 Premium 30 Penalty 60

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	153	250	910
$S_{1,1}$	$S_{2,3}$	$S_{3,1}$	$S_{4,2}$	$S_{5,2}$	141	650	620
$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$	143	450	510

OPTIMUM SOLUTION - FOUND TIME 59.7 sec.

Problem 70

Due Date 134 Premium 50 Pentlay 35

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	141	250	495
$S_{1,3}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	137	350	455
$S_{1,1}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	133	500	450

OPTIMUM SOLUTION - FOUND TIME 60.4 sec.

Problem 71

Due Date 105 Premium 50 Penalty 80

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	115	250	1,050
$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$	108	350	590

OPTIMUM SOLUTION - FOUND TIME 48.6 sec.

Multiple Pairs Interchange Results

Problem 72

Due Date 50 Premium 10 Penalty 100

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	77	250	2,950
$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	70	400	2,400

OPTIMUM SOLUTION - FOUND TIME 43.3 sec.

Problem 73

Due Date 160 Premium 10 Penalty 20

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	155	250	200
$S_{1,3}$	$S_{2,3}$	$S_{3,2}$	$S_{4,3}$	$S_{5,3}$	149	300	190

OPTIMUM SOLUTION - FOUND TIME 50.4 sec.

Problem 74

Due Date 66 Premium 10 Penalty 200

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	69	250	850

OPTIMUM SOLUTION - NOT FOUND TIME 22.8 sec.

Multiple Pairs Interchange Results

Problem 75

Due Date 70 Premium 30 Penalty 60

Decision Jobs					Schedule Length	Job Cost	Total Cost
$S_{1,3}$	$S_{2,3}$	$S_{3,3}$	$S_{4,3}$	$S_{5,3}$	77	250	670
$S_{1,2}$	$S_{2,3}$	$S_{3,2}$	$S_{4,2}$	$S_{5,3}$	70	400	400

OPTIMUM SOLUTION - FOUND TIME 37.5 sec.

Problem 76

Due Date 470 Premium 30 Penalty 70

Schedule Length	Job Cost	Total Cost
482	836	1,676
448	1,689	1,029
441	1,739	867

OPTIMUM SOLUTION TIME 468 sec.

Problem 77

Due Date 280 Premium 30 Penalty 50

Schedule Length	Job Cost	Total Cost
294	836	1,536
270	1,681	1,381
276	1,208	1,088
267	1,304	914

TIME 380 sec.

APPENDIX E

APPENDIX E

Branch and Bound Program Description

A. Main Program, Subroutines and Tapes:

One, Bonnie, Clyde, Five

The program will call tape .99. to find the number of the problem tape. Format (I4)

B. Problem Tape:

1. First Line LIM, TIME, BON, TLEX, LAST, IJND, JY

Format (I5, 3F5.0, 3I5) where

LIM = Number of Decision Nodes

TIME = Due Date

BON = Penalty

TLEX = Premium

LAST = Node Number of Artificial Finish Job.

IJND = Number of "other" Interdependency Constraints

JY = Number of Decision Nodes with More than Three Alternatives.

2. Next "LIM + JY" Lines

Job Number, Job Number, Job Number, Cost, Cost, Cost

Format (3I5, 3F5.0, I5).

If there are not <u>exactly</u> three alternatives then in the last position of the line put the difference between the number and three.

3. The next set of lines show precedence relations in the reduced network.

   Predecessors Number, Successor Number, Time Format

   (2I4, F4.0)

