WORKING PAPER
ALFRED P. SLOAN SCHOOL OF MANAGEMENT

DECISION CPM: A METHOD FOR SIMULTANEOUS PLANNING, SCHEDULING AND CONTROL OF PROJECTS

By

W. Crowston and G. L. Thompson

187-66

MASSACHUSETTS
INSTITUTE OF TECHNOLOGY
50 MEMORIAL DRIVE
CAMBRIDGE, MASSACHUSETTS 02139

Management Sciences Research Report No 45

DECISION CPM: A METHOD FOR SIMULTANEOUS PLANNING, SCHEDULING AND CONTROL OF PROJECTS

By

W. Crowston and G. L. Thompson

187-66

June, 1965

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie Institute of Technology, under Contract Nonr 760(24), NR 047-048 with the U. S. Office of Naval Research. Distribution of this document is unlimited. Reproduction of this paper in whole or in part is permitted for any purpose of the United States Government

MANAGEMENT SCIENCES RESEARCH GROUP
GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION
CARNEGIE INSTITUTE OF TECHNOLOGY
PITTSBURGH, PENNSYLVANIA 15213

HD28
.M414
no.157-66

1. Introduction

Critical Path analysis is commonly considered to be a technique for planning and scheduling of projects. The planning phase is usually identified with the construction of the project graph, during which time specific decisions are made on the method of performing jobs as well as their technological ordering. At the same time standard times are assigned to these jobs. At the completion of the planning stage it is possible, using the conventional CPM calculations, to schedule the starting time of each job in the project. Unless several different plans are evaluated in this way, or unless the technique of job crashing is used, there is no interaction between the planning and the scheduling phase of the usual CPM analysis.

We shall show in this paper that if an overall optimum is to be obtained a much greater degree of interaction is essential, and shall give methods for solving the more general problem. Thus if there are a number of competing methods of performing some of the jobs, each method having a different cost, a different time duration and different technological dependencies, we shall include all these in the project graph, rather than making the decisions in advance. Then in the scheduling phase, we shall consider the effects of all alternate methods of performing a task on the total cost of completing the project and choose those alternatives which minimize this cost. We may apply the same method to the control of projects being carried out. Thus decisions previously optimal, may be changed during the execution of the project due to certain jobs being delayed. We call the complete problem

Decision CPM,[1] and we shall show how to set it up as a formal mathematical problem and to solve it using various techniques.

In Section 2 a decision project graph is defined which contains information on all jobs to be completed in a project, alternative methods of performing some of the jobs and precedence relations between the jobs. A simple example is presented in Section 3. Section 4 and 5 present integer programming and heuristic techniques for solving such a graph for the set of jobs which are to be performed and the criticality of these jobs. Finally Section 6 will briefly discuss the application of the technique to dynamic monitoring and control of projects during their execution.

1/ The authors of references (2) and (4) have discussed networks with decision nodes but in both cases they refer to decision nodes with probabilistic research outcomes. In addition S. F. Elmahgraby (5) has given a more general discussion of probabilistic decision nodes.

2. The Mathematical Basis of Decision CPM

This section will follow in part the article [9] 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, some of the jobs from each job set must be completed. Associate with each job set

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

k(i) variables

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

having the property that

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

If exactly one of the jobs must be performed (this may or may not be the case) then the mutually exclusive interdependence condition is expressed by

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

Many other possibilities exist, as we shall see.

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) $$d_{ij} + d_{mn} \leq 1$$

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

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

(d) $$d_{ij} \leq d_{hk} + d_{mn} - d_{hk}d_{mn}$$

Finally the design problem may not always be the simple choice of one job from each job set. For instance,

(e) $$\sum_{j=1}^{k(i)} d_{ij} = 2$$

(f) $$\sum_{j=1}^{k(i)} d_{ij} \leq 5$$

(g) $$\sum_{j=1}^{k(i)} d_{ij} + \sum_{j=1}^{k(\ell)} d_{\ell k} = 3.$$

Note that (e) says that we must choose exactly two alternatives; (f) says we may choose at most 5 alternatives; and (g) says that at the two decision nodes i and ℓ we must choose exactly 3 alternatives.

The above discussion illustrates some of the possible richness 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 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 $S_{ij} = a_1 \ll a_2 \ll . . . \ll a_n = a_1 = S_{ij}$. 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

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

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.

* Note that this assumption differs from the corresponding assumption in (9) in that the requirement of K-intransitivity is omitted. For this reason theorem 1 of that reference does not hold in the present context.

Definition: A relation that is transitive and asymmetric is said to be a <u>preference relation</u>.

Theorem* 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 $J^* = \{a_1, a_2 \ldots a_n\}$ 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* 2: Assumption 1 holds if and only if it is possible to list the jobs in J in a technologically ordered job list J*.

In addition to these definitions and theorems from reference (9) 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, 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

* The proofs of these theorems are exactly as in reference (9).

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.

