Division of Research
Graduate School of Business Administration
The University of Michigan

April 1981

An Algorithm for a General Class of Precedence and Resource Constrained Scheduling Problems

Working Paper No. 250

F. Brian Talbot

The University of Michigan

ABSTRACT

A simple yet very general backtracking algorithm capable of optimally or heuristically solving a large variety of precedence- and resource-constrained scheduling problems is presented. Examples of such problems include project scheduling and job shop scheduling under multiple resource constraints, and assembly line balancing. Although many applications of the algorithm are possible, this paper focuses on the solution of nonpreemptive, resource-constrained project scheduling problems in which each job to be scheduled can be characterized by a set of alternative choice modes, only one of which is ultimately selected for scheduling. Two major scheduling objectives are considered: minimizing project duration and maximizing project net present value. In addition, two major classes of resources are considered: renewable or period-related resources, and nonrenewable or project-related resources. The latter category specifically includes cash flow constraints that can subtract from, or add to, cash availability during the life of the project. Solved numerical examples illustrate these concepts.

I. INTRODUCTION

This paper presents a simple yet general backtracking algorithm for heuristically or optimally solving precedence- and resource-constrained scheduling problems. In order to illustrate the logic and mechanics of this procedure, the discussion will focus on the solution of nonpreemptive resource constrained, project-scheduling problems. As indicated previously [13, 15], this problem is the generic form of a class of scheduling problems which includes resource-constrained job shop and assembly line balancing problems. Thus, the algorithm to be introduced can be used, with appropriate modifications, to solve these and related scheduling problems.

Simply stated, the problem addressed is how to schedule precedent-related and resource-constrained jobs in a project in order to accomplish a given managerial objective. Over the past twenty years a number of techniques have been developed to help project managers answer this question, the applicability of each technique being a function of project characteristics and managerial objectives (see, for example, Davis [2] and Elmaghraby [6] for comparisons of the various techniques previously investigated). The current study introduces a scheduling technique that is capable of heuristically or optimally solving most of the nonpreemptive forms of project scheduling problems previously examined in the literature. This includes, but is not restricted to simple time-based, time-cost trade-off, time-resource trade-off, and resource constrained projects. In addition, the proposed algorithm permits the scheduling of jobs where job performance can increase as well as decrease the availability of resources such as cash. The latter situation often occurs in multiproject environments where the cash flow generated by the completion of one project supports the continuation of others. It is also observed within single projects where performance payments are based upon the satisfactory completion of key activities. These payments in turn facilitate the completion of other activities in the project.

The specific form of the scheduling problem examined here first appeared in Elmaghraby [6, p. 173] as a cost-minimization model with resource-duration interactions, and has more recently been investigated in greater detail by Talbot [14]. In this model each job can be performed in one of several ways called operating modes, or simply modes. Each mode represents a different way of combining resources to accomplish a given job. This approach is analogous to, but much richer than, the job definition used in time-cost trade-off models. In the latter it is assumed that job duration can be affected only by expenditure levels; that is, by varying _one_ resource: money. In the proposed model, a variety of multiple resource-duration interactions, such as using different technologies or types of labor to accomplish the same job, can be expressed and evaluated.

Following the scheme suggested by Slowinski [11, 12] and Weglarz [16], resources are assumed to be renewable, nonrenewable, or doubly constrained. _Renewable_ resources are used and constrained on a period-by-period basis. For example, skilled labor would be considered a renewable resource if it is used each day of a project and if it is also available in limited quantities each day. This is the category of resource which has been most frequently modeled by researchers in the past (see, for example [1], [3], [4], [8], [9], [10], [13], [15], and [17]). _Nonrenewable_ resources are available and consumed on a total-project basis. For example, money may be considered a nonrenewable resource if only X dollars can be spent to support all the activities in a project. _Doubly constrained_ resources are simultaneously constrained on a period and project basis. Money would be doubly constrained if both per-period cash flow and total project expenditures were restricted. These three categories permit the evaluation of a variety of common resource restrictions. However, this paper proposes a broader interpretation of nonrenewable resources, which specifically permits the decrease or increase of these resources as a function of activity status. This will allow, for example, the modeling of progress payments which can increase cash flow for use by other activities or projects. In addition, this notion of cash inflows and outflows can better capture the importance of timing of large cash transactions in projects.

Given the problem of selecting and scheduling job modes in a project under the various resource conditions posed above, two scheduling objectives will be considered in this paper: minimizing project completion time and maximizing project net present value. In the following section these problems will be defined formally as zero-one integer programs. Section III of the paper introduces a backtracking algorithm as a heuristic and optimal procedure for solving the project completion time minimization problem. In Section IV, modifications needed by the algorithm to solve the net present value criterion problem are presented. Solved numerical examples illustrating both of these problems are given in Section V, with computational considerations discussed in Section VI. Summary comments are found in Section VII.

II. FORMULATION OF THE PROJECT SCHEDULING PROBLEMS

It is assumed that the project can be depicted as an acyclic network such as that shown in Figure 1. Activities or jobs are represented by integer-labeled nodes, such that the label of a node is always higher than the labels of all its immediate predecessor nodes. Arrows represent precedence relationships between jobs. Unique starting and ending dummy jobs with zero duration are appended to the network. Without loss of generality, all model variables and parameters are assumed to be integer-valued. In some cases, this involves scaling parameters by appropriate powers of ten.

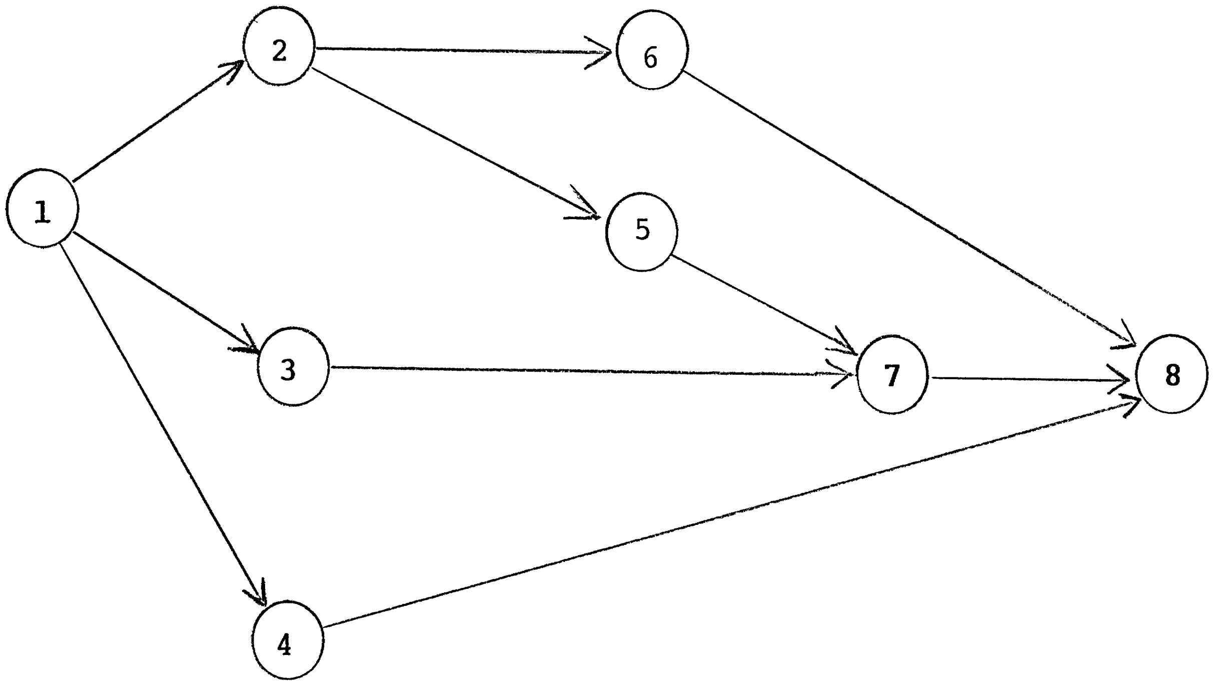


Figure 1
Precedence Diagram of An Eight-Job Project

Each job j may be accomplished in one of M_j modes. These modes are non-preemptable once started. The duration of mode m of job j is D_{jm} and the amount of renewable resource k it requires each period it is in process is r_{jmk}. In order to keep the notation simple, only one renewable resource, cash, will be considered. Mode m of job j consumes or generates, depending upon algebraic sign, c_{jmd} dollars during the d-th period this mode is in process. (If other nonrenewable resources were included, a fourth subscript to c_{jmd} would be required to indicate the type of nonrenewable resource.) Also associated with each job mode is a single non-negative cash payment c^*_{jmv} which accrues v periods after the completion of mode m job j. This parameter can be used to represent delayed receipt of performance payments, while the c_{jmd} can be viewed as period cash flows. The decision variables are x_{jmt} where x_{jmt} = 1 if mode m has been selected to accomplish job j and it is assigned a completion time t. Otherwise, x_{jmt} = 0. Definitions of other variables and parameters used in the discussion are given in Table 1.