4. A line with zero in column 1

5. Next IJND Lines.

"Other" Interdependency Constraints

Key, Job Number, Job Number

Format 3I5

Key: 1 for $\neq$

2 for $\geq$

3 for =

ONE MADTRN

```
ONE MADTRN       05/22  1429.2

      DIMENSION MPATH(5000),KLIST(200,2),COST(200),KBD(600,2),TB1(10)
      DIMENSION FBD(600),BD(3,650),TB2(10),TB3(10),LPATH(10),IJ(50,2)
      DIMENSION IO(62)
      COMMON MPATH,BD,TB1,TB2,TB3,TCOST,K,LIM,COST,TIME,BON,TLEX.LM,M,JY
      CALL RSCLCK
      READ (99,100) KFILE
  100 FORMAT (I4)
  130 FORMAT (3I5,3F5.0,I5)
  140 FORMAT (2I4,F4.0)
      MODNUM=1
      TCOST=0
      READ (KFILE,290) LIM,TIME,BON,TLEX,LAST,IJND,JY
      JY=JY+LIM
      LEM=3*JY
      KEOF=LEM+2
      DO 101 I-1,KEOF
  101 KLIST(I,2)=0
      KLIST(LEM+1,1)=1
      KLIST(LEM+2,1)=LAST
      LEM=LEM-2
  290 FORMAT (I5,3F5.0,3I5)
      N=1
      DO 165 I=1,LEM,3
      L=I+2
      READ (KFILE,130)(KLIST(J,1),J=I,L),(COST(J),J=I,L),IO(N)
      N=N+1
      IF(IO(N-1)-99) 238,239,239
  238 DO 152 M=1,2
      MN=I+M
      IF(COST(MN)) 149,150,150
  150 IF(COST(MN)-COST(I))151,152,152
  151 KTEMP1=COST(MN)
      KTEMP2=KLIST(MN,1)
      COST(MN)=COST(I)
      KLIST(MN,1)=KLIST(I,1)
      COST(I)=KTEMP1
      KLIST(I,1)=KTEMP2
  152 CONTINUE
  149 TCOST=TCOST+COST(I)
      SOC=COST(I)
  239 DO 160 J=I,L
      COST(J)=COST(J)-SOC
      KY=KLIST(J,1)
  160 MPATH(KY)=3*MODNUM+J-I-2
  165 MODNUM=MODNUM+1
      PRINT 2 ,KFILE
    2 FORMAT (I5)
      MPATH(1)=3*JY+1
      MPATH(LAST)=3*JY+2
      I=1
  170 READ (KFILE,160) L1,L2,FBD(I)
      IF(L1) 180,115,180
  180 KBD(I,1)=MPATH(L1)
      KBD(I,2)=MPATH(L2)
      I=I+1
```

```
      GO TO 179
  115 LEX=I-1
      IF (IJND) 188,175,188
  188 M=0
      DO 329 I=1,IJND
      READ (KFILE,270) KEY,I1,I2
      GO TO (310,320,320),KEY
  310 M=M+1
      IJ(M,1)=MPATH(I1)
      IJ(M,2)=MPATH(I2)
      GO TO 329
  320 DO 328 K=1,3
      IMIKE=(MPATH(I1)/3)*3+K
      IF(COST(IMIKE)) 328,321,321
  321 IF(IMIKE-MPATH(I1))322,328,322
  322 M=M+1
      IJ(M,1)=IMIKE
      IJ(M,2)=MPATH(I2)
  328 CONTINUE
      GO TO (329,329,325),KEY
  325 I3=I2
      I2=I1
      I1=I3
      KEY=2
      GO TO 320
  329 CONTINUE
      IJND=M
  270 FORMAT (3I5)
  175 DO 190 I=1,LEX
      KOMP=KBD(I,2)
      IF (KLIST(KOMP,2)) 182,181,182
  181 KLIST(KOMP,2)=I
  182 DO 185 J=I,LEX
      IF (KBD(J+1,2)-KBD(I,2)) 185,184,185
  184 KBD(I,2)=J+1
      GO TO 190
  185 CONTINUE
      KBD(I,2)=0
  190 CONTINUE
      CALL JOBTM(KTIME)
      PRINT 1000,KTIME
 1000 FORMAT (10H TIME USED,I5)
      CALL CPM (KBD,FBD,KLIST,IJ,IJND,IO)
      CALL JOBTM(KTIME)
      PRINT 1000,KTIME
      CALL EXIT
      END
3.083+1.600
```