3. Decision Project Graphs

A graphical representation of the combined planning and scheduling problem is shown in the decision project graphs of Figures 1 and 5. In these graphs the circular nodes represents jobs and the triangular nodes introduce the mutually exclusive job nodes of a job set. In Figure 1 we have the additional interdependence of a contingent relationship between jobs S_{51} and S_{22}. We may include job S_{22} if and only if we perform job S_{51}. Therefore the possible sets of decisions are $\{S_{21}\ S_{51}\}$, $\{S_{21}\ S_{52}\}$ or $\{S_{22}\ S_{51}\}$. The project graphs resulting from each of these sets of decisions are shown in Figure 2, 3 and 4 respectively. It should be noted in Figure 1 that the links $\begin{tikzcd} S_1 \arrow[r] & S_7 \end{tikzcd}$, $\begin{tikzcd} S_1 \arrow[r] & S_3 \end{tikzcd}$ and $\begin{tikzcd} S_4 \arrow[r] & S_6 \end{tikzcd}$ are only necessary because jobs S_{21} and S_{51} are members of job sets.

The total project cost given any set of decisions will vary with the due date established because of overtime penalties and early finish premiums. For example if jobs S_{21}, S_{52} are selected the cost of performing all jobs would be \$1650. Since it would require 97 days to finish the project, a due date of 98 days would give the project a total cost of \$1650. However given a daily penalty and premium of \$150 and \$25 respectively a due date of 97 would cost \$1650 + 150 = \$1800 and a due date of 101 days would cost \$1650 - (3)(25) = \$1575. Exhibit I illustrates how total cost changes with due date for each set of decisions. It also indicates that the optimal decisions (as indicated by the boxes) vary with the given due date. This example demonstrates the difficulty of making decisions in this project without the knowledge of scheduling information and due date.

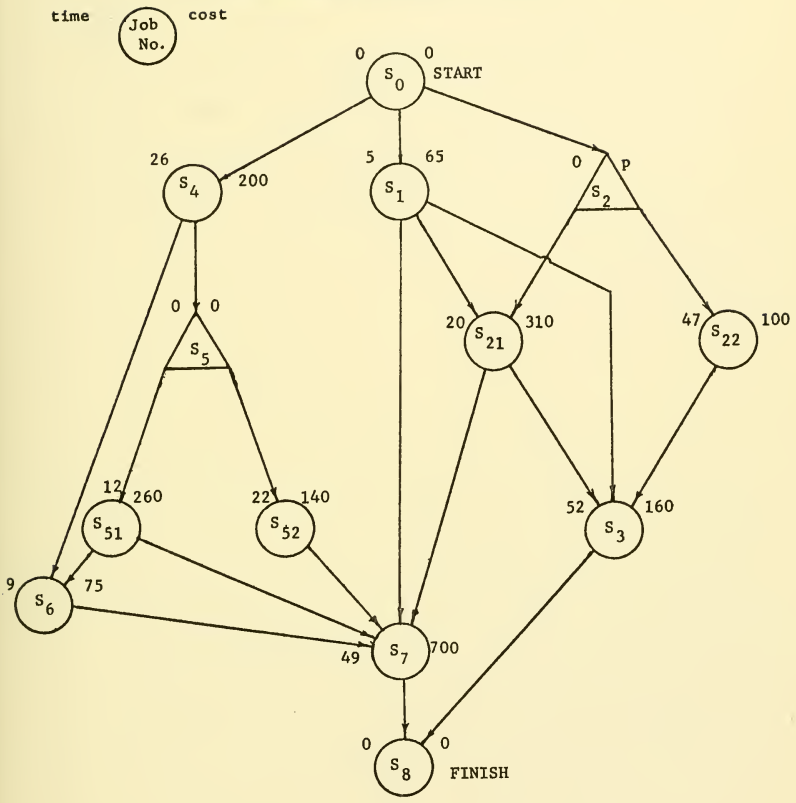
Due Date	Decision Set S_{21} S_{51}	S_{21} S_{52}	S_{22} S_{51}
101	1670	1575	$\boxed{1535}$
99	1720	$\boxed{1625}$	1710
97	$\boxed{1770}$	1800	2010

Total Cost of Project with Given Decision

Sets and Due Date

Exhibit I

DECISION PROJECT GRAPH - No. 1



s.t. $d_{51} \geq d_{22}$

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

LATE PENALTY $150
EARLY PREMIUM 25

FIGURE 1

PROJECT GRAPH I - DECISIONS S_{21}, S_{51}

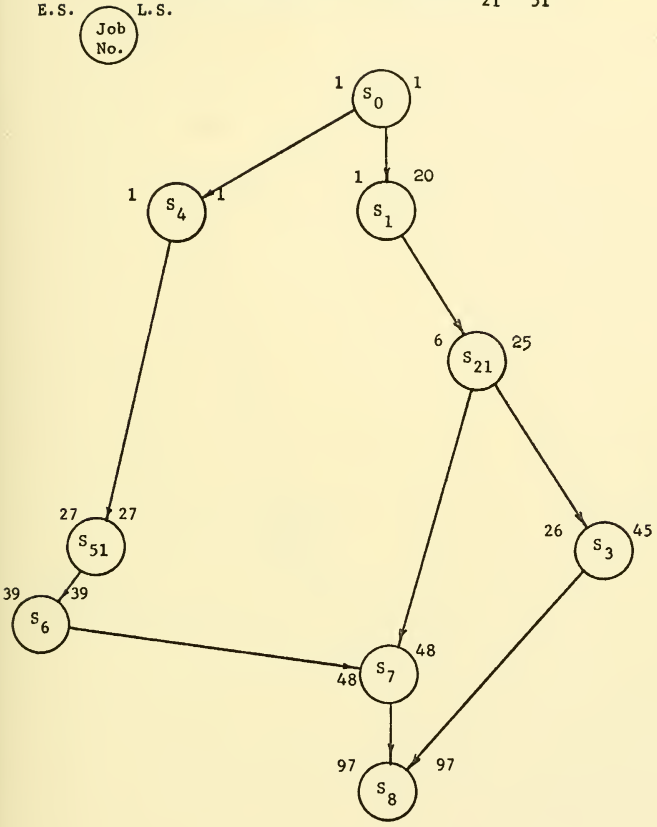


FIGURE 2

PROJECT GRAPH I - DECISIONS S_{21}, S_{52}

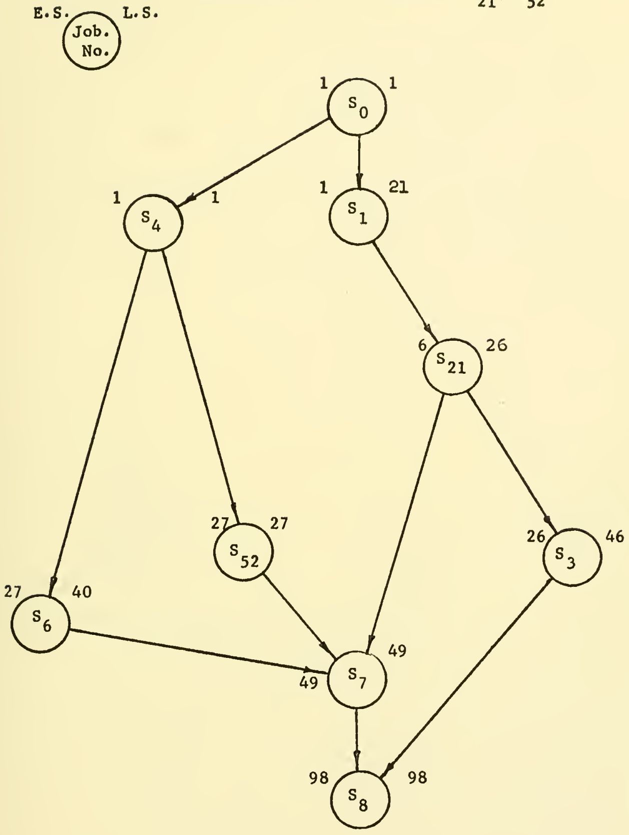


FIGURE 3

PROJECT GRAPH I - DECISIONS S_{22}, S_{51}

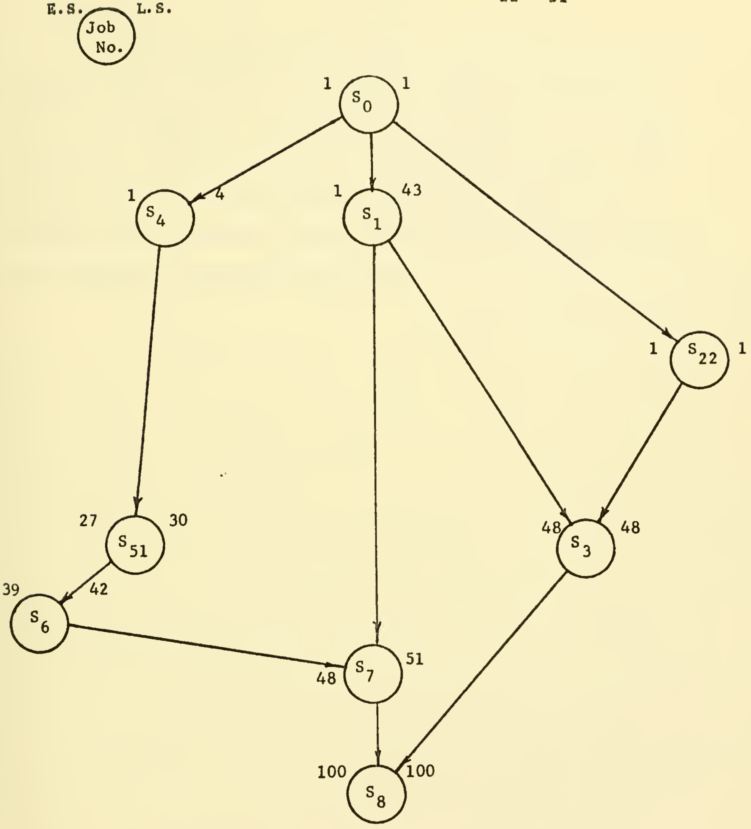


FIGURE 4

It is relatively simple matter to solve very small decision project graphs by making a complete listing of alternatives. Experimental results of this approach will be discussed in Section 5. It is clear however that the number of possible combinations rises rapidly. For example the decision project graph of Figure 5 has a total of 864 possible decision patterns. It has therefore been necessary to develop other techniques to handle large problems.