The zero-one integer program (1)-(5) depicts the project scheduling problem described above when the objective is to minimize project completion time.

$$\text{Minimize} \quad \sum_{t = E_J}^{L_J} t x_{J1t} \tag{1}$$

$$\text{Subject to} \quad \sum_{m=1}^{M_j} \sum_{t=E_j}^{L_j} x_{jmt} = 1 \tag{2}$$

$$\text{for } j = 1,\ldots,J$$

Table 1

Definition of Terms

Symbol (Listed Alphabetically)	Definition
B_i	the i-th job number in the ordered set of job numbers representing the current best solution.
B_i^*	the completion time assignment of the i-th job in B_i.
c_{jmd}	the cash flow of job j mode m in its d-th period in process ($d = 1, \ldots, D_{jm}$). If $c_{jmd} < 0$ there is a cash withdrawal. If $c_{jmd} > 0$ there is a cash inflow.
c_{jmv}^*	a single non-negative cash flow v periods after the completion of job j mode m ($v \geq 1$).
C_t	a non-negative integer variable indicating the net cash position in period t. C_0 is the cash available at the beginning of the project.
D_{jm}	the duration of mode m of job j: $D_{jm} > 0$, except $D_{J1} = 0$, and $D_{11} = 0$.
E_j (L_j)	initially calculated as the earliest (latest) critical-path-based completion time for any mode of job j.
J	the identifying number of the unique dummy terminal job in the project without successors. Job J has one mode with zero duration and it consumes no resources. However, a positive delayed cash flow may occur after it is assigned.
M_j	the number of modes associated with job j ($m = 1, \ldots, M_j$).
$n(.)$	the number of elements in a set.
N_j^* and N_j	N_j^* is the number of immediate descendants of job j currently in Y_j. N_j is the position index in Y_j of the descendant of j that is to be evaluated for assignment next.

Table 1-cont'd

Symbol	Definition
P_j (S_j)	the set of all immediate predecessor (successor) jobs of job j.
$P(S)$	the set of all pairs of immediate predecessor (successor) jobs. $(a,b)\varepsilon P$ indicates that job a is an immediate predecessor of job b.
R_{kt}	the amount of renewable resource k currently available in period t ($R_{kt} \geq 0$).
r_{jmk}	the amount of renewable resource k required by mode m of job j each period m is in process ($r_{jmk} \geq 0$).
T^*	an arbitrary due date for the project.
w_t	a single-payment, present-value discount multiplier for period t at interest I. $w_t = \left(\frac{1}{1+I}\right)^{t-1}$
x_{jmt}	a zero-one variable which equals zero unless mode m of job j is assigned a completion time in period t. Then, $x_{jmt} = 1$.
Y_j	the set of all immediate descendants of job j in the precedence tree for the current partial solution. Y_{jg} is the job number of the g-th immediate descendant of job j contained in set Y_j for the current partial solution.
Z_i	the i-th job number in the ordered set of job numbers for jobs that have a feasible assignment in the current partial solution ($i = 1,\ldots,J$).
Z_i^*	the completion time of the i-th job in Z_i.

$$-\sum_{m=1}^{M_a} \sum_{t=E_a}^{L_a} t x_{amt} + \sum_{m=1}^{M_b} \sum_{t=E_b}^{L_b} (t-D_{bm})\, x_{bmt} \geq 0 \qquad (3)$$

for all $(a,b) \in P$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} \sum_{q=t}^{t+D_{jm}-1} r_{jmk}\, x_{jmq} \leq R_{kt} \qquad (4)$$

for $k = 1,\ldots,K$

$t = 1,\ldots,T$

$$C_{t-1} + \sum_{j=1}^{J} \sum_{m=1}^{M_j} \sum_{q=t}^{t+D_{jm}-1} c_{jm(D_{jm} + t-q)}\, x_{jmq} + c^*_{jmv} x_{jm(t-v)} = C_t \qquad (5)$$

for $t=1,\ldots,T$

This objective (1) is achieved by scheduling the unique terminal job J as early as possible. Constraint set (2) insures that exactly one mode of each job is selected and scheduled. Precedence relationships are maintained by (3) and renewable resource restrictions are imposed by (4). Constraints in (5) are balance equations for the nonrenewable resource cash. Basically they insure that a job mode is scheduled only if enough cash is available for its use each period it is in process. Beginning with a cash position C_0, cash inflows would typically arise from performance payments c^*_{jmv} following the completion of key activities. As written, (5) precludes the use of funds from external sources once the project is underway. However, (5) could be modified and various other constraints could be added to depict the impact of external funds transfers. Finally, doubly constrained resources would simply be those resources included in both (4) and (5).

When large cash flows occur over a lengthy time horizon, the scheduling objective of maximizing the net present value of the project or projects is often more consistent with long-term organizational goals than is the objective of minimizing project duration. This objective, which has been considered in a non-resource-constrained model by Doersch and Patterson [5], can be stated by substituting (6) for (1).

$$\text{Maximize} \quad \sum_{t=1}^{T} (C_t - C_{t-1}) w_t + C_0 \tag{6}$$

If a performance payment is given upon the completion of the projects then (2)-(6) is the appropriate formulation. Alternatively, (6) could be modified to include lateness penalties as in [5], or per-period project overhead costs as in [14]. Without performance payments, lateness penalties, or period costs, it is possible that jobs with only cash outflows will be delayed indefinitely in an effort to maximize net present value. This problem can be overcome by appending a constraint such as (7), which forces the completion of all jobs before a given project due date T*.

$$\sum_{t=E_J}^{L_J} t x_{J1t} \leq T^* \tag{7}$$

III A BACKTRACKING ALGORITHM FOR THE PROJECT COMPLETION MINIMIZATION PROBLEM

Even for modestly sized scheduling problems, the formulations stated above translate into very large integer problems. Given the limited capabilities of current general purpose integer programming codes, a special purpose algorithm has been developed to solve (1)-(5) and (2)-(7) in a more efficient manner than would otherwise be possible. In developing the algorithm the major goals were to create a procedure that (1) could be implemented on fairly small core computer systems; (2) would provide always-feasible solutions (this guarantees heuristic results if premature algorithm termination is desired before optimality is assured); (3) would be relatively fast; (4) would be fairly simple to code; and (5) would permit the consideration of a variety of scheduling objectives and constraint types. These criteria were selected because, to the extent to which they can be attained, they largely determine the usefulness of such a scheduling tool.

The proposed algorithm is a branch-and-bound procedure of the backtracking variety. This approach was selected because it appeared most promising in meeting the above five goals. It identifies solutions by systematically considering job modes for precedent- and resource-feasible completion time assignments. Optimality is assured by implicitly or explicitly evaluating all possible solutions.

The dominant feature of this procedure is an efficiently generated "precedence tree" that guides the search for solutions. In order to explain the concept and mechanics of this tree, the discussion is initially restricted to the problem described by (1)-(4). The single mode version of (1)-(4), incidently, is the problem examined in detail first by Davis [1], and subsequently by most researchers investigating resource-constrained project scheduling (see, for example, [3], [4], [8], [9], [10], [13] and [15]).

The Precedence Tree

The concept of a precedence tree can be understood by examining a feasible (optimal or heuristic) solution to (1)-(4). The solution consists of two

J-element vectors. The first vector identifies the job modes selected for scheduling and the second indicates the start time for each mode selected. (In this section we will use start times rather than finish times because it simplifies the presentation. Clearly, either is valid.) These two vectors uniquely specify a solution and permit the calculation of all other solution characteristics, such as resources consumed, cash flows, and so on. The solution space thus consists of all precedent- and resource-feasible pairs of solution vectors. To find the optimal solution in this space, a branch-and-bound procedure would implicitly or explicitly evaluate all these vector pairs in a systematic way.

The proposed algorithm accomplishes this search of the solution space indirectly via a mapping of the starting time vector. This mapping substitutes for the starting time vector a rank-ordered set of job numbers of the starting time vector. The procedure then finds the optimal solution by implicitly or explicitly evaluating all possibilities in the solution space which is comprised of the mode selection vector and the vector of ordered job numbers. The benefit of this approach is that the mapping generates a "precedence tree" of sequences of job numbers that can be used to structure the search through the solution space.