BONNIE MADTRN 05/22 1417.0

```
              SUBROUTINE CPM (KBD,FBD,KLIST,IJ,IJND,IQ)
              DIMENSION IQ(62),START(5000),SLNGTH(60,10),SL(10)
              DIMENSION MPATH(5000) ,BD(3,650),COST(200),TB1(10),LISTBN(100)
              DIMENSION TB2(10),LIVE(350),FBD(600),IJ(50,2)
              DIMENSION IFEAS(10),D(200),LISTEM(10),LPATH(10),TB3(10),KDC(60)
              DIMENSION LOUT(60),KBD(660,2),LSTDIJ(62,2),KLIST(200,2),KZERO(650)
              COMMON MPATH,BD,TB1,TB2,TB3,TCOST,K,LIM,COST,TIME,BON,TLEX,LM,M,JY
              LOCATE(MDUM,KDUM)=(2+LIM)*(MDUM-1)+KDUM
              READ (80,10)   (KDC(I),I=1,LIM)
         10   FORMAT(30I2)
              KLNGTH=5000/(LIM+2)
              DO 20 I=1,KLNGTH
         20   KZERO(I)=I
              BD(1,1)=0
              BD(2,1)=0
              BD(3,1)=0
              LEND=1
              KNTR=1
              MEND=1
              LPLAC=1
              M=1
              LIVE(1)=1
              LM=0
              IBN=0
              BESBND=9999999.
              MPATH(1)=0
              KEOP=3*(JY)+2
              D(KEOP-1)=1
              KSTOP=KEOP-2
              D(KEOP)=1
         70   FORMAT (15I4)
              KZTOP=0
              KZPLAC=1
       1000   GO TO  7000
       1900   LM=LOCATE(M,0)
              IQUIT=3-IQ(MIN)
              GO TO 8000
       2010   IF(LL-1) 3700,2100,2015
       2015   DO 2020 N=1,LL
              LT=LL-N
              DO 2019 L=1,LT
              LP1=LISTEM(L)
              LP2=LISTEM(L+1)
              IF (TB3(LP1)-TB3(LP2)) 2019,2019,2018
       2018   LTEMP=LISTEM(L)
              LISTEM(L)=LISTEM(L+1)
              LISTEM(L+1)=LTEMP
       2019   CONTINUE
       2020   CONTINUE
       2100   MPATH(LM+2)=1
              N=LISTEM(1)
              L=LPATH(N)+LM+2
              MPATH(L)=IN+N
              START(L)=SL(N)
              IF(LPATH(N)-LSF) 2109,2110,2110
       2109   L=LPATH(N)+1
```

```
      DO 2115 I=L,LSF
      J=LM+I+2
      MPATH(J)=LSTDIJ(I-1,1)
 2115 START(J)=SLNGTH(I-1,N)
 2110 BD(1,M)=TB1(N)
      BD(2,M)=TB2(N)
      BD(3,M)=TB3(N)
      IF(LL-1) 3700,3700,3000
 3000 KOUNT=LPLAC+1
      DO 3080 N=2,LL
      KNTR=KNTR+1
      KZPLAC=KZPLAC+1
      IF (KZPLAC-KLNGTH) 3050,3050,3060
 3060 PRINT 3334,LEND,KZTOP,KZPLAC
 3334 FORMAT (18H LEND KZTOP KZPLAC,3I5)
       KZPLAC=1
 3050 MEND=KZERO(KZPLAC)
      LMEND=LOCATE(MEND,0)
      MPATH(LMEND+1)=MPATH(LM+1)+1
      MPATH(LMEND+2)=1
      KN=LISTEM(N)
      LISTEM(N)=MEND
      IF (MPATH(LM+1)) 3051,3051,3055
 3051 MPATH(LMEND+3)=IN+KN
      START(LMEND+3)=SL(KN)
      GO TO 3086
 3055 LSP=LSF+2
      J=0
      DO 3085 I=3,LSP
      KB=LMEND+I
      IF(I-LPATH(KN)-2) 3070,3065,3070
 3065 MPATH(KB)=IN+KN
      START(KB)=SL(KN)
      GO TO 3085
 3070 J=J+1
      MPATH(KB)=LSTDIJ(J,1)
      START(KB)=SLNGTH(J,KN)
 3085 CONTINUE
 3086 BD(1,MEND)=TB1(KN)
      BD(2,MEND)=TB2(KN)
      BD(3,MEND)=TB3(KN)
 3080 CONTINUE
      IF (KOUNT-LEND) 3120,3120,3155
 3120 KTEMP=0
      DO 3150 LPUSH=KOUNT,LEND
      MIKE=LEND+LL-1-KTEMP
      MEL=LEND-KTEMP
      LIVE(MIKE)=LIVE(MEL)
 3150 KTEMP=KTEMP+1
 3155 DO 3200 N=2,LL
      MIKE=KOUNT+N-2
 3200 LIVE(MIKE)=LISTEM(N)
      LEND=LEND+LL-1
 3700 MPATH(LM+1)=MPATH(LM+1)+1
      IF(LL) 5000,5000,4000
 4000 IF(MPATH(LM+1)-LIM)1000,2125,2125
 2125 IF(BD(3,M)-BESBND)2150,2130,5000
 2130 IBN=IBN+1
      LISTBN(IBN)=M
      GO TO 4036
 2150 BESBND=BD(3,M)
```

```
      IBN=1
      LISTBN(IBN)=M
 4030 CALL JOBTM(MTI)
      PRINT 3333,LEND,MEND,BD(1,M),BD(2,M),BD(3,M),MTI
 3333 FORMAT (22H LEND MEND BOUNDS TIME,2I5,3F10.1,I5)
 4050 IF (LPLAC-LEND) 4100,4060,4060
 4060 LPLAC=1
      GO TO 4150
 4100 LPLAC=LPLAC+1
 4150 IF(LEND-IBN) 6000,6000,4151
 4151 M=LIVE(LPLAC)
      LM=LOCATE(M,0)
      IF (BD(3,M)-BESBND) 4000,4000,5000
 5000 IF (LEND-LPLAC) 5001,5001,5005
 5001 LPLAC=1
      GO TO 5011
 5005 DO 5010 N=LPLAC,LEND
 5010 LIVE(N)=LIVE(N+1)
 5011 LEND=LEND-1
      IF(KZTOP-KLNGTH) 5020,5030,5030
 5020 KZTOP=KZTOP+1
      GO TO 5035
 5030 KZTOP=1
 5035 KZERO(KZTOP)=M
      GO TO 4150
 7000 MIN=MPATH(LM+1)
      IN=(KDC(MIN+1)-1)*3
      KN=MPATH(LM+1)+2
      MIN=KDC(MIN+1)
      GO TO 1900
 8000 LL=0
      KSTOP=KEOP-2
      DO 8011 I=1,KSTOP
 8011 D(I)=0
      LSF=0
      IF (KN-2) 9260,9260,9210
 9210 DO 9250 L=3,KN
      MIKE=LM+L
      I=MPATH(MIKE)
      LSF=LSF+1
      LSTDIJ(LSF,1)=I
      LSTDIJ(LSF,2)=KLIST(I,1)
 9250 D(I)=1
 9260 LSF=LSF+1
      LSTDIJ(LSF,1)=KEOP
      LSTDIJ(LSF,2)=KLIST(KEOP,1)
      DO 9900 K=1,IQUIT
      IO=IN+K
      D(IO)=1
      DO 8220 I=1,LSF
      IF (KLIST(IO,1)-LSTDIJ(I,2)) 8230,8220,8220
 8220 CONTINUE
 8230 LPATH(K)=I
      IF(IJND) 8030,8800,8030
 8030 IF (KN-2) 8800,8800,8050
 8050 DO 8300 IL=1,IJND
      IMIKE=IJ(IL,1)
      IMEL=IJ(IL,2)
      TOST=D(IMIKE)+D(IMEL)-2.
      IF (TOST) 8300,9900,8300
```

```
      8300 CONTINUE
      8800 IBO=I-1
           IF(IBO) 8805,8810,8805
      8805 DO 8806 IZ=1,IBO
           MIKE=LM+2+IZ
      8806 SLNGTH(IZ,K)=START(MIKE)
      8810 CALL TOME(KBD,KLIST,D,FBD,LSF,LSTDIJ,I,IO,SLNGTH,SL)
           CKOST=COST(IO)
           CALL MONEY (CKOST)
           IF(TB3(K)-BESBND) 9400,9400,9500
      9400 LL=LL+1
           LISTEM(LL)=K
           GO TO 9900
      9500 CONTINUE
      9900 D(IO)=0
           GO TO 2010
      6000 DO 6020 IK=1,IBN
           I=LISTBN(IK)
           MIKE=LOCATE(I,0)+2
           JPQ=LIM+2
           PRINT 6070,BD(1,I),BD(2,I),BD(3,I)
           DO 6025 J=3,JPQ
           MIKE=MIKE+1
           NIKE=MPATH(MIKE)
      6025 LOUT(J)=KLIST(NIKE,1)
      6020 PRINT 6050,I,(LOUT(J),J=3,JPQ)
           DO 6030 N=1,IBN
      6030 PRINT 6060,BESBND,LISTBN(N)
           PRINT 6071,KNTR
      6071 FORMAT(5H KNTR,I5)
      6050 FORMAT (1X,I4,5H PATH,11I6/(11X,10I6))
      6060 FORMAT (1X,F10.1,5X,I4)
      6070 FORMAT (1X,7H BOUNDS,3F10.1)
           PRINT 6999,(KDC(I),I=1,LIM)
      6999 FORMAT (11H NODE ORDER,15I4)
           RETURN
           END
.083+2.783
```