4. Decision Graph Solution by Integer Programming

Consider a 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 discussed in Section 2 of this article or other constraints showing various types of complicated interdependencies between jobs in the project.

As in the graph of Figure 1, we associate with each job, S_{ij}, a time, t_{ij} , and a cost, c_{ij}. Also we assume a reward payment 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. We can now formulate the integer programming problem of selecting the best project graph and finding its critical path.

(5) $$\text{Min} \sum_{i=1}^{h} \sum_{j=1}^{k(i)} d_{ij} c_{ij} - rW_F^- + pW_F^+$$

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

(6) $$0 \leq d_{ij} \leq 1$$

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

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

(8) $$W_F - W_F^+ + W_F^- - D = 0$$

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 in the objective function. We assume that $p > r$ so that not both W_F^+ and W_F^- will be in the basis.

Other constraints must hold because of precedence relationships. For instance if S_i and S_m are <u>unit set jobs</u> and $S_i \ll S_m$ we have

$$W_i + t_i \leq S_m \tag{9}$$

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

$$-M(1 - d_{ij}) + W_{ij} + t_{ij} \leq W_m \tag{10}$$

where M is a large enough number so that the inequality is restrictive only if $d_{ij} = 1$. If S_{ij} is not performed (i.e., $d_{ij} = 0$) the inequality does not constrain the variables. Thus all paths through jobs which are not performed will be broken.

It is now possible to set up the problem of Figure 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 start and decision nodes in the precedence constraints.

Minimize $150W_F^+ - 25W_F^- + 260d_{51} + 140d_{52} + 310d_{21} + 100d_{22}$

<u>St</u>

Precedence

$$t_1 \le W_3$$
$$t_1 \le W_{21}$$
$$t_1 \le W_7$$
$$t_4 \le W_6$$
$$t_4 \le W_{51}$$
$$t_4 \le W_{52}$$
$$t_6 + W_6 \le W_7$$
$$-M(1 - d_{51}) + t_{51} + W_{51} \le W_6$$
$$-M(1 - d_{51}) + t_{51} + W_{51} \le W_7$$
$$-M(1 - d_{52}) + t_{52} + W_{52} \le W_7$$
$$-M(1 - d_{21}) + t_{21} + W_{21} \le W_7$$
$$-M(1 - d_{21}) + t_{21} + W_{21} \le W_3$$
$$-M(1 - d_{22}) + t_{22} + W_{22} \le W_3$$
$$t_7 + W_7 \le W_8$$
$$t_3 + W_3 \le W_8$$

Due Date $\quad W_8 - W_F^+ + W_F^- = D$

Interdependence

$$d_{51} + d_{52} = 1$$
$$d_{21} + d_{22} = 1$$
$$d_{22} \le d_{51}$$

$$0 \le d_{ij} \le 1 \quad \text{integer}$$

This problem was solved by the stopped simplex algorithm due to G. L. Thompson (11) in approximately 5 seconds (IBM 7090) for each of the three cases given in Exhibit 1.

As stated above, each constraint in the precedence set represents an immediate predecessor relationship in the original graph. The precedence relationships could also be represented by a series of constraints, a separate constraint being included for each path in the original graph. For the problem of Figure 1 the precedence constraints would be

(1) $t_4 + t_6 + t_7 \leq W_8$

(2) $t_4 - M\,(1-d_{51}) + t_{51} + t_6 + t_7 \leq W_8$

(3) $t_4 - M\,(1-d_{52}) + t_{52} + t_7 \leq W_8$

(4) $t_1 + t_7 \leq W_8$

(5) $t_1 + t_3 \leq W_8$

(6) $t_1 - M\,(1-d_{21}) + t_{21} + t_3 \leq W_8$

(7) $- M\,(1-d_{21}) + t_{21} + t_7 \leq W_8$

(8) $- M\,(1-d_{21}) + t_{21} + t_3 \leq W_8$

(9) $- M\,(1-d_{22}) + t_{22} + t_3 \leq W_8$

where M is a large positive number.

From the numerical data of the problem

$$t_4 + t_6 + t_7 > t_1 + t_3 > t_1 + t_7$$

therefore equations (4) and (5) may be dropped if equation (1) is retained. Also

$$t_1 + t_{21} + t_3 > t_{21} + t_3 > t_{21} + t_7$$

therefore equations (7) and (8) are eliminated by equation (6). Finally, since either S_{51} or S_{52} must be performed, a lower bound on W_8 may be set as the minimum of $t_4 + t_{51} + t_6 + t_7$, (2) or $t_4 + t_{52} + t_7$, (3). Since the minimum, $t_4 + t_{51} + t_6 + t_7$, is $> t_4 + t_6 + t_7 > t_1 + t_{21} + t_3$, equations (1) and (6) are also eliminated.