By considering the hypothetical eight-job project depicted by the precedence diagram in Figure 1, characteristics of this mapping may be examined in more detail. Suppose the optimal solution to this problem has the starting time vector given in Figure 2a--that is, job 1 starts in period 1, job 2 starts in period 5, and so on. (The mode solution vector is not shown since it is irrelevant to the discussion.) In Figure 2c the elements in this vector have been ranked in nondecreasing order and the job numbers in 2d have been correspondingly changed. Assuming for the moment that the mapping of 2a to 2d

Starting Time Vector	(1, 5, 8, 3, 10, 7, 11, 12)	a
Corresponding Job Vector	(1, 2, 3, 4, 5, 6, 7, 8)	b
Starting Time Vector Rank Ordered	(1, 3, 5, 7, 8, 10, 11, 12)	c
Corresponding Ordered Job Vector	(1, 4, 2, 6, 3, 5, 7, 8)	d
Starting Time Vector Rank Ordered	(1, 3, 5, 7 9, 10, 11, 12)	e
Corresponding Ordered Job Vector	(1, 4, 2, 6, 3, 5, 7, 8)	f

Figure 2

Hypothetical Solution Vectors
to Illustrate Mapping

is homomorphic, it is now possible to see that by evaluating all precedent- and resource-feasible ordered vectors of job numbers such as those in 2d, then the entire solution space of starting time vectors is considered. In particular, an algorithm that sequentially generates all feasible ordered job vectors is in effect generating all starting time vectors.

This new solution space (ignoring the mode selection vector) can be drawn as a precedence tree, as shown in Figure 3 for the eight-job project. The numbers in the tree are job numbers which, when read from top to bottom along connecting lines, give the elements of the ordered vector of job numbers. For example, the solution in Figure 2d is given by the fifth sequence from the right. When this sequence is generated by bringing jobs 1, 4, 2, 6, etc., into solution <u>in order</u>, then in effect the starting time vector 2a is found.

This analysis is clearly based on the assumption that the mapping is homomorphic--which it is in certain cases. For the problem given by (1)-(4), the mapping is homomorphic if the optimal solution desired is where all modes selected are scheduled as early as possible, given precedence and resource constraints. To avoid difficulties posed by the lack of the homomorphic property, this "early start" solution is the one directly found by the proposed algorithm. Other optimal solutions to this minimization problem could be found by delaying noncritical jobs within their slack times, as defined by the optimal project completion time.

To illustrate why the mapping is not homomorphic in general, refer again to Figure 2. Vectors 2d and 2f are identical and yield identical project completion times. However, the rank ordered starting times from which they derive are different. Specifically, job 3 has a starting time of eight in vector 2c and nine in 2e. The significance of this is that for problems such as (1)-(5) or (2)-(7) where the early start job assignment may be suboptimal,

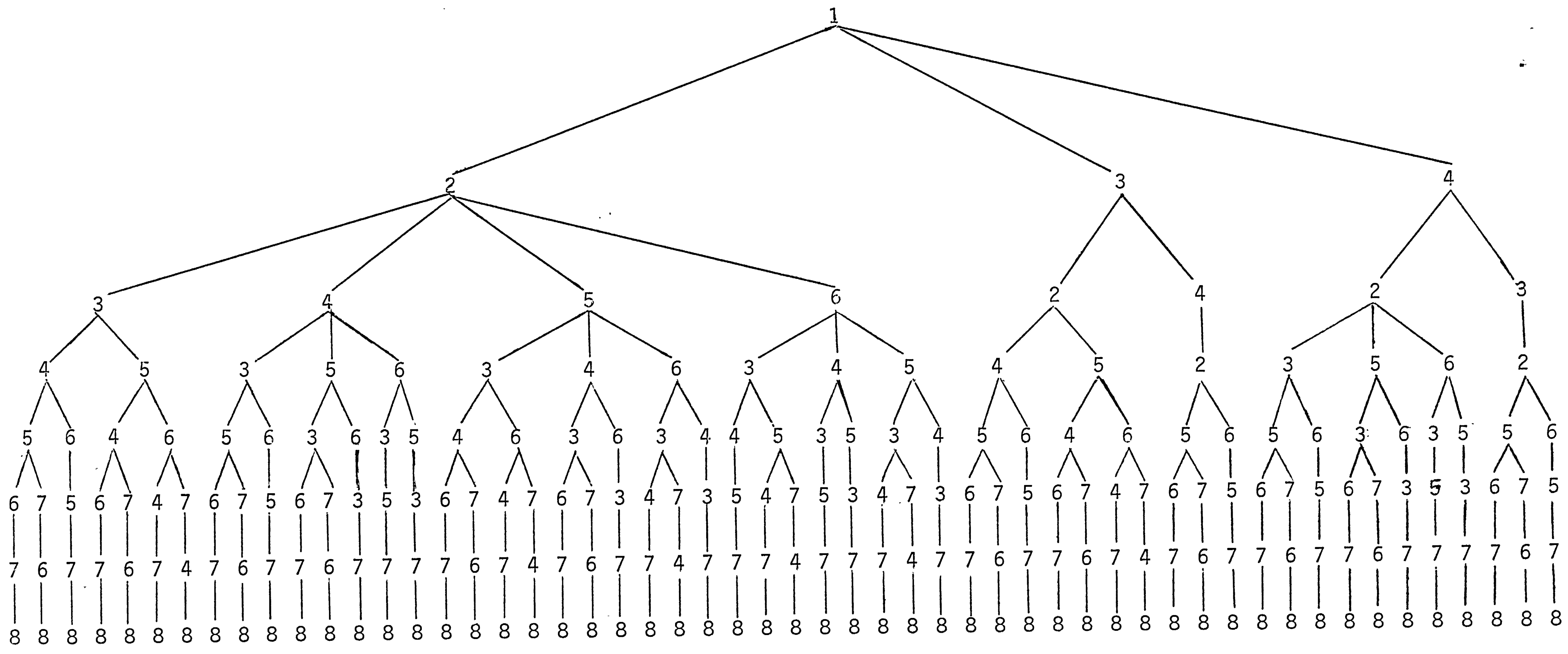


Figure 3

Precedence Tree for the Hypothetical Eight-Job Project knowledge of the ordered set of job numbers is not sufficient information to construct the solution. In this case, the starting time vector is again needed to completely describe the solution. However, the mechanical use of the precedence tree as a means for directing the search through the solution space is still valid. It becomes necessary only to examine each possible starting time for each job <u>within</u> each ordered sequence of job numbers and to record starting times for feasible mode assignments.

This is basically how the proposed algorithm works: jobs are considered for early start (i.e., finish) assignment in the order specified by the precedence tree. If the ordered set of job numbers generated by the tree is not sufficient to completely describe a solution, then jobs are considered for later assignments that are possible <u>without</u> violating the current structure of the tree. In this manner all possible elements of the solution space are examined.

<u>Algorithm Details</u>

The solution procedure can now be stated. It consists of two phases, a problem conditioning phase and an enumeration phase. In the conditioning phase, jobs are relabeled (renumbered) and are assigned late (L_j) critical-path-based completion times. The purpose of the relabeling scheme is to provide a good heuristic solution to the problem on the first pass through the precedence tree. Relabeling conventions are modifications of priority dispatch scheduling rules as described in [14]. Unlike their use in [14], however, the rules do <u>not</u> fix the order in which jobs are considered for assignment. Late finish times (L_j) for each job are calculated in the usual way critical-path late finish times are determined: a reverse pass through the precedence diagram is made after assuming a latest project completion time value L_J. The only modification to this procedure that is required with multiple job modes is that the shortest duration mode be used for all jobs. The purpose of the late finish time parameters L_j is to provide upper bounds for job assignments during the enumeration phase of the algorithm.