CLYDE MADTRN 05/22 1419.0

```
      SUBROUTINE TOME(KBD,KLIST,D,FBD,LST,LSTDIJ,INSERT,IO,SLNGTH,SL)
      DIMENSION SLNGTH(60,10),SL(10)
      DIMENSION D(200),LSTDIJ(62,2),KLIST(200,2),KBD(600,2),LPATH(10)
      DIMENSION TLNGTH(200),TB1(10),FBD(600),MPATH(5000),BD(3,650)
      DIMENSION TB3(10),TB2(10),COST(200)
      COMMON MPATH,BD,TB1,TB2,TB3,TCOST,K,LIM,COST,TIME,BON,TLEX,LM,M,JY
      MIKE=3*JY+2
      DO 150 I=1,MIKE
  150 TLNGTH(I)=0
      IBO=INSERT-1
      IF(IBO) 120,137,120
  120 DO 125 I=1,IBO
      KO=LSTDIJ(I,1)
  125 TLNGTH(KO)=SLNGTH(I,K)
  137 I=IBO
      LST=LST+1
      DO 200 J=INSERT,LST
      IF (J-INSERT) 130,135,130
  130 I=I+1
      NOW=LSTDIJ(I,1)
      GO TO 155
  135 NOW=IO
  155 KNOW=KLIST(NOW,2)
  100 ITEST=KBD(KNOW,1)
      IF (D(ITEST)) 300,300,250
  250 F=TLNGTH(ITEST)+FBD(KNOW)
      IF (TLNGTH(NOW)-F) 260,300,300
  260 TLNGTH(NOW)=F
  300 KNOW=KBD(KNOW,2)
      IF (KNOW) 100,200,100
  200 CONTINUE
      TB1(K)=TLNGTH(MIKE)
      LST=LST-1
      DO 900 I=INSERT,LST
      KO=LSTDIJ(I,1)
  900 SLNGTH(I,K)=TLNGTH(KO)
      SL(K)=TLNGTH(IO)
      RETURN
      END
```

R 1.900+.066

```
      SUBROUTINE MONEY(CKOST)
      DIMENSION MPATH(5000),BD(3,650),TB1(10),TB2(10),TB3(10),COST(200)
      COMMON MPATH,BD,TB1,TB2,TB3,TCOST,K,LIM,COST,TIME,BON,TLEX,LM,M,JY
      TB2(K)=BD(2,M)+CKOST
      A1=TB1(K)-TIME
      IF(A1) 9650,9670,9670
 9650 A1=TLEX*A1
      GO TO 9680
 9670 A1=BON*A1
 9680 TB3(K)=TB2(K)+A1+TCOST
      RETURN
      END
```

.900+.316

BIBLIOGRAPHY

BIBLIOGRAPHY

1. Agin, N. "Optimum Seeking with Branch and Bound," Management Science, Vol. 13, No. 4, December 1966.

2. Alpert, L. and D. S. Orkand "A Time-Resource Trade-Off Model for Aiding Management Decisions" Operations Research Inc., Technical Papers No. 12. Silver Spring, Maryland (1962).

3. Bali ski, M. L. "Integer Programming: Methods, Uses, Computation" Management Science, Vol. 12, No. 3, November 1965.

4. Barnes, R. M. Work Methods Manual, John Wiley and Sons Inc., New York (1944).

5. Bellman, R. "Some Applications of the Theory of Dynamic Programming - A Review," Operations Research, Vol. 2, No. 3, August 1954.

6. Bellman, R., "Mathematical Aspects of Scheduling Theory," Journal of the Society for Industrial and Applied Mathematics, Vol. 4, No. 3, 1956.

7. Berman, E. B. "Resource Allocation in a PERT Network Under Continuous Activity Time-Cost Functions," Management Science, Vol. 10, No. 4, July 1964.

8. Bowman, E. H. "The Schedule Sequencing Problem," Operations Research Vol. 7, No. 5, September-October 1959.

9. Brooks, G. H. and White, C. R., "An Algorithm for Finding Optimal and Near Optimal Solutions to the Production Scheduling Problem," Journal of Industrial Engineering, Vol. 16, No. 1, January-February 1965.

10. Brown, A. P. G. and Z. A. Lomnicki "Some Application of the 'Branch and Bound' Algorithm to the Machine Scheduling Problem," Operational Research Quarterly, Vol. 17, No. 2, June 1966.

11. Buffa, E. S., G. Armour, C. Gordon and T. E. Vollman "Allocating Facilities with CRAFT," Harvard Business Review, Vol. 42, No. 2, March-April 1964.

12. Burgess, A. R. and J. B. Killebrew "Variation in Activity Level on a Cyclical Arrow Diagram," Journal of Industrial Engineering, Vol. 13, No. 2, March-April 1962.

Bibliography, Cont'd.

13. Battersby, A. "Network Analysis," St. Martin'e Press, 1964.

14. Carroll, D. C. "Heuristic Sequencing of Single and Multi-Component Orders," Ph.D. Thesis, M.I.T., 1965.

15. Charnes, A. and W. W. Cooper "A Network Interpretation and a Directed Subdual Algorithm for Critical Path Scheduling," Journal of Industrial Engineering, Vol. 13, No. 4, August 1962.

16. Charnes, A., W. W. Cooper, J. E. Devoe, and D. B. Learner "Demon Decision Mapping Via Optimum Go-No Networks - A Model for Marketing New Products," Management Science, Vol. 12, No. 11, July 1966.

17. Clermont, P. "Substitution in a Job Shop," M.Sc. Thesis, M.I.T., 1966.

18. Conway, R. W. and W. L. Maxwell "Network Scheduling by the Shortest Operation Discipline," Industrial Scheduling, Muth and Thompson Eds., Prentice-Hall Inc., 1963.

19. Crowston, W. B. and G. L. Thompson "Decision CPM: A Method for Simultaneous Planning, Scheduling and Control of Projects," Operations Research, Vol. 15, No. 3, May-June 1967.

20. Crowston, W. B., F. Glover, G. L. Thompson and J. Trawick "Probabilistic Combinations of Local Job Shop Scheduling Rules," O.N.R. Research Memorandum, No. 117, Carnegie Institute of Technology, 1964.

21. Crowston, W. B. and G. L. Thompson "Reduction and Solution of Decision Networks," Working Paper, Sloan School of Management, Forthcoming.

22. Crowston, W. B. and R. E. Smylie "The Application of Decision CPM to Incentive Contracts," Working Paper, Sloan School of Management, Forthcoming.

23. Crowston, W. B. and M. Wagner "A Comparison of Tree Search Schemes for Decision Networks," Working Paper, Sloan School of Management, Forthcoming.

24. Dantzig, G. B. "Discrete Variable Extremum Problems," Operations Research, Vol. 5, No. 2, April 1957.

25. Davis, Edward W. "Resource Allocation in Project Network Models - A Survey,' The Journal of Industrial Engineering, Vol. XVII, No. 4, April 1966.

Bibliography, Cont'd.

26. "DOD and NASA Guide, PERT/Cost," Office of the Secretary of Defence and National Aeronautics and Space Administration, Washington D.C., 1962.

27. Eastman, W. L., "Linear Programming with Pattern Constraints," Ph.D. Thesis, Harvard University, 1958.

28. Efroymson, M. A. and T. L. Ray "A Branch-Bound Algorithm for Plant Location," Operations Research, Vol. 14, No. 2, May-June 1966.

29. Eisner, H. "A Generalized Network Approach to the Planning and Scheduling of a Research Project," Operations Research, Vol. 10, No. 1, January-February 1962.

30. Elmaghraby, S. F. "An Algebra for Analysis of Generalized Activity Networks," Management Science, Vol. 10, No. 3, April 1964.

31. Fernando, E. P. C. 'The Implementation of Industrial Development Programmes Using Critical Path Network Theory," Working Paper 149, Sloan School of Management, M.I.T., 1965.

32. Ford, L. P. Jr. and D. R. Fulkerson "A Simple Algorithm for Finding Maximum Network Flows and an Application to the Hitchcock Problem," Canadian Journal of Mathematics, Vol. 9, 1957.

33. Ford, L. P. Jr. and D. R. Fulkerson "Flows in Networks," Princeton University Press, 1963.

34. Freeman, Raoul J. "A Generalized PERT." Operations Research, Vol. 8 No. 2, May-June 1960.

35. Fulkerson, E. R. "A Network Flow Computation for Project Cost Curves," Management Science, Vol. 7, No. 2, January 1961.

36. Gannt, H. L. Organizing for Work, Harcourt, New York, 1919.

37. Gavett, J. W. and N. V. Plyter "The Optimal Assignment of Facilities to locations by Branch and Bound," Operations Research, Vol. 14, No. 2, March-April 1966.

38. Gere, W. S. Jr. "A Heuristic Approach to Job Shop Scheduling," Ph.D. Thesis, Carnegie Institute of Technology, 1962.

Bibliography, Cont'd.

39. Gilmore, P. C. and R. E. Gomory "A Linear Programming Approach to the Cutting Stock Problem," Operations Research, Vol. 9, No. 6 November-December 1961.

40. Glover, F. "The Knapsack Problem: Some Relations for an Improved Algorithm," Management Sciences Research Report, No. 28, Carnegie Institute of Technology, 1965.

41. Gomory, R. A. "An All-Integer, Integer Programming Algorithm" Industrial Scheduling, Thompson and Muth, Eds., Prentice-Hall, Inc.