Thus the final "Reduced Path" formulation of the problem of Figure 1 is written as

$$\text{Minimize } 150\, W_F^+ - 25\, W_F^- + 260\, d_{51} + 140\, d_{52}$$
$$+310\, d_{21} + 100\, d_{22}$$

<u>S.t.</u>

Reduced Paths $\quad -M(1 - d_{51}) + 96 \leq W_8$

$$-M(1 - d_{52}) + 97 \leq W_8$$
$$-M(1 - d_{22}) + 99 \leq W_8$$

Due Date

$$W_8 - W_F^+ = W_F^- = D$$

Interdependence

$$d_{51} + d_{52} = 1$$
$$d_{21} + d_{22} = 1$$
$$d_{22} \leq d_{51}$$
$$0 \leq d_{ij} \leq 1 \text{ integer}$$

This problem was solved by the stopped simplex algorithm in approximately (2) seconds (IBM 7090) in each of the three cases given in Exhibit 1.

Finally it should be pointed out that it is not necessary to determine the reduced set of paths from the complete set of feasible paths in the graph. It can be shown that the reduced set may be derived directly from the original Decision C.P.M. network by an algorithm which is an extension of the longest path algorithm.

5. Decision Graph Solution by Heuristic Techniques

The first integer programming formulation of section four requires a constraint for each link of the graph (except for the links emanating from the start node or a decision preceded only by the start node), one constraint for each multi-job set plus a final due date constraint. The second formulation requires a constraint for each path. In both cases there may be interdependence constraints as well. In very large problems the integer programming solution technique becomes impractical because of the resulting large number of constraints and variables. For that reason we have developed heuristic solution techniques for solving the problem. The heuristic methods to be described have been designed to handle graphs containing only the mutually exclusive type of interdependency. Furthermore, all jobs in a job set S_i are assumed to be bound by the same precedence relationships and jobs in a decision set have the following cost and time relations. (One of the authors is presently developing heuristics for more general cases.)

$$t_{i1} < t_{i2} < \dots < t_{ik(i)}$$

$$c_{i1} > c_{i2} > \dots > c_{ik(i)}$$

The heuristic routine contains the following steps.

(1) Technologically order the jobs.

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

(3) Calculate the critical path.

(4) Reorder by EARLY START (also a technological ordering).

(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 on 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 changed to see if sufficient slack has been generated to allow the reintroduction of a longer but cheaper alternative. HALT.

Some explanation is required for several of the steps. First in (7) it is only necessary to examine decisions on the critical path since for decisions not on the path any reduction in job time would have no effect on the penalty cost. Secondly, the method of calculating cost reduction in (8) is as follows. The additional cost of switching to a more expensive alternative is simply the difference in costs. The saving to be made from reduced overtime penalty on increased premium is not straightforward. The total length of the critical path is not necessarily reduced by the same number of days as the job length is reduced since a parallel chain might become critical if the job is reduced in length by one or more days. It is therefore necessary to check the slack available in all parallel chains and calculate the reduction as the smallest value of slack found.

In the program the maximum "shrink" or minimum "total slack" is calculated as follows. Given the decision node S_i find the slack of all jobs S_{mn} such that EARLY $\text{START}_i \leq$ EARLY $\text{START}_{mn} \leq$ LATE FINISH_i . Given these conditions no job selected could be a sucessor or predecessor of S_i and from all chains not containing S_i at least one job will be selected. Also, given that the length of the project is long compared to the length of the longest job in a multi-job set, and that the jobs are ordered by EARLY START time it is only necessary to examine a small fraction of the jobs in the search for parallel jobs.

This program has been applied to two small problems: (a) the graph of Figure 5 with eight decision nodes and (b) a seven node problem formed by dropping decision node 29 from the graph of Figure 5. Exhibit 2 compares the results of this program with the results of a complete enumeration routine[1/] and the stopped simplex integer programming algorithm. The heuristic technique finds the optimum for the eight node problem in three steps. First with all the minimum cost alternatives chosen the project ends on day 79 (or the artificial FINISH job begins on day 80) and the total cost is $10,298. In the second stage job $S_{29,1}$ is replaced by $S_{29,3}$ at an incremental cost $425 - 200 = $225 but since the project now ends on day 72 the cost is reduced to $7198. Finally $S_{14,1}$ is replaced by $S_{14,2}$ at an incremental cost of $125. The project now ends on day 69 and since the due date is day 70 no penalty or premium is

1/ The complete enumeration program and the results from that program quoted above are a contribution of J. D. Trawick.

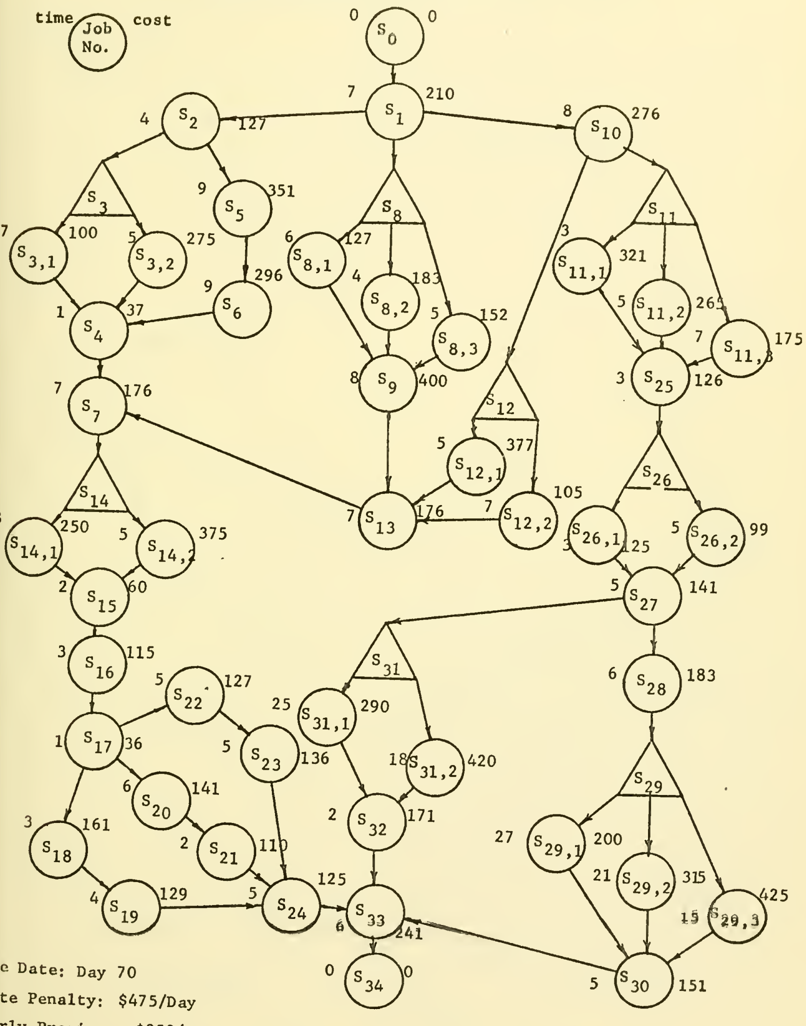
incurred. This optimum solution, with a cost of $5898 is shown in Figure 6. Similarly the routine finds the optimum of the seven node problem with a total time of 69 days and a cost of $5473.

Number of Decision Nodes	Program	Early Start of FINISH	Plans Examined	Computation Time (sec.)
7	Heuristic	70	2	3 1/2
7	Complete Enumeration	70	288	21
7	Integer Programming (Reduced path form)	70	-	4
8	Heuristic	70	3	4 1/2
8	Complete Enumeration	70	864	51
8	Integer Programming (Reduced path form)	70	-	5

Program Performance on Seven and Eight Node Problems - Due Date Day 70

Exhibit 2

DECISION PROJECT GRAPH NO. 2



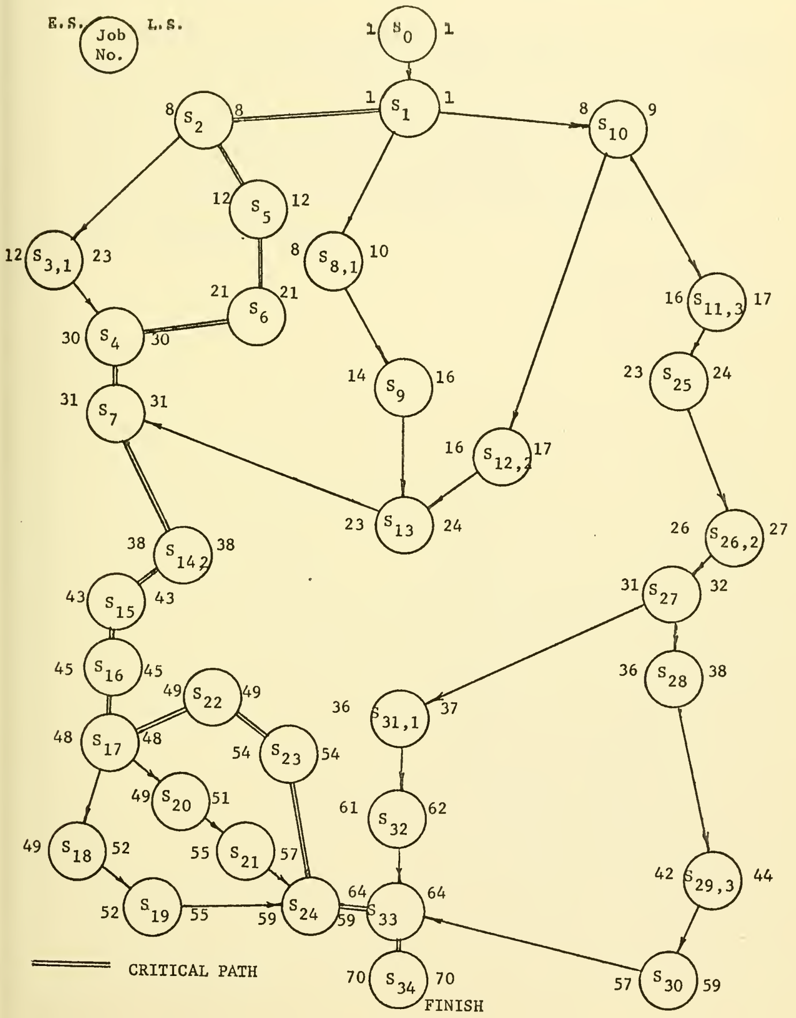
Due Date: Day 70

Late Penalty: $475/Day

Early Premium: $250/Day

Possible Solutions: 864

DECISION PROJECT GRAPH NO. 2 - OPTIMUM SOLUTION



Although the heuristic approach achieved the optimum plan on the two problems tested, we do not wish to imply that it will always do so. At present the program is being modified to improve its efficiency and to allow it to handle other types of interdependencies. It is hoped that with these improvements the program will handle efficiently a wide range of planning problems.