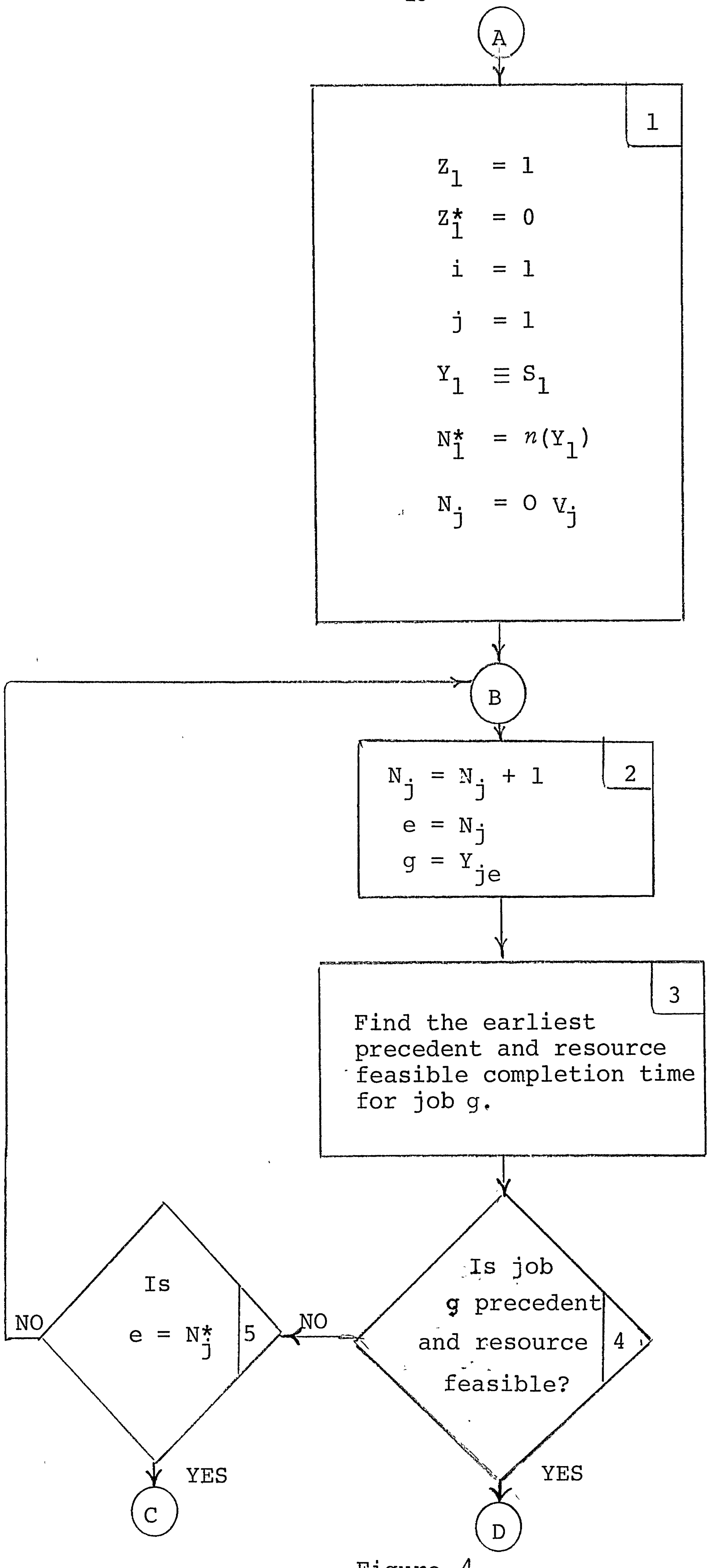
The enumeration phase consists of a procedure for systematically generating the precedence tree, and for implicitly or explicitly evaluating all possible job assignments that could lead to an optimal solution. Figure 4 is an outline of this procedure which is a branch-and-bound algorithm utilizing backtracking as a means for directing the search in the precedence tree once a bound is reached. Backtracking rather than some other form of skiptracking was selected to reduce the bookkeeping overhead required and to significantly reduce the otherwise massive data storage requirements of the procedure.

The algorithm develops a precedence tree, such as the one illustrated in Figure 3, where each numerical node is generated only upon the successful precedent- and resource-feasible assignment of the job with that identifying number. Once job j is assigned to a completion time, a list Y_j of current precedent-feasible jobs (descendants of j) is created. An attempt is then made to assign one of these descendant jobs to its earliest precedent- and resource-feasible completion time. If job j cannot be assigned a completion time within its upper bound L_j, an attempt is made to assign another descendant of the last job successfully assigned. If this also fails, the algorithm backtracks up the precedence tree along the same limb from which it extended to the most recently assigned job. Its next untested descendant is then tested for a feasible completion time, and so on. Once all jobs have been assigned, upper bounds are redefined to correspond to one less than the bounds for the solution found, which forces the algorithm to always seek improved, reduced completion time solutions. Optimality is assured when an attempt is made to backtrack to job zero or when a solution equal to a known lower bound has been found.

The flow diagram in Figure 4 illustrates these details. Initially, however, the discussion will be restricted to the linear program represented by (1)-(4) where each job has one mode. Before step 1 is reached in Figure 4, a number of parameters and arrays are initialized, such as the resource available array R_{kt}, and immediate predecessor P_j and successor S_j arrays. In step 1 the dummy job 1 is assigned a completion time of zero. Its descendant list Y_1 is simply equal to its set of immediate successors. ('$\equiv$' indicates that the set on the left hand side is made equal to the set on the right hand side. For scalers, '=' is used). In step 2, the first descendant of job 1, $g=Y_{11}$, is selected for possible assignment. In general, the search for a resource-feasible assignment for job g is restricted to the interval t' to L_g where

$$t'=\max \{(Z^*_i + 1) \text{ for all i such that } Z_i \in P_g, \text{ and} \qquad (8)$$
$$(Z^*_i - D_{gm} + 1) \text{ for the } \underline{\text{current}} \text{ value of i}\}.$$

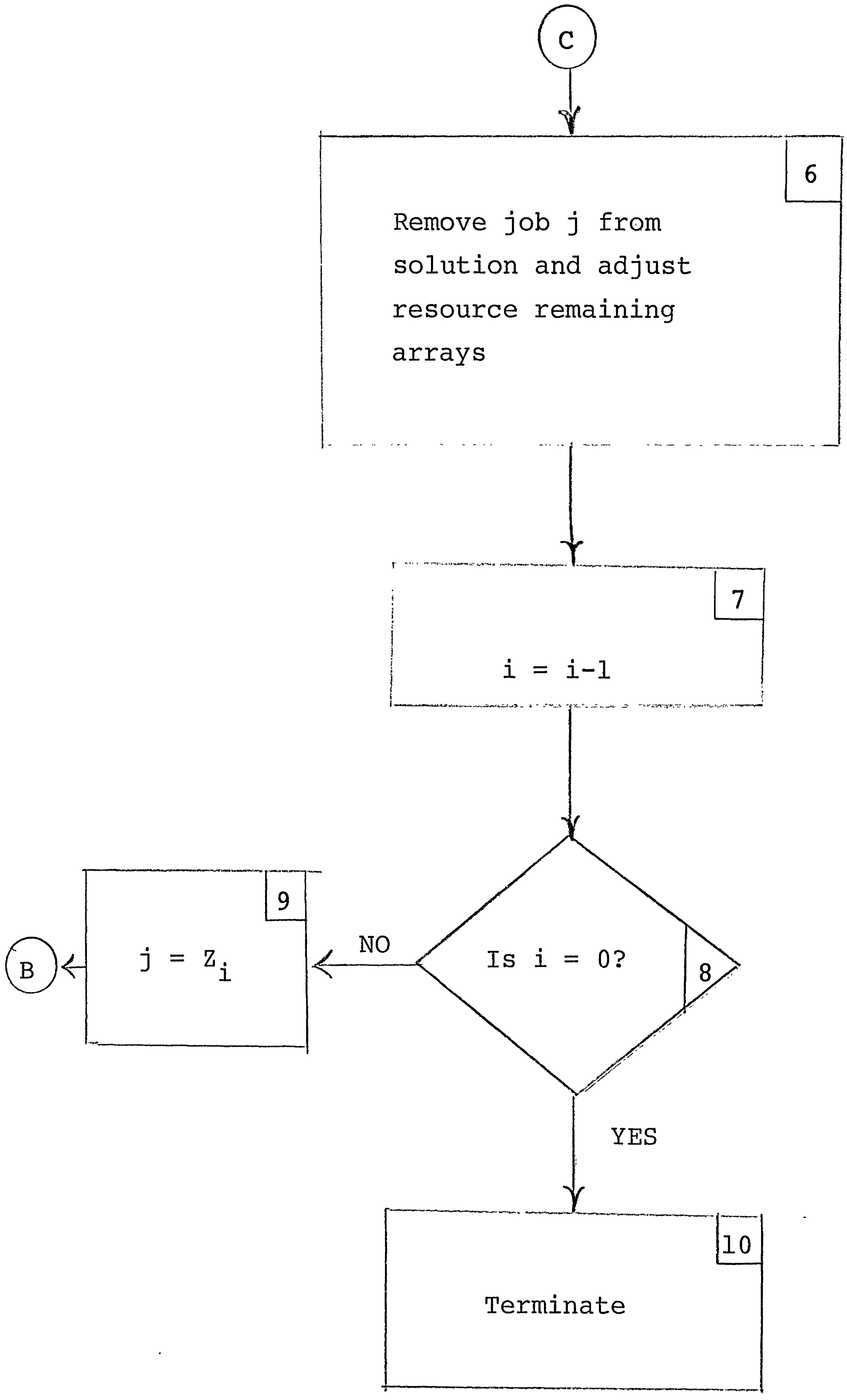
The first part of (8) insures that precedence relationships are not violated. The second invokes the job numbering sequence assumption underlying the use of the precedence tree. That is, the starting time of an as-yet unassigned job cannot be earlier than the starting time of the job most recently assigned. (At any stage of the enumeration i jobs have been assigned.) The job g is thus assigned a feasible completion time when the earliest contiguous interval D_{gm} periods long is identified where $r_{gmk} \leq R_{kt}$ for $t \in (t', L_g)$ and $k=1,\ldots,K$. If this completion time assignment is t*, then in step 11 the resource availability array R_{kt} is reduced by r_{gmk} for all k and $t=t^*-D_{gm}+1,\ldots,t^*$. If a



Assign job 1 a completion time of zero and make its immediate descendant set equal to its set of immediate successors.

Select the next untested job from the descendant set of job j for feasibility testing.

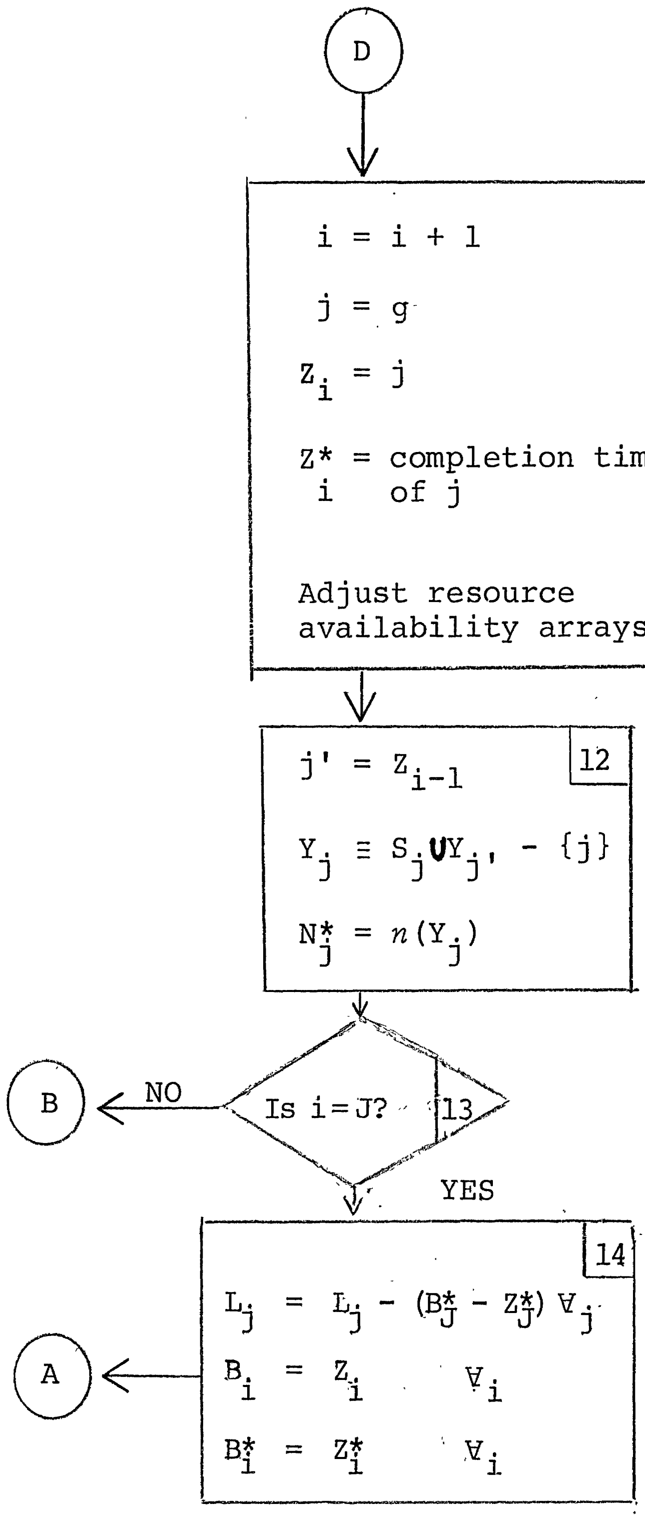
Figure 4
A General Flow Diagram of the Enumeration Phase


C
Remove job j from solution and adjust resource remaining arrays
6
i = i-1
7
Is i = 0?
8
NO
j = Z_i
9
B
YES
Terminate
10

All immediate descendants of j have been evaluated so backtrack to the next untested immediate descendant of j's parent.

Terminate if an attempt is made to backtrack to job zero. The optimal solution is contained in B_i and B_i^* for $i = 1,\ldots,J$.

Figure 4 - continued


D
11
$i = i + 1$
$j = g$
$Z_i = j$
Z^*_i = completion time of j
Adjust resource availability arrays.
12
$j' = Z_{i-1}$
$Y_j \equiv S_j \cup Y_{j'} - \{j\}$
$N^*_j = n(Y_j)$
Is $i = J$?
13
NO
B
YES
14
$L_j = L_j - (B^*_J - Z^*_J) \; \forall_j$
$B_i = Z_i \; \forall_i$
$B^*_i = Z^*_i \; \forall_i$
A

Record the sequence and completion time of job j, and adjust resource arrays to reflect the assignment.

Define immediate descendant set for job j.

If all jobs have been assigned then save the solution and adjust upper bound late finish times before restarting.

Figure 4 - continued

feasible assignment for job g does not exist, then an effort is made via step 5 to assign a different descendant of the last job successfully assigned.

This procedure continues until either all jobs are assigned or it is impossible to assign any immediate descendants of the last job assigned. In the former case, the upper bounds L_j for all jobs are decreased to one less than those corresponding to the current best solution and the enumeration begins again. As indicated in Figure 4, step 14, it is possible to restart with job 1. However, this may needlessly lead to the regeneration of a sizable portion of the precedence tree. An improved restarting procedure, which is incorporated in the algorithm actually programmed, requires only that the restart begin at the lowest indexed job with a current completion time later than its revised L_j. Thus at step 14, rather than transfer to A and job 1, the algorithm would transfer to job $j'=j \varepsilon Z$ for the minimum i, such that $Z_i^* > L$. Continuation to B would then occur after appropriate updating of affected arrays and parameters.

If it is impossible to assign any of the immediate descendants of the last job j assigned, then the algorithm backtracks to the next untested descendant of j's parent. This is illustrated in steps 6-9. If an attempt is made to backtrack to job 0, then all sequences of job numbers that could potentially lead to an improved solution have been either explicitly or implicitly evaluated, so the algorithm terminates. The optimal solution is the last solution found.

Thus far the discussion has been limited to the single-mode case of (1)-(4). The multiple-mode version requires one substantive modification to the logic depicted in Figure 4. During the job selection phase (steps 2-5), if a job mode fails the feasibility test, then the next mode of that job needs to be tested. A transfer to a different job occurs only when a feasible mode is found or when all modes for that job have failed the feasibility test. Of course, during backtracking a transfer is made to the next mode of the last job assigned rather than to the next untested descendant of the last job assigned. Thus the multiple-mode model, although potentially increasing the number of calculations required for solution, has little effect on the amount of bookkeeping required or on the complexity of the algorithm.

The inclusion of (5) greatly increases the generality of the model and slightly increases the bookkeeping overhead. However, it does not affect the basic logic of the algorithm. Additional checks and adjustments of cash flow must simply be made in steps 3, 6, and 11 of Figure 4. Thus, the model given by (1)-(5) is quite similar to that given by equations (1)-(5) in [14]. The major <u>formulation</u> differences are that in [14] nonrenewable resource requirements are associated with an entire job mode, and [14] does not explicitly permit delayed positive cash flows to affect resource availability. The current formulation defines nonrenewable (cash) requirements on a period basis for each job mode and does permit delayed positive cash flows, both of which conditions make the current formulation much more versatile and capable of depicting real-world project conditions.

<u>Algorithmic</u> differences between [14] and the proposed approach are more fundamental. In [14], since job modes are considered for assignment in a predetermined sequence, it is not possible to always maintain cash flow feasibility when positive cash flows exist. Temporary infeasibility would occur, for example, when a smaller-numbered job in the sequence requires the use of cash that would not become available until a larger-numbered job is assigned. The proposed algorithm, by considering jobs for assignment according to the precedence tree, insures that all partial solutions are <u>totally</u> feasible. This is a more natural approach to scheduling jobs in a project environment, and

it can significantly reduce the amount of explicit enumeration required to find either heuristic or optimal solutions.