42. Jackson, J. R. "A Computing Procedure for a Line Balancing Problem," Management Science, Vol. 2, No. 3, 1956.

43. Johnson, T. J. R. "An Algorithm for the Resource Constrained Project Scheduling Problem," Ph.D. Thesis, Sloan School of Management, M.I.T., 1967.

44. Kelley, J. E. Jr. and M. R. Walker "Critical Path Planning and Scheduling," 1959 Proceedings of the Eastern Joint Computer Conference.

45. Kelley, J. E. Jr. "The Critical Path Method: Resources Planning and Scheduling," Industrial Scheduling, Muth and Thompson, Eds., Prentice-Hall Inc., 1963.

46. Knight, R. M. "Resource Allocation and Multi-Project Scheduling in a Research and Development Environment," S. M. Thesis, M.I.T., 1966.

47. Lawler, E. L. and D. E. Wood "Branch-and-Bound Methods: A Survey," Operations Research, Vol. 14, No. 4, July-August 1966.

48. Lawler, E. L. "The Quadratic Assignment Problem," Management Science, Vol. 9, No. 4, July 1963.

49. Levy, F. K., G. L. Thompson and J. D. Wiest "Mathematical Basis of the Critical-Path Method," Industrial Scheduling, Thompson and Muth, Eds., Prentice-Hall Inc., 1963.

50. Levy, F. K., G. L. Thompson and J. D. Wiest "Multi-Ship, Multi-Shop Workload Smoothing Program," Naval Research Logistics Quarterly, Vol. 9, No. 1, March 1962.

Bibliography, Cont'd.

51. Little, J. D. C., K. G. Murty, D. W. Sweeney, and C. Karel "An Algorithm for the Traveling-Salesman Problem," Operations Research Vol. 11, No. 6, November-December 1963.

52. Lorie, J. H. and L. J. Savage "Three Problems in Capital Budgeting," Journal of Business, XXVIII, No. 4, October 1955.

53. MacCrimmon, Kenneth R. and Charles A. Ryavec "An Analytical Study of the PERT Assumptions," Operations Research, Vol. 12, No. 1, January-February 1964.

54. Manne, A. S. "On the Job Shop Scheduling Problem," Operations Research, Vol. 8, No. 2, March-April 1960.

55. Mansoor, E. M. "Improvement on Gujahr and Nemhauser's Algorithm for the Line Balancing Problem," Management Science, Vol. 14, No. 3, November 1967.

56. Marglin, S. A. Approaches to Dynamic Investment Planning, North Holland Publishing Company, 1963.

57. Meyer, W. L. and L. R. Shaffer Extensions of the Critical Path Method Through the Application of Integer Programming, Report Issued By the Department of Civil Engineering, University of Illinois, Urbana, Illinois, July 1963.

58. Moder, J. J. and C. R. Phillips Project Management with CPM and PERT, Rheinhold Publishing Corporation, 1964.

59. NASA, "Incentive Contracting Guide," NTC 403, National Aeronautics and Space Administration, January 1965.

60. Philips, C. R. "Fifteen Key Features of Computer Programs for CPM and PERT," The Journal of Industrial Engineering, Vol. 15, No. 1, January-February 1964.

61. Pierce, J. F. Some Large Scale Production Scheduling Problems in the Paper Industry, Prentice-Hall Inc., 1964.

62. Pierce, J. F. "Application of Combinatorial Programming to a class of All-Zero One Integer Programming Problems," IBM Cambridge Scientific Center Report 36,403, July 1966.

Bibliography, Cont'd.

63. Root, J. G. "An Application of Symbolic Logic to the Selection Problem," Operations Research, Vol. 12, No. 4, July-August 1964.

64. Rowe, A. "Towards a Theory of Scheduling," Journal of Industrial Engineering, Vol. 11, No. 2, March-April 1960.

65. Russo, F. J. "A Heuristic Approach to Alternate Routing in a Job-Shop," M.Sc. Thesis, M.I.T., 1965.

66. Salveson, M. E. "The Assembly Line Balancing Problem," The Journal of Industrial Engineering, Vol. 6, No. 3, May-June 1955.

67. Shapiro, Donald M. "Algorithms for the Solution of the Optimal Cost and Bottleneck Traveling-Salesman Problem," Washington University Sc.D. Thesis, 1966, Mathematics.

68. Shapiro, J. E. "A Group Theoretic Branch and Bound Algorithm for the Zero-One Integer Programming Problem," M.I.T. Sloan School Working Paper, December 1967.

69. Smylie, R. E. "The Application of Decision CPM to Incentive Contracts," S.M. Thesis, Sloan School of Management, M.I.T., 1967.

70. Thompson, G. L. "The StoppedSimplex Method: I. Basic Theory for Mixed Integer: Integer Programming," Revue Francaise de Recherche Operationnelle, 1964.

71. Thompson, G. L. "Recent Developments in the Job Shop Scheduling Problem," Naval Research Logistics Quarterly, Vol. 7, No. 4, December 1960.

72. Tonge, F. M. A Heuristic Program for Assembly Line Balancing, Prentice-Hall Inc., 1961.

73. Wagner, H. M. "An Integer Linear-Programming Model for Machine Scheduling," Naval Research Logistics Quarterly, Vol. 7, No. 4, December 1960.

74. Weingartner, H. M. and D. N. Ness "Methods for the Solution of the Multi-Dimensional 0/1 Knapsack Problem," Working Paper 177-66 Sloan School of Management, M.I.T., April 1966.

75. Weingartner, H. M. "Capital Budgeting of Interrelated Projects: Survey and Synthesis," Management Science, Vol. 12, No. 2, March 1966.

Bibliography, Cont'd.

76. Weingartner, H. M. Mathematical Programming and the Analysis of Capital Budgeting Problems, Prentice-Hall, Inc., 1963.

77. Wester, L. and M. P. Killbridge "A Heuristic Method of Line Balancing," The Journal of Industrial Engineering, Vol. 12, No. 4, July-August 1961.

78. Wester, L. and M. P. Killbridge "Heuristic Line Balancing: A Case," The Journal of Industrial Engineering, Vol. 13, No. 3, May-June 1962.

79. Weist, J. D. "Some Properties of Schedules for Large Projects with Limited Resources," Operations Research, Vol. 12, No. 3, May-June 1964.

80. Wiest, J. D. "The Scheduling of Large Projects with Limited Resources," Ph.D. Thesis, Carnegie Institute of Technology, 1962.

81. Wiest, Jerome D. "A Heuristic Model for Scheduling Large Projects with Limited Resources," Management Science, Vol. 13, No. 6, February 1967.

82. Wilson, R. C. "Assembly Line Balancing and Resource Levelling," University of Michigan Summer Conference, Production and Inventory Control, 1964.

Unclassified
Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Graduate School of Industrial Administration Carnegie-Mellon University	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP Not applicable

3. REPORT TITLE

DECISION NETWORK PLANNING MODELS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)
Technical Report, May 1968

5. AUTHOR(S) (Last name, first name, initial)

Wallace B. S. Crowston

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
May 29, 1968	208	82

8a. CONTRACT OR GRANT NO. NONR 760(24) b. PROJECT NO. NR 047-048 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) Management Sciences Research Report No. 138 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)

10. AVAILABILITY/LIMITATION NOTICES
Distribution of this document is unlimited.

DECISION NETWORK PLANNING MODELS

11. SUPPLEMENTARY NOTES None	12. SPONSORING MILITARY ACTIVITY Logistics and Mathematical Statistics Branch Office of Naval Research Washington, D. C. 20360

13. ABSTRACT

See front pages 3 and 4.

DD FORM 1 JAN 64 1473

Unclassified
Security Classification

Unclassified
Security Classification

14. KEY WORDS	LINK A ROLE	LINK A WT	LINK B ROLE	LINK B WT	LINK C ROLE	LINK C WT
Decision Critical Path Method						
Critical Path Method						
Integer Programming Application						
Branch and bound application						
Project planning models						

INSTRUCTIONS

1. ORIGINATING ACTIVITY: Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. REPORT SECURITY CLASSIFICATION: Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. GROUP: Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. REPORT TITLE: Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. DESCRIPTIVE NOTES: If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. AUTHOR(S): Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. REPORT DATE: Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. TOTAL NUMBER OF PAGES: The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. NUMBER OF REFERENCES: Enter the total number of references cited in the report.

8a. CONTRACT OR GRANT NUMBER: If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. PROJECT NUMBER: Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. ORIGINATOR'S REPORT NUMBER(S): Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. OTHER REPORT NUMBER(S): If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. AVAILABILITY/LIMITATION NOTICES: Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through
__________________________________ ."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through
__________________________________ ."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through
__________________________________ ."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. SUPPLEMENTARY NOTES: Use for additional explanatory notes.

12. SPONSORING MILITARY ACTIVITY: Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. ABSTRACT: Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS)*, *(S)*, *(C)*, or *(U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. KEY WORDS: Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, roles, and weights is optional.

Unclassified
Security Classification