6. Application of Decision CPM to Dynamic Monitoring and Control of Projects

In the previous sections Decision CPM has been presented as a technique to solve combined planning and scheduling problems for construction type projects, as originally stated. However it is clear that such things as project crashing can be handled by the technique if the assumption is made that there are several discrete levels of performance possible, rather than a continuous linear relation between job cost and job time. The assumption of discreteness is particularly fitting in construction projects where there is a choice such as one or two shift operation and the use of regular or quick-drying cement etc.

Furthermore the potential of the technique for project design and job crashing has important implications for the dynamic monitoring control of the project as we carry out the plan. Suppose that daily information is collected concerning the status of jobs in a project being work on. At any given time some jobs will have been finished on time or early and others will have been delayed. Given this information it is possible to solve again for a new optimum in the decision project graph of the remaining problem. It is true that some earlier decisions may be unchangeable, perhaps because of the purchase of material, but some decision sets and many possibilities for crashing of jobs will usually remain. Given the actual status of the project new decisions may now be better than those previously accepted.

An example of such a situation is given in Figure 6 which shows the state of the project graph of Figure 5 at the beginning of day 31. According to the original schedule jobs S_7 and S_{27} should have been started. However $S_{26,2}$ an immediate predecessor of S_{27} with a job time of five days has been in progress only two days.

An examination of the original schedules showed that S_{27} had one day of slack. Therefore the three day delay will lengthen the completion date by two days and incur a penalty of $900. If the graph is solved with the new information a switch from $S_{31,1}$ to $S_{31,2}$ is shown to be desirable. For an incremental job cost of $130 one day can be removed from the completion cost (Figure 7). The total cost of the delay is therefore reduced to 900 - 450 +130 = $580. Thus specifying alternatives as is done in the decision project graph allows continual adjustment of decisions in response to operating experience.

In summary the dynamic design and control processes will operate as follows.

(1) Collect information on

- (a) tasks to be performed and alternative methods of performing them (jobs)
- (b) technological relations between jobs
- (c) interdependences among decisions
- (d) job times
- (e) job costs
- (f) job crashing methods and costs
- (g) project due date, penalty and premium

(2) Solve problem using decision CPM for original plan

(3) Begin project

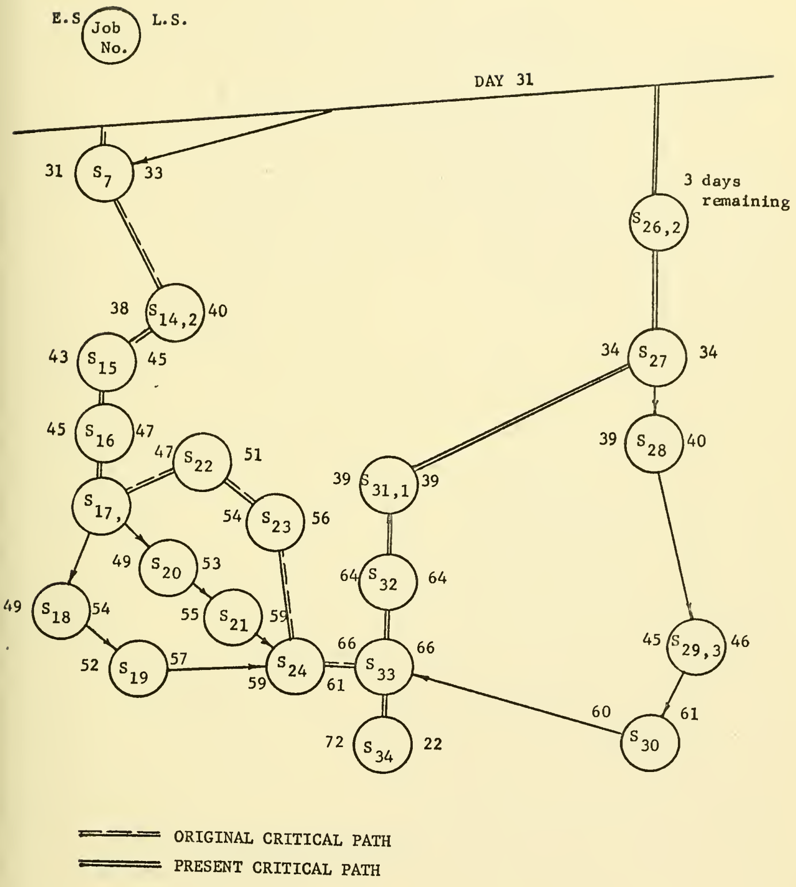
(4) At regular intervals collect information on project progress

(5) Update job cost and job time data

(6) Replace decision nodes by single jobs as decisions become irrevocable over time

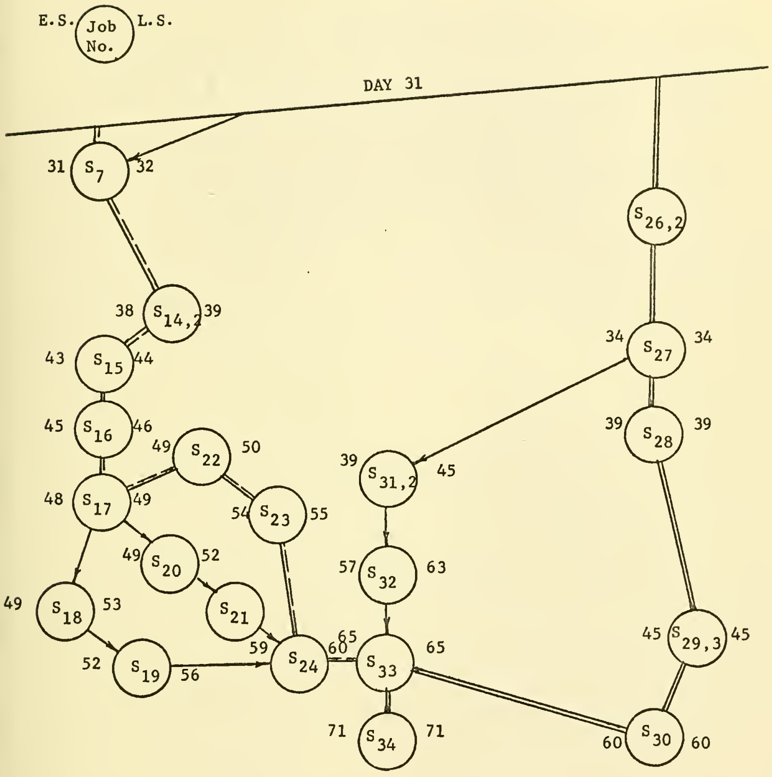
(7) At regular intervals resolve the remaining decision project graph to see if savings are possible by the implementation of a new plan. Go to 4

(8) Halt when project is complete.



E.S
Job No.
L.S.
DAY 31
3 days remaining
31 S_7 33
38 $S_{14,2}$ 40
43 S_{15} 45
45 S_{16} 47
47 S_{22} 51
$S_{17,}$
54 S_{23} 56
49 S_{20} 53
49 S_{18} 54
55 S_{21} 59
52 S_{19} 57
59 S_{24} 61
66 S_{33} 66
72 S_{34} 22
64 S_{32} 64
39 $S_{31,1}$ 39
$S_{26,2}$
34 S_{27} 34
39 S_{28} 40
45 $S_{29,3}$ 46
60 S_{30} 61
ORIGINAL CRITICAL PATH
PRESENT CRITICAL PATH

FIGURE 7



ORIGINAL CRITICAL PATH

FINAL CRITICAL PATH

FIGURE 8

Conclusion

The paper demonstrates the possibility that savings can be made in the total cost of projects if the planning and scheduling problems are solved simultaneously. For small projects the joint problem can be solved by means of integer programming using only information that is available to the planner and the scheduler. For larger projects heuristic techniques show some promise of giving good solutions. It is expected that in practice large savings can also be made from a control system that allows continuing redesign of project plans.

BIBLIOGRAPHY

1. 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, 1962.

2. Charnes, A., W. W. Cooper, J. K. DeVoe and D. B. Learner. "DEMON - Decision Mapping Via GO-NO-GO Networks; A Model for New Products Marketing", Office of Technical Service Report.

3. Charnes, A., W. W. Cooper and G. L. Thompson, "Critical Path Analysis Via Chance Constrained and Stochastic Programming", Operations Research, Vol. 12, No. 3, May-June 1964.

4. Eisner, H., "A Generalized Network Approach to the Planning and Scheduling of a Research Program". Journal of the Operations Research Society, Vol. 10, No. 1, 1962.

5. Elmaghraby, S. F., "An Algebra for the Analysis of Generalized Networks," Management Science, Vol. 10, No. 3, April 1964.

6. Ford, L. K. and D. K. Fulkerson, Flows in Networks, Princeton, N. J. University Press, 1962.

7. Kelley, J E. and M. R. Walker, "Critical Path Planning and Scheduling", Proc. Eastern Joint Computer Conference, 1959.

8. Levy, F. K., G. L. Thompson and J. D. Wiest, "The ABC's of Critical Path Scheduling, Harvard Business Review, Vol. 41, Sept.-Oct., 1963.

9. Levy, F. K., G. L. Thompson and J D. Wiest, "Mathematical Basis of the Critical Path Method", Industrial Scheduling, J. Muth and G. L. Thompson eds. Prentice-Hall, 1963.

10. Malcolm, D. G., J. H. Rosenbloom, G. E. Clark and W. Fazan, "Application of a Technique for Research and Development Program Evaluation", Journal of Operations Research, Vol. 7, No. 5, 1959.

11. Thompson, G. L. "The Stopped Simplex Method: 1. Basic Theory for Mixed Integer Programming; Integer Programming", Revue Francaise De Recherche Operationelle, 1964, pp. 159-182.

12. Weingartner, H. M., Mathematical Programming and the Analysis of the Capital Budgeting Problem, Prentice-Hall, 1963.

87

1886