There is one "pathological case" that could prevent the proposed algorithm from finding the optimal solution. This occurs when two or more job modes must be assigned simultaneously to maintain cash flow feasibility. For example, suppose job mode m has $c_{jm1} = -3$, and $c_{jm2} = +5$, and job j' mode m' has $c_{j'm'1} = +3$, and $c_{j'm'2} = -5$. Further assume that the current cash available C_t in the time interval in which j and j' could potentially be assigned is only 1. Clearly neither j nor j' would pass the feasibility test in step 3 of Figure 4: j would fail in its first time period since it requires 3 units of cash whereas only 1 is available; j' would fail in its second period because it requires 5 units of cash and only 4 would be available. So by <u>sequentially</u> testing job modes for feasibility, the proposed algorithm would incorrectly conclude that a feasible partial solution doesn't exist for a job sequence Z of either (...,j,j',...) or (...,j',j,...), when in fact it does exist. The only algorithm modifications required to deal with this case involve additional tests in step 3 for identifying simultaneous feasibility conditions. These tests have not been included in the programmed procedure. Rather, all problems tested have $c_{jmd} \geq 0$. Under these conditions, the above pathological case does not arise.

IV ALGORITHM MODIFICATIONS FOR THE OBJECTIVE OF MAXIMIZING PROJECT NET PRESENT VALUE

When the minimum completion time scheduling objective (1) is replaced with the maximum net present value objective (6), the problem generally becomes much more difficult to solve. This is due in part to the increased complexity of the cash flow interactions overtime, and in some cases to the absence of strong cash bounds on the solution. If the discount rate is zero and per-period overhead costs are added to (6), then (2)-(6) becomes the maximize profit, limited-resource, cost-time trade-off problem considered previously [14]. This simplified form of the net present value problem can be solved by the proposed procedure without any substantive algorithmic modification. However, when the discount rate is greater than zero, then both the completion time and the numerical sequence of jobs are required to completely define an optimal solution. As indicated earlier, this is due to the fact that positioning of jobs within a given sequence may affect the value of the objective function. Specifically, when the total net cash flow for mode m is negative

(i.e., $\sum_{d=1}^{D_{jm}} c_{jmd} + c^*_{jmv} < 0$), then its net present value increases as

its completion time is delayed. So it is no longer always advantageous to find the earliest finish time assignment for all jobs. Unfortunately, because of cash and other resource trade-offs, neither is it simply a matter of finding the early finish time solution for the project and then delaying those jobs with negative cash flows (although this may yield good heuristic solutions). What is needed is a device that will systematically make these resource trade-offs. This is accomplished in the proposed algorithm by introducing a "right-shift" mechanism.

This mechanism permits the evaluation of delayed mode completion times by introducing modifications in step 6 of Figure 4. Prior to the removal of a mode from solution, an attempt is made to find the next earliest feasible completion time for this mode. If the original schedule were depicted on a Gantt chart, with time increasing from left to right on the horizontal axis, this time delay would appear as a "right-shift" of the mode. If it is possible to find a later completion time for the mode, the algorithm, rather than continuing to step 7, would proceed to step 11. If it is impossible to find a later completion time within the upper bound L_j, the algorithm would continue with step 7. By right-shifting to every later feasible completion time in this fashion, all potential optimal solutions can be identified. However, since many more possible mode assignments must be considered for this objective function, this procedure can significantly increase the computational time of the algorithm.

Method for Reducing Right-Shifting

One way to reduce the computational impact of this approach is to apply right-shifting only to modes with negative total net cash flows. The following argument justifies the omission of right-shifting for modes with zero or positive cash flow. In order for the right-shifting of a mode with zero or positive net cash flow to have a beneficial effect on the objective function, one of three situations must occur: the right-shifting permits (a) the earlier assignment of at least one as-yet unassigned mode with positive net cash flow, or (b) the delay of at least one as-yet unassigned mode with negative cash flow that would not otherwise be delayed, or (c) the selection and scheduling of a different mode for at least one as-yet unscheduled job than was possible without the right-shifting.

The proofs that situations a, b, or c cannot occur rely primarily on three observations. First, right-shifting can only decrease the interval (t', L_j) over which a search for a resource-feasible assignment is conducted for job j, as is indicated by definition (8). Second, renewable resource requirements r_{jmk} are constant each period over the duration of a given mode. Thus, right-shifting cannot improve (increase) nonrenewable resource availability in the search interval. Third, since right-shifting cannot improve the availability of nonrenewable resources, its only potential benefit would be from the increased availability of cash. However, it is assumed that positive cash flows can occur only v periods after job completion in the form of a performance payment. Thus, delaying a job with positive cash flows can only delay the benefit of these cash flows, not their magnitude.

The proof that condition (a) cannot arise follows directly from observations one, two, and three. Condition (b) cannot occur because of the same three observations, plus the fact that all net negative cash flow modes will be right-shifted by the algorithm. The proof that condition (c) cannot occur also follows directly from the three observations. In fact, if any effect were to occur, it would be the elimination of modes from consideration. Thus, given the model assumptions, it is necessary to right-shift only modes with negative net cash flows.

The Relaxation of Model Assumptions

Before looking at a numerical example of a project scheduling problem of the sort discussed here, it is worthwhile to investigate briefly the impact upon the proposed algorithm of various relaxations of the model assumptions. As mentioned previously, it is possible to construct "pathological" problems that can cause premature backtracking because of the failure of the algorithm to detect "simultaneous" feasibility. This happens when the assumption that positive cash flows can only occur v periods after job completion is relaxed. If this assumption is relaxed for the maximize net present value problem, then it would be necessary to incorporate logical checks for simultaneous feasibility testing during the right-shift process as well.

If the assumption that renewable resource usage r_{jmk} is constant over the duration of a mode is relaxed, then right-shifting of <u>all</u> modes is required for either problem. This is due to the fact that mode delays might differentially affect renewable resource availabilities on a period-to-period basis. In effect, delays might open renewable resource "windows" where a particular mode might fit. These windows could be identified only by right-shifting.

The assumption that modes, once started, cannot be preempted would not affect the logic of the algorithm if preemption were limited to integer durations, but it would probably substantially increase the overhead burden and computational effort. If all job modes could be preempted to the unit duration level, then the problem would be analogous to solving a problem identical to the original except that each original mode would be replaced by D_{jm} unit duration modes.

Resource-constrained project-scheduling research to date has assumed that all jobs in a project must be accomplished before the project is completed. However, the multimode job definition used here permits the modeling of "go, no-go" situations as well by specifying that one mode for a job has zero duration. That is, the assumption that $D_{jm} > 0$ is relaxed. This would be useful in capital allocation project selection problems where, because of the magnitude and timing of cash flows, the sequencing of activities in a project significantly affects the attractiveness of the project. Figure 5 illustrates this point further.

In Figure 5 are depicted three projects linked by dummy start and finish jobs. For example, project 1 is comprised of jobs 2-6. It is specified that each job can be accomplished in at least two ways (modes), including one mode with zero duration which represents not doing the job at all. If a project is selected, however, it is assumed that all jobs in that project must be completed. The managerial objective of interest may be to allocate funds to that combination of projects which has the highest net present value.

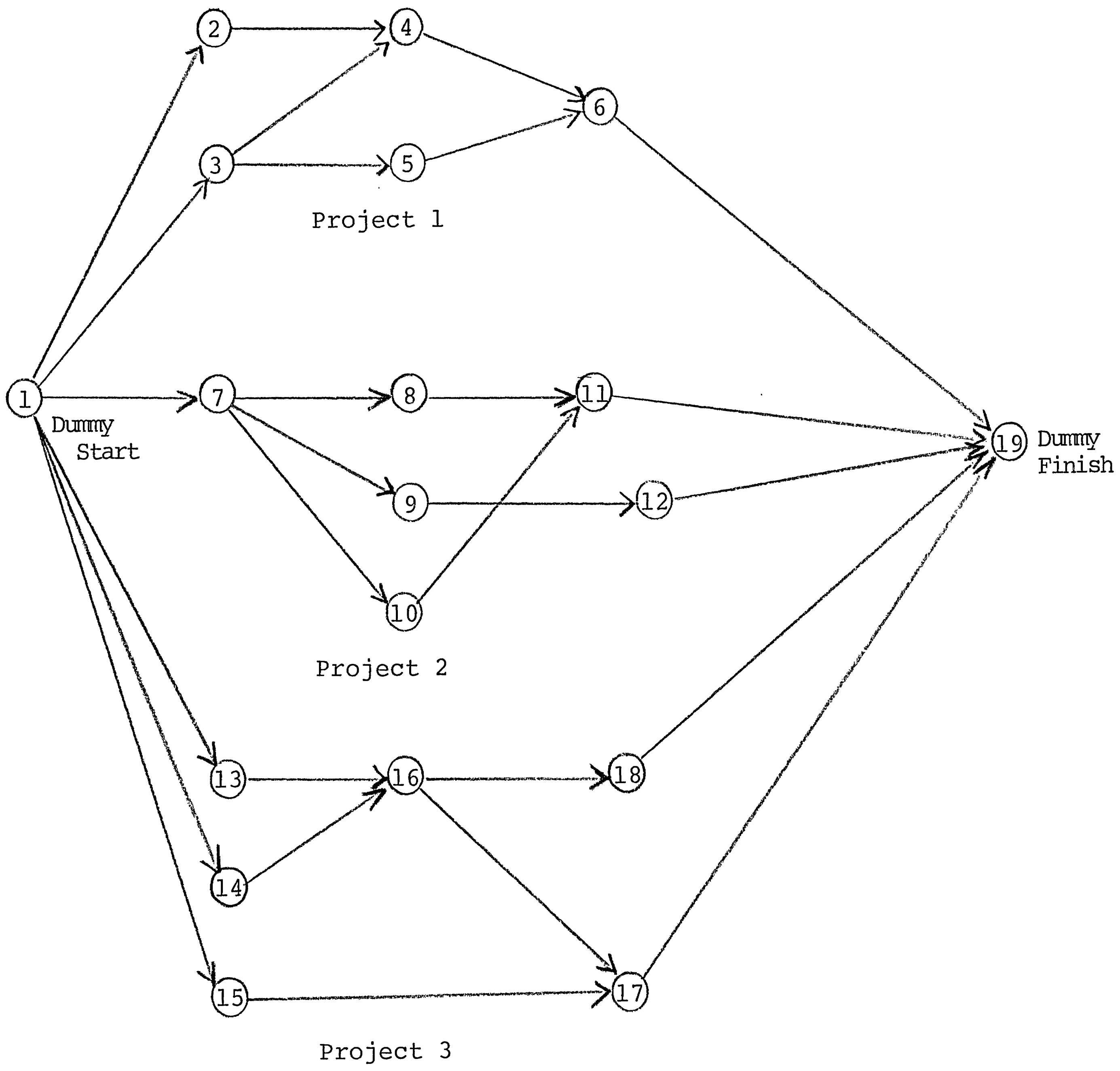


Figure 5
Precedence Diagram of Three Hypothetical Projects

The proposed algorithm could solve this resource-constrained selection-scheduling problem if one minor modification were added to its logic: either all modes in a given project selected for scheduling have zero duration, or all modes in a project selected have durations larger than zero. An answer that selected modes with zero durations for jobs 7 through 12, for example, would mean that net present value is maximized by funding only projects 1 and 3. Of course, in this situation the objective function may be designed to recognize the opportunity cost of capital not spent on these projects. This could be accomplished by "investing" unused capital at a hurdle rate of return. Only that group of projects making more than this hurdle would be selected, with unused funds invested in a bogus alternative.

V. NUMERICAL ILLUSTRATIONS

Minimum Completion Time Objective Function Example

The problem found in Elmaghraby [6, p. 179] has been modified to illustrate the formulations with minimum project completion time and maximum project net present value objective functions. Figure 1 is the precedence diagram for this problem, and Figure 3 is the entire solution precedence tree. Table 2 contains the duration and resource data for each job mode in this project. For example, mode 2 of job 6 has a duration of four and requires the use of three renewable resources: types 1, 3, and 4. One unit of type 1 and two units of types 3 and 4 are required each period the mode is in process. One, two, six, and eight units are available each period for renewable types 1, 2, 3, and 4, respectively. Mode 2 of job 6 consumes three units of cash during each of its four periods of operation, and generates no cash flow at its completion. Associated with job 5 is a performance payment of 32 which is made one period after its successful completion. Hence, $c^*_{5m1} = 32$. It is assumed that the initial cash position C_0 is 200.

Table 2

Data For Hypothetical Project

ob umber	Mode Number	Duration	Per Period Resource Requirements by Type 1	2	3	4	Cash Requirements by Period 1	2	3	4	5	6	7	Delayed Single Period Cash Payment
1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	3	0	1	2	1	5	5	5					0
	2	2	1	0	2	1	10	10						0
3	1	3	0	1	3	2	6	6	6					0
	2	1	1	0	1	2	15							0
4	1	4	0	1	1	4	2	2	2	2				0
	2	3	1	0	1	4	4	4	4					0
5	1	7	0	1	1	3	1	1	1	1	1	1	1	32
	2	5	1	0	1	3	4	4	4	4	4			32
6	1	6	0	1	2	2	1	1	1	1	1	1		0
	2	4	1	0	2	2	3	3	3	3				0
7	1	4	0	1	3	4	2	2	2	2				0
	2	1	1	0	3	4	5							0
8	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Per Period Resource Availability			1	2	6	8								

Figure 6 is a minimum completion time solution to the above problem. The Gantt chart illustrates the modes selected and their scheduled times. The resource profile shows the end cash position each period and the renewable resources required each period of the project. Since the initial cash position of 200 exceeded the total cash requirements for the project, cash was not a limiting resource and had no effect on the schedule. Renewable resources were constraining, however, and delayed the project one period beyond its critical-path completion time of eight periods.

Figure 7 contains a solution illustrating the impact that cash flows can have upon scheduling. For this solution, renewable-resource availabilities were all changed to 10 units per period, the initial cash balance was changed to 22, and the performance payments associated with job 5 were changed to 207 for mode 1 and zero for mode 2. Since 22 cash units were available initially, the only feasible assignment before the arrival of more cash from mode 1 of job 5 was the longer duration and less expensive mode of job 2. Jobs 3, 4, 6, and 7 could not be scheduled prior to the completion of job 5 because of the lack of cash.

Maximize Net Present Value Objective Function Example

Table 3 contains the terminal cash payment data used to solve the sample problem with a net present value objective function. It indicates, for example, that a cash payment of 30 will be made one period after the completion of job 3. The precedence diagram given in Figure 1 and the resource requirements and period cash needs are the same as those given in Table 2.

Figure 8 is the solution to this problem as modeled by equations (2)-(7). The per-period discount factor is 2%, T*=10, and renewable resources available by type are 1, 2, 6, and 8. Without the latter resource restrictions, one

Gantt Chart

Job Number	Mode Number	1	2	3	4	5	6	7	8	9
2	1	X	X	X						
3	1	X	X	X						
4	2	X	X	X						
5	2				X	X	X	X	X	
6	1				X	X	X	X	X	X
7	2									X

Time Period

Resource Profile

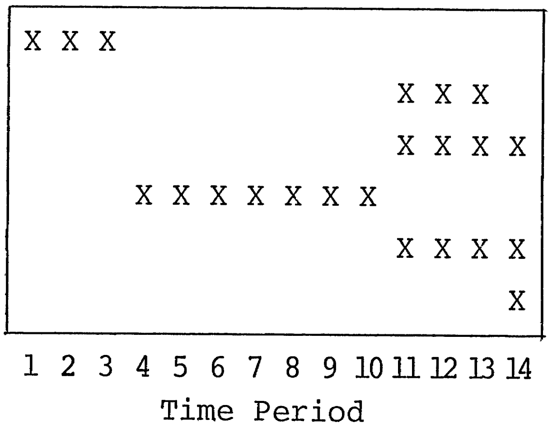
Time Period	End Cash Position	Resources Used by Type			
		1	2	3	4
1	185	1	2	6	7
2	170	1	2	6	7
3	155	1	2	6	7
4	150	1	1	3	5
5	145	1	1	3	5
6	140	1	1	3	5
7	135	1	1	3	5
8	130	1	1	3	5
9	124	1	1	5	6

Figure 6

Solution to the Hypothetical Project with a Completion Time Minimization Objective

Gantt Chart

Job Number	Mode Number
2	1
3	1
4	1
5	1
6	2
7	2



Resource Profile

Time Period	End Cash Position	Resources Used by Type 1	2	3	4
1	17	0	1	2	1
2	12	0	1	2	1
3	7	0	1	2	1
4	6	0	1	1	3
5	5	0	1	1	3
6	4	0	1	1	3
7	3	0	1	1	3
8	2	0	1	1	3
9	1	0	1	1	3
10	0	0	1	1	3
11	196	1	2	6	8
12	185	1	2	6	8
13	174	1	2	6	8
14	164	2	1	6	10

Figure 7

Solution to Hypothetical Project with Completion Time Minimization Objective

Table 3

Additional Data for the Net Present Value Objective Function Example

Job Number	Mode Number	Delayed Single Period Cash Payment
1	1	60
2	1	0
	2	0
3	1	30
	2	30
4	1	0
	2	0
5	1	40
	2	40
6	1	0
	2	0
7	1	0
	2	0
8	1	200

Gantt Chart

Job Number	Mode Number	1	2	3	4	5	6	7	8	9	10
2	2	X	X								
3	2					X					
4	1	X	X	X	X						
5	1			X	X	X	X	X	X	X	
6	1					X	X	X	X	X	X
7	2										X

Time Period

Resource Profile

Time Period	End Cash Position	Net Cash Flow	Present Value	Resources Used By Type 1	2	3	4
0	60	60	60.0	0	0	0	0
1	48	-12	-12.0	1	1	3	5
2	36	-12	-11.8	1	1	3	5
3	33	-3	-2.9	0	2	2	7
4	30	-3	-2.8	0	2	2	7
5	13	-17	-15.7	1	2	6	7
6	41	28	25.4	0	2	3	5
7	39	-2	-1.8	0	2	3	5
8	37	-2	-1.7	0	2	3	5
9	35	-2	-1.7	0	2	3	5
10	69	34	28.5	1	1	5	6
11	269	200	164.1	0	0	0	0
	Total Net Present Value		227.6				

Figure 8

Solution to the Hypothetical Project with a

Maximize Net Present Value Objective

would expect early start assignments for each job with positive net present value (jobs 3, 5, and 8), and late finish assignments for jobs with negative net present value (jobs 2, 4, 6, and 7). Given the resource restrictions, however, job 4 is actually scheduled before job 3, although job 3 is scheduled as early as possible within its remaining slack interval of periods 3 to 9. This is indicative of the complex and often non-obvious trade-offs that result with this model. It also highlights a potential use of the model: sensitivity analysis. For example, a planner might like to measure the impact of various resource allocations upon cash flows, completion times, or the net present value of a project.

VI COMPUTATIONAL CONSIDERATIONS

The algorithm proposed in this paper is capable of optimally or heuristically solving a larger class of nonpreemptive, resource-constrained project scheduling problems than procedures such as [3], [8], [9], [13], and [15]. However, the generality of the algorithm carries with it an increased computational cost which could exceed the benefits derived from modeling these more complex resource relationships. The potential user of such a procedure should be reminded that solving combinatorial problems is still as much a craft as a science, and that many factors, including model complexity, affect algorithmic suitability and performance.

In the process of developing this procedure explicit recognition was given to the fact that the benefit derived from knowing the optimal solution, versus merely a heuristic solution, may simply not be worth the extra computational cost involved in obtaining it. Thus the procedure was designed to generate always-feasible, partial solutions so that premature termination of the computer code would yield at least good heuristic solutions to the problem.

Heuristics also play several other roles in the use of such a tool. For example, it is well known that the form in which a combinatorial problem is posed may significantly affect computational time. Thus, the first stage of the proposed algorithm includes a variety of restructuring heuristics that re-order the jobs and modes of a problem before optimization is attempted, in an effort to find such a good "form." Restructuring rules for jobs, based upon modifications of priority dispatch scheduling rules [14], initially affect the order in which jobs are considered for assignment. Mode-sorting rules, such as smallest duration first, longest duration first, largest net cash flow first, and so on, directly control the order in which job modes are evaluated for assignment. If these rules are judiciously selected in a manner consistent with the scheduling objective, then computation times may be concomitantly reduced. Also, it is possible to start with a detailed schedule for all jobs and let the algorithm attempt to improve upon it. Given the well-defined search procedure based on the precedence tree, it is possible in effect to start anywhere in the tree and continue seeking an improved solution without regenerating solutions that would have been found prior to the heuristic start.

Many refinements other than those mentioned above could be incorporated into the procedure by exploiting the structure of a particular class of problems. This would reduce the computational cost of the algorithm but would also tend to limit its applicability. Improved upper and lower bounds, infeasibility tests, and backtracking methods are examples of some of these logical refinements.

VII. SUMMARY AND CONCLUSIONS

This paper has presented a branch-and-bound backtracking algorithm capable of optimally or heuristically solving a large class of nonpreemptive resource-constrained project scheduling problems. Included in this class of problems are those with per-period resource restrictions and per-project resource constraints. Specifically included in the latter category are cash flow constraints where activities can consume and generate cash flows. Multimode job definitions which permit the evaluation of a variety of resource-duration interactions are modeled. The discussion focused on the solution of project scheduling problems with one of two objectives: minimize project completion time or maximize project net present value. It was also shown how the algorithm could be modified to solve the multiproject selection problem when the scheduling of the projects affects the attractiveness (in particular net present value) of the project. This is the capital-rationing problem, but with the addition of precedence and multiple-resource side constraints.

It has been shown elsewhere (for example, [13] and [15]) that procedures capable of solving the resource-constrained project-scheduling problem can be modified to solve resource-constrained job shop and assembly line balancing problems. By generalizing the precedence tree logic, it is also possible to extend this approach to the resource-constrained decision critical-path problem which, in its unconstrained form (DCPM), has recently been solved by use of dynamic programming [7]. A related problem, amenable to solution by a modified version of the proposed algorithm, is the multimode (not multimodel) assembly line balancing problem. For example, as in project scheduling, it may be possible to accomplish an assembly task in a number of different ways (modes), following generalized precedence relationships as in DCPM. Extensions to the proposed algorithm are currently being investigated to solve these problems.

REFERENCES

[1] Davis, E.W. "An Exact Algorithm for the Multiple Constrained-Resource Project Scheduling Problem." Ph.D dissertation, Yale University, May 1968.

[2] ________. "Project Scheduling under Resource Constraints--Historical Review and Categorization of Procedures." AIIE Transactions 5, no. 4 (December 1973): 297-312.

[3] ________, and Heidorn, G.E. "An Algorithm for Optimal Scheduling under Multiple Resource Constraints." Management Science 17, no. 12 (August 1971): B-803-816.

[4] Davis, E.W., and Patterson, J.H. "A Comparison of Heuristic and Optimal Solutions in Resource-Constrained Project Scheduling." Management Science 21, no. 8 (April 1975): 944-955.

[5] Doersch, R.H., and Patterson, J.H. "Scheduling a Project to Maximize Its Present Value: 0-1 Programming." Management Science 23, no. 8 (April 1977): 882-889.

[6] Elmaghraby, Salah E. Activity Networks: Project Planning and Control by Network Models. New York: John Wiley & Sons, 1977.

[7] Hindelang, T.J., and Muth, J.F. "A Dynamic Programming Algorithm for Decision CPM Networks." Operations Research 27, no. 2 (March-April 1979): 225-241.

[8] Patterson, J.H., and Huber, D. "A Horizon-Varying, Zero-One Approach to Project Scheduling." Management Science 20, no. 6 (February 1974): 990-998.

[9] ____________, and Roth, G.W. "Scheduling a Project under Multiple Resource Constraints: A Zero-One Programming Approach." AIIE Transactions 8, no. 4 (December 1976): 449-455.

[10] Pritsker, A.B.; Watters, L.J.; and Wolfe, P.M. "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach." Management Science 16, no. 1 (September 1969): 93-108.

[11] Slowinski, Roman. "Two Approaches to Problems of Resource Allocation among Project Activities--A Comparative Study." Journal of the Operational Research Society 31, no. 8 (August 1980): 711-723.

[12] Slowinski, R. "Multiobjective Network Scheduling with Efficient Use of Renewable and Non-renewable Resources." European Journal of Operational Research, in press.

[13] Stinson, J.B.; Davis, E.W.; and Khumawala, B.M. "Multiple Resource-Constrained Scheduling Using Branch and Bound." AIIE Transactions 10, no. 3 (September 1978): 252-259.

[14] Talbot, F.B. "Project Scheduling with Resource-Duration Interactions: The Nonpreemptive Case." Working Paper No. 200, Graduate School of Business Administration, The University of Michigan, January 1980.

[15] ___________, and Patterson, J.H. "An Efficient Integer Programming Algorithm with Network Cuts for Solving Resource-Constrained Scheduling Problems." Management Science 24, no. 11 (July 1978): 1163-1174.

[16] Weglarz, J. "On Certain Models of Resource Allocation Problems." Kybernetes, 9, no. 1 (January 1980): 61-66.

[17] ___________; Blazewicz, J.; Cellary, W.; and Slowinski, R. "An Automatic Revised Simplex Method for Constrained Resource Network Scheduling." ACM Transactions on Mathematical Software 3, no. 3 (September 1977): 295-300.