Management Science

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem

James H. Patterson,

To cite this article:
James H. Patterson, (1984) A Comparison of Exact Approaches for Solving the Multiple Constrained Resource, Project Scheduling Problem. Management Science 30(7):854-867. http://dx.doi.org/10.1287/mnsc.30.7.854

MANAGEMENT SCIENCE
Vol. 30, No. 7, July 1984
Printed in U.S.A.

A COMPARISON OF EXACT APPROACHES FOR SOLVING THE MULTIPLE CONSTRAINED RESOURCE, PROJECT SCHEDULING PROBLEM*

JAMES H. PATTERSON
University of Missouri–Columbia, Columbia, Missouri 65201
and
Nijenrode, The Netherlands School of Business, Straatweg 25, 3621 *BG Breukelen, The Netherlands*

A recurring problem in managing project activity involves the allocation of scarce resources to the individual activities comprising the project. Resource conflict resolution decisions must be made whenever the concurrent demand for resources by the competing activities of a project exceeds resource availability. When these resource conflict resolution decisions arise, project managers seek direction on which activities to schedule and which to delay in order that the resulting increase in project duration is the minimum that can be achieved with the given resource availabilities. The procedures examined in this paper are all designed to provide for this type of decision support. Each procedure examined is enumerative based, methodically searching the set of possible solutions in such a way that not all possibilities need be considered individually. The methods differ in the manner in which the tree representing partial schedules is generated and is saved, and differ in the methods which are used to identify and discard inferior partial schedules. Each procedure was found to be generally superior on a specific class of problems, and these classes are identified.

(PROJECT MANAGEMENT–RESOURCE CONSTRAINTS; PROGRAMMING–INTEGER ALGORITHM, BRANCH AND BOUND; NETWORKS/GRAPHS–APPLICATIONS)

1. Introduction

Subsequent to the introduction of the network scheduling techniques of PERT and CPM, efforts commenced to expand upon these techniques to include (as well as other enhancements) resource allocation. These efforts commenced, because while the initial applications of these two techniques were eminently successful—PERT in scheduling the development and the manufacture of the Fleet Ballistic Missile (FBM) weapon system for the Polaris submarine (Malcolm et al. 1959) and CPM in scheduling the construction and the maintenance of a chemical plant facility (Kelley and Walker 1959)—the assumptions under which the techniques were used often changed. For example, resources are often available per period in limited amounts, and while the PERT/CPM solution dictates a precedence based or a technological based start time for the activities of a project, it is often found that resources are not available in sufficient quantities to schedule all of the activities when the original versions of PERT and CPM indicate the activities should be scheduled. Under conditions of limited availabilities of resources, project planners seek direction on which activities to schedule and which to delay in order to minimize the resulting increase in project duration. Davis' classic papers (1966, 1973) survey the development of the constrained-resource, project scheduling problem, as well as the resource leveling problem and the time/cost tradeoff problem in project network analysis.

Early attempts to resolve the resource-constrained version of this scheduling problem concentrated in two areas: the formulation and solution of the problem as a mathematical (usually integer) programming problem, and the development of heuris-

*Accepted by David G. Dannenbring; received May 25, 1983.

0025-1909/84/3007/0854$01.25

tic or approximate solution procedures for obtaining good or satisfying solutions to the problem. Because the early attempts at using integer programming to solve the exact version of this problem were unsuccessful (Brand et al. 1964, Davis 1973) numerous specialized (usually enumerative) approaches for solving certain variants of this problem optimally (Balas 1970, Davis 1969, Davis and Heidorn 1971, Fisher 1973, Gorenstein 1972, Johnson 1967, Patterson and Huber 1974, Patterson and Roth 1976, Schrage 1970, Stinson 1976, 1978, Talbot 1976) were developed. To date, however, there have appeared no comparative studies to indicate which of these exact procedures is to be preferred, especially given a specific problem or a specific problem type to be solved. This paper presents such a comparison of exact procedures for solving the resource-constrained, project scheduling problem. A comparison of heuristic procedures for the same problem treated in this paper can be found in Davis and Patterson (1975).

The specific problem addressed in this paper is the multiple constrained-resource, single project scheduling problem, in which it is assumed that an activity cannot be interrupted once begun (i.e., the nonjob splitting version of this problem). Further, resources are assumed to be available per period in constant amounts, and are also demanded by an activity in constant amounts throughout the duration of the activity. (Although in §6, we do comment upon expanding these techniques to treat variations of the stated problem type, such as varying the amount of each resource available per period over the duration of a project.) Each of the approaches evaluated represents the state of the art in its respective area for solving this resource-constrained problem.

In the next section, we briefly describe each of the solution procedures investigated. In §3, we indicate why other approaches for solving this (or variants of this) problem were not included (usually because each of the latter approaches was dominated in some way by an approach which was included). §4 describes 110 project scheduling problems which served as the test vehicle for evaluating each of the solution procedures examined, and §5 presents summary operating characteristics and computational results using each of the approaches. In §6 we comment upon expanding or modifying each of the solution procedures for solving variants of the project scheduling problem not treated in this paper. §7 then summarizes the results of this investigation.

2. Description of Techniques Investigated

In this section, we describe the three solution procedures included in our investigation. The first procedure described is based upon a *bounded enumeration* of activity completion times, the second procedure described is based upon a *branch and bound* solution approach, and the third procedure is based upon a systematic evaluation (*implicit enumeration*) of activity completion times. Each procedure investigated begins the optimization process by solving only part of the scheduling problem through relaxing certain of the constraints or ignoring temporarily certain imposed restrictions. As fewer and fewer of the restrictions are ignored, a tree of partial solutions (schedules) is generated. The methods differ in the way in which the tree of partial solutions is generated (i.e. the order in which candidate problems or candidate schedules are considered), and differ in the manner in which inferior schedules or solutions are recognized and are discarded. We will thus distinguish among the three solution procedures on the basis of (1) the composition of a partial schedule or partial solution, (2) the order in which candidate problems or partial schedules are considered (which in turn influences the way in which the tree of partial solutions is generated), and (3) the methods used by each of the solution procedures to recognize and remove inferior partial schedules from consideration. Examples of the tree representing partial and complete solutions (schedules) can be found in Davis and Heidorn (1971) and in Stinson et al. (1978).

Description of the Davis and Heidorn Bounded Enumeration Procedure

The procedure developed by Edward W. Davis (1969) and programmed for computation by George E. Heidorn (1971) (which the authors refer to as *bounded enumeration*) uses techniques originally developed for solving the assembly line balancing problem (Gutjahr 1963, Gutjahr and Nemhauser 1964) to solve the multiple-constrained resource, project scheduling problem. Their procedure initially divides each of the original activities of the project into a series of unit duration tasks, the number of unit duration tasks created for an activity being equal to the original activity duration. The project scheduling problem with all unit duration tasks is then transformed similarly to the transformation which occurs in solving the comparable line balancing problem into a problem of finding the shortest route in a finite directed network (termed an A-network).

The A-network for problem solution is created by first generating the nodes of the network, each node representing a precedence feasible assignment for some subset of the unit duration tasks. Hence, resource restrictions are originally relaxed in their procedure in generating the tree of partial solutions. Arcs are then added to the network, each arc connecting a precedence *and* a resource feasible assignment for some subset of the unit-duration tasks in the project. The number of stages (levels) at which nodes are added in their approach is equal to the original critical path length, each level corresponding to a different period in network construction. The number of nodes (precedence feasible assignments) at each stage or at each level is determinable using a procedure due to Held, Karp, and Shareshian. Dynamic programming is used to determine the shortest route in the generated A-network of partial solutions, resulting in the determination of the minimal schedule length for the resource-constrained problem.

Because the number of precedence feasible assignments at each stage (level) can grow quite large, subset elimination criteria using both resource based and precedence based techniques are employed, each in turn being based on starting with a feasible (heuristic) solution to the problem. A latest performance time (LPT), for example, is established for each activity based upon achieving a selected target duration. Nodes corresponding to partial solutions in which an activity would be scheduled beyond its LPT are thus discarded (i.e., the candidate problem is omitted in network generation). Similarly, to each stage corresponds a minimum usage of required resources in order for sufficient resources to *remain* available to complete the project within the target duration. Partial solutions corresponding to insufficient resource usage amounts at a stage (for *any* of the resources involved) can thus be similarly discarded, or the candidate problem can be omitted. Methods are also used in their approach to reduce the number of arcs generated, recognizing that (1) arcs can only connect nodes within or between adjacent stages, (2) for an arc $i - j$ to exist, i must be less than j, and (3) at most one arc can lead into each precedence feasible node. Further, the nodes are segregated in network generation into (1) those that could appear on a path within the target duration or less and (2) those that could not, further reducing the computational effort required with their approach.

The primary operating constraint of their procedure appears to be the number of precedence feasible subsets saved after the subset elimination criteria described above are employed (these are consequently stored). Among the advantages cited for their approach are that (1) it is a relatively easy matter to split the activities in the original network with no additional increase in computational effort, thereby relaxing the activity continuity assumption, and that (2) resource requirements can vary over job duration, enabling activities to be completed with varying resource usage levels.

Description of Stinson's Branch and Bound Procedure

A branch and bound (skiptracking) procedure was developed by Joel P. Stinson (1976, 1978) to solve the multiple constrained-resource, project scheduling problem. His procedure has been tested extensively on both multi-resource, project scheduling problems, and on job-shop scheduling problems with notable success. Nodes in the branch and bound solution tree with Stinson's procedure correspond to precedence *and* resource feasible assignments for a subset of the activities of the project. Hence, restrictions or constraints on activities not yet scheduled at a given node are initially ignored in his approach, to be considered at a subsequent level in tree development. The maximum number of levels in the branch and bound solution tree with Stinson's procedure is equal to the number of activities to be scheduled, and the maximum number of nodes which can be created from a parent (previous level) node is 2^n, where n represents the number of activities which could be scheduled due to (1) not being previously scheduled and due to (2) having all predecessors completed at the given partial schedule.

The order in which candidate problems are considered in Stinson's approach is determined during execution of the algorithm. Nodes or partial solutions are considered "next" for further evaluation based upon a series of partial schedule attributes which comprise a "decision vector" in his approach. That is, the decision vector consists of a series of tie-breaking rules for selecting the next node or partial schedule to branch to, or the next candidate problem to consider. If no ties exist on the first attribute, the remaining three are ignored; if ties exist on the first attribute but not on the second, the remaining two are ignored; etc. After considering several alternate decision vectors, Stinson's procedure was ultimately programmed for candidate problem selection with a four-element decision vector consisting of the components (in order): (The largest of three lower bounds (described below); The current partial schedule "time"; The total resource idleness from past decisions; The total number of immediate followers of activities which could be scheduled at the given partial solution). Stinson also recognized that this skiptracking procedure (unless modified) would tend to grow uniformly downward, requiring vast amounts of computer memory, and would not concentrate on reaching terminal areas of the tree where complete schedules and improved bounds on the optimal solution might exist. Good partial schedules (where recognizable) can thus serve as a point of departure (i.e., search origin) from which the enumeration procedure can proceed directly downward in hopes of determining an improved upper bound and a complete schedule. Stinson's procedure establishes a new search origin for candidate problem selection as the first node or partial solution encountered that has a lower bound on project completion time greater than that of any previous solution encountered. This has the effect of generating good complete schedules and improved upper bounds on the completion time of a project earlier than if the changing of the search origin did not occur.

Candidate problems can be eliminated from consideration or "pruned" (authors' description) from the solution tree using Stinson's procedure whenever it can be established that further branching cannot lead to complete schedules which are better than other complete schedules that are (1) either known to exist or (2) could be developed in further branching from some other partial solution. These pruning operations are accomplished two ways in his approach: by dominance pruning and by lower bound pruning. Dominance pruning occurs whenever it can be shown that an activity can be left-shifted and the resulting schedule is both time and resource feasible. Lower bound pruning is accomplished in one of three ways: (1) A precedence based lower bound is computed as the earliest time that any unscheduled activity can

be started plus the critical path length of the remaining unscheduled activities; (2) A resource based lower bound is computed as the sum of the earliest start time for an unscheduled activity plus the work-period requirements for the unscheduled activities divided by the per-period availability of the resources for the resource yielding the maximum remaining length; and (3) a "critical-sequence" lower bound is computed by Stinson which *simultaneously* considers both precedence and resource constraints. The largest of the three lower bounds is taken as the lower bound for the partial schedule and is used to eliminate inferior partial schedules from consideration.

In initially evaluating his procedure, Stinson obtained optimal solutions for 97% of the problems attempted. Apparent in his experiments was the effect of the critical sequence lower bound in eliminating candidate problems. In those problems for which an optimal solution could not be found, for example, the influence of the critical sequence lower bound was much less pronounced.

Description of Talbot's Implicit Enumeration Procedure

An implicit enumeration procedure (backtracking) for solving the resource constrained, project scheduling problem was written by F. Brian Talbot (1976). The formulation used in his approach, although equivalent in an implementation sense to several binary programming formulations of this problem, uses instead integer variables, resulting in greatly reduced memory requirements over these other approaches. The procedure consists of a systematic evaluation (enumeration) of all possible job finish times for the activities of a project. Fathoming rules much stronger than those used with a general implicit enumeration algorithm are employed, however, to eliminate from explicit consideration possible job finish times (i.e., candidate problems) that cannot possibly lead to improved schedules. Most noteworthy in this regard is the concept of a "cut" introduced in his procedure to eliminate from explicit consideration possible inferior completion times for activities earlier in the enumeration phase of the algorithm.

The procedure begins by determining the earliest possible completion time for the first job (Job No. 1) in the project. Jobs (activities) are then considered consecutively to be assigned to their earliest feasible job completion time. When the last job of the network has been assigned a finish time, an improved solution (i.e., shorter duration schedule) has been found. If, in the augmentation phase of this procedure, an activity cannot be assigned a resource *and* a precedence feasible completion time below a calculated upper bound for the activity, then the procedure backtracks to the next lower job number. If the lower numbered activity can be assigned a finish time in a period which would not lengthen the duration of the existing or incumbent schedule, an attempt is again made to determine a feasible completion period for the original job for which a resource feasible finish period could not be determined. If a more attractive finish or completion period for the next lower numbered activity cannot be found, then backtracking proceeds to the second lower numbered job of the network; etc. Optimality is assured with this approach when the attempt is made to backtrack past Job No. 1, the first activity in the network.

Talbot's procedure is thus very similar to the Balasian (1965) implicit enumeration procedure, but with some very important differences. First, nonnegative integer variables are used rather than binary or zero-one variables. Second, the project scheduling problem is initially structured such that variables are augmented (i.e., candidate problems are considered) sequentially (by activity number) rather than by using feasibility tests as with a standard implicit enumeration approach. Much like the determination of the decision vector components used in Stinson's branch and bound procedure, much experimentation was performed in the development of Talbot's procedure to determine a likely best sequence in which to consider candidate prob-

lems. Further, the specific structure of the project scheduling problem is exploited in Talbot's approach in order to expedite fathoming (the elimination of inferior candidate problems) and accelerate backtracking over a standard implicit enumeration approach. Employing the concept of a network chain, for example, it is often possible to backtrack beyond the next lower activity number when infeasible completion times for select activities are considered in his approach. Note that unlike Stinson's procedure, Talbot's procedure determines the order of candidate problem selection *before* execution of the algorithm, rather than *during* execution. Note also that similar to Stinson's procedure, a node in the solution tree corresponds to a resource *and* a precedence feasible assignment for a subset of the activities in the project—with Talbot's procedure, these jobs or activities are always $j, j-1, j-2, \ldots, 1$. These two procedures are thus in contrast to Davis' procedure on this characteristic, where a node in the Davis procedure corresponds to a *precedence* feasible assignment for a subset of the activities in the project. As with the Stinson procedure, the maximum number of levels in the solution tree using Talbot's approach is equal to the number of nondummy activities in the original problem. The number of immediate descendant nodes from a parent node (corresponding to adjacent levels in the solution tree) is equal to the difference between an upper bound computed for the activity (which is updated as improved solutions are found) and the Early Finish of the activity (determined by conventional critical path methods) plus one with Talbot's procedure (i.e., the maximum number of completion periods or immediate descendant problems from a given node at level $j-1$ is equal to $u_j - EF_j + 1$—author's notation).

The network cuts used in Talbot's procedure for eliminating candidate problems from consideration are employed selectively and dynamically, recognizing that a partial solution fathomed or a candidate problem eliminated earlier in the enumeration phase of the algorithm will have much greater impact in eliminating inferior partial solutions from consideration than will fathoming later in the network. Working with 50 test problems, Talbot was able to reduce the average central processing time for solution by over two-thirds by employing the cut concept. Reductions in the variance in solution times employing the cut concept for eliminating inferior partial schedules have been equally impressive with his approach.

3. Techniques Not Investigated

Each of the solution procedures described in the previous section represents the current state of the art in its respective area for solving the resource-constrained project scheduling problem. For example, the branch and bound procedure of Joel Stinson considers the multiple resource problem, the problem of primary interest in this investigation, and is programmed with much stronger lower bound pruning procedures than are earlier attempts to solve this problem. For this reason, the branch and bound procedure of T. J. R. Johnson (1967) was not considered separately in this investigation. Similarly, the implicit enumeration procedure developed by Talbot (1976) (even omitting his cut concept) represents a superior approach for solving this problem over previous backtracking procedures. For this reason, the implicit enumeration procedures of Patterson and Huber (1974) and of Patterson and Roth (1976) were not considered separately.

One approach which was considered, but which was later abandoned for solving this problem consisted of the use of a problem formulator (matrix generator) and a general purpose solution procedure (MPSX/MIP/370) for solving this problem as an integer (binary) programming problem. Using Tomlin's integrated Special Ordered Sets (SOS) procedure to attempt solution of these specially structured binary programming problems (along with other program enhancements such as the pre-programmed selection of variables to consider for evalation to one) resulted in only the smallest

problems in the problem data base described in the next section being solved within the time limit imposed. For this reason, the solution approach was abandoned in favor of the more computationally attractive approaches described for solving this problem.

4. Description of Problems Solved

One hundred and ten test problems were assembled for evaluating each of the solution procedures described in §2. These problems represent an accumulation of all multi-resource problems existing in the literature today (that are readily available). Plus, some problems are included in the problem data base that have been optimally solved for the first time. The number of activities included in these test problems varies between 7 and 50, with the number of resource types required per activity varying between one and three. The majority of the projects (103) consists of activities which require the use of the full complement of three different resource types for their performance. Eleven of the problems included consist of the available literature problems at the time of the writing of Patterson and Huber (1974). These problems are more fully described there, and consist of projects with as few as 7 and as many as 22 activities per project. Eighty-three of the test problems served as the test vehicle for evaluating the Davis and Heidorn bounded enumeration procedure, and are more fully described in Davis (1969) and in Davis and Patterson (1975). Six additional problems were contributed to the problem data set by Edward W. Davis for evaluating these three solution approaches. These six problems represent actual project scheduling problems, and consist of between 22 and 35 activities per project, with three scarce resource types being required by each activity. Finally, ten problems consisting of 50 activities each are included that were used by Talbot and Patterson (1978) in evaluating Talbot's implicit enumeration solution procedure.

5. Computational Results

Each of the procedures described in §2 was programmed in FORTRAN V for use on an Amdahl 470/V8 computer. Each solution procedure was written to accept problems in a common problem data format to facilitate evaluation. Further, each solution procedure was redimensioned (where necessary) to accept problems with like characteristics. Each program in its currently dimensioned form will accommodate project scheduling problems with as many as 100 original activities per project, and with each activity requiring as many as three different resource types. In some instances, this meant a "down sizing" of the original versions of these programs in such areas as the maximum number of activities accommodated per project, the maximum number of resource types required per activity, etc. This was done in order to compare approaches on similar problem types and with similar limitations. Finally, all solution approaches were adapted to not require writing to auxiliary storage during execution.

Table 1 gives the primary computer storage required to implement each solution approach based upon the specifications described above. For each procedure, Table 1 gives the required bytes without input and output buffers considered (i.e., storage required for arrays, variables, and program steps), required storage with input and output buffers, and actual allocations based upon processing on a given computing system. These latter quantities are provided because the number of I/O buffers required for job processing are likely to differ at different installations, influencing the total amount of storage required. Further, the integer increments provided in requested storage are also bound to differ at other installations. The final column in Table 1 gives the primary computer storage to compile, load, and run each of the three programs at a particular facility.

TABLE 1

Primary Computer Storage Required with Each Solution Procedure

Program Name	Required Bytes Without I/O Buffers**	Required Bytes With I/O Buffers (Go Step)*	Allocated Memory (Bytes) UMC System*
DAVIS (Bounded Enumeration)	1,540,768	1,520 K	1,536 K
STINSON (Branch and Bound)	489,856	492 K	512 K
TALBOT (Implicit Enumeration)	76,088	88 K	128 K

*K = 1,024 Bytes.

**(I.e., storage required for arrays, variables, and program steps.)

Caution should be exercised in interpreting the quantities given in Table 1, as it is possible only with the implicit enumeration procedure of Talbot to know precisely and *in advance* of problem solution the amount of storage required to solve a given problem. This is because the order in which the candidate problems or partial solutions are considered (and hence the manner in which the tree of partial solutions is generated with Talbot's approach) is known in advance of problem solution (and in fact, this is one the strengths of using Talbot's approach to solve these problems). The storage requirements using Davis' or Stinson's solution procedures, on the other hand, are determined *during* execution of the algorithm, and *not before*. The actual amount of storage required is a function of the number of nodes in the solution tree that are stored with each of these other approaches, and this is often an indeterminable function of specific problem characteristics and node storage decision rules at the time of execution of either of their procedures. The storage quoted in Table 1 is thus an estimate of required storage based upon problem size characteristics for the problems attempted *and* recommendations provided by the authors of these two approaches.

A second reason why the quantities quoted in Table 1 should be interpreted with a deal of caution is that it is possible to program each procedure to write to auxiliary storage when primary memory is exceeded in storing the solution tree and required pointers. And in fact, the Davis and Heidorn procedure is very cleverly written to do just this and to use auxiliary storage when main computer memory is exceeded. (This enhancement gives one virtually an unlimited storage capacity for the problem solution tree using their approach, albeit at an increase in computation time.) For purposes of this investigation, however, it was decided not to confound other aspects of algorithm performance with the memory requirements/computer storage trade-off issue, leaving this matter for further investigation. Hence, the estimates given in Table 1 were used for actually compiling and running each of the procedures in our evaluation.

Using the computer primary storage estimates given in Table 1, each solution procedure was compiled and run on an Amdahl 470/V8 computer using the FORTRAN H compiler with optimization feature (OPT = 2). Within a time limit imposed of 5 minutes of CPU time per problem, each of the three procedures attempted solution of all 110 test problems in the problem data set described. Summary operating details with each of the solution procedures are given in Table 2. As indicated in Table 2, only the Branch and Bound procedure of Joel Stinson was able to solve all 110 test problems within the time limit of 5 minutes per problem imposed, and this procedure additionally required *much less* CPU time on average to solve a given problem. Note

TABLE 2
Summary Operating Characteristics of Each Solution Procedure

Characteristics	Solution Procedure DAVIS (Bounded Enumeration)	STINSON (Branch and Bound)	TALBOT (Implicit Enumeration)
Number of Problems Solved (out of 110)*	96	110	97
Average Solution Time Per Problem Solved**	14.02	0.82	14.98
Number of Problems in Which Minimum Solution Time Was Recorded	9	76	25

*Within time limit imposed of 5 minutes (CPU time) per problem.
**Amdahl 470/V8 CPU time, in seconds.

also, however, from Table 2, that each procedure was able to solve a portion of the test problems in less computation time than the other two approaches. Stinson's Branch and Bound procedure performed best on this performance criterion also, however, solving 76 of the 110 test problems in less computation time than the remaining procedures.

Step-wise regression (SAS—maximum R-squared improvement) was then used in an attempt to identify individual problem characteristics which might be used as a guide for indicating specific problems or specific classes of problems that might be more amenable to solution with one of the solution approaches. Problem and network characteristics from Patterson (1976) were calculated for each of the 110 test problems, and these computed characteristics served as independent variable values in an attempt to predict solution time with each of the three solution procedures. The independent variables used in this analysis fall into one of three categories: (1) Time and Network Based Scheduling Statistics Computed Prior to Critical Path Analysis of a Problem; (2) Time and Network Based Scheduling Statistics Computed Subsequent to Critical Path Analysis (based on the nonresource constrained version of each problem; and (3) Resource Based Scheduling Statistics Computed Subsequent to Critical Path Analysis. These later estimates provide an indication of the "tightness" or the "constrainedness" of resources in contributing to the increase in project duration beyond the critical path length. In addition, a regression model was also developed to estimate the number of precedence feasible subsets surviving elimination and consequently stored using the Davis and Heidorn approach. The *estimate* of the total number of precedence feasible subsets surviving elimination ($R^2 = 0.83$) was then used as the final independent variable in the regression analysis. In total, 48 independent variables served as candidates to enter the regression equation. The results of this regression analysis for each of the solution procedures are given below.

DAVIS (Bounded Enumeration)

The best one variable model for predicting solution time, as might be expected with this approach, was the estimated number of precedence feasible subsets surviving elimination as the independent variable. This result is expected because the larger the precedence feasible solution tree, the larger the number of candidate problems which

have to be considered explicitly. Pertinent statistics are given below:

$$\text{ESTIMATED SOLUTION TIME} = -5.1088 + 0.00266\text{FS}$$

$$R^2 = 0.44 \quad F = 63.14 \qquad \text{where:}$$

FS = the *estimated* number of precedence feasible subsets remaining after elimination.

The best five independent variable regression model resulted in an R-squared value of 0.49 with variables included (in addition to FS which wasn't removed from any of the regression models) reflecting characteristics of activity slack and, to a lesser extent, reflecting one measure of resource constrainedness. The full 48 independent variable model resulted an R-squared value of 0.61.

STINSON (Branch and Bound)

The first independent variable entered into the regression model using Stinson's procedure was also the estimated number of precedence feasible subsets surviving elimination. Pertinent statistics are provided below:

$$\text{ESTIMATED SOLUTION TIME} = -0.24 + 0.000133FS$$

$$R^2 = 0.47 \quad F = 75.01.$$

The best five independent variable model included (in addition to the estimated number of precedence feasible subsets surviving elimination) variables reflecting the constrainedness or tightness of resources, unlike the best five independent variable model for the Davis procedure, which included characteristics of the slack present in the activities of the project. The R-squared value for this five independent variable model was 0.52, and the R-squared value obtained for the 48 independent variable model was 0.82.

TALBOT (Implicit Enumeration)

The first independent variable entered into the regression equation for predicting solution time with the implicit enumeration procedure of Talbot is a measure of the average resource constrainedness for a problem. The inclusion of this independent variable first might be expected, as the higher the resource constrainedness in a problem, the more difficult it is for Talbot's procedure to penetrate to the lowest level of the solution tree to determine and verify the optimal solution. Regression results are provided below:

$$\text{ESTIMATED SOLUTION TIME} = -139.236 + 533.028\bar{X}\text{-CON}$$

$$R^2 = 0.32 \quad F = 33.09 \qquad \text{where:}$$

$\bar{X}$-CON = Average Resource Constrainedness (see Patterson 1976).

The variables included in the best five independent variable model for Talbot's procedure reflect additional measures of resource constrainedness, as well as include the original critical path length of the project. The R-squared value for the five independent variable model is 0.50, and $R^2 = 0.91$ for the full 48 independent variable model.

A simple linear regression equation was also developed for the Talbot procedure using the estimated number of precedence feasible subsets surviving elimination (FS) as the independent variable. The R^2 value obtained from this analysis is equal to 0.0009, indicating a lack of significance of this independent variable in predicting solution times using the depth first search solution strategy of the Talbot procedure.

One should contrast this result to the results obtained using the search strategies (order of candidate problem selection) of Davis and of Stinson, where this independent variable explained more of the variability in solution times for these two approaches than did *any* of the other independent variables included.

Finally, one additional experiment was performed in which the best five independent variable regression model for each procedure was used to estimate the computation time required to solve each problem. Using the minimum estimate obtained to select the solution approach for a given problem, interest centered upon the number of problems (out of 110) for which the solution procedure requiring the minimum computation time could be correctly determined. (Note that to be useful, such an evaluation model would have to predict correctly on at least 77 or 70% of the test problems, because if we simply use Stinson's procedure for problem solution, minimum solution times would result on 76 of the test problems as indicated in Table 1.) Based upon the regression results obtained, the procedure yielding the minimum computation time was correctly estimated on 98 of the 110 test problems, or on about 89 percent of them.

A further examination of problem characteristics revealed that once the number of precedence feasible subsets surviving elimination exceeds 5,000, the computation times with the bounded enumeration approach of Davis grow rapidly to the point where either one of the other solution approaches will likely be preferred. Similarly, the average resource constrainedness on the set of test problems examined varies between 0.20 and 0.50. Whenever this measure increases beyond 0.34, the implicit enumeration procedure of Talbot begins to experience a great deal of difficulty in determining the optimal solution to these problems. Of the 110 test problems attempted, both the Davis bounded enumeration and the Talbot implicit enumeration procedures were unable to obtain optimal solutions on eight common test problems. In all instances, these eight problems are characterized by a predicted number of precedence feasible subsets surviving elimination exceeding 20,000 and an average resource constrainedness exceeding 0.343.

Thus, the following characterization of the preferred solution (in terms of minimum computation time) approach (where computer primary memory is not a limiting factor) results. Whenever the predicted number of precedence feasible subsets surviving elimination is less than 5,000, the bounded enumeration procedure of Davis is likely to produce an optimal solution in the minimum computation time. This is especially true when the average resource constrainedness of a problem is less than 0.35. Whenever the average resource constrainedness is less-than-or-equal-to 0.30 and

TABLE 3

*Procedure Expected to Produce an Optimal Solution Requiring the Minimum Computation Time as a Function of Problem Structure**

		Predicted Number of Precedence Feasible Subsets Surviving Elimination		
		< 5,000	5,000–20,000	> 20,000
Resource Constrainedness	< 0.30	DAVIS	TALBOT	TALBOT
	0.30–0.35	DAVIS	STINSON	TALBOT
	> 0.35	STINSON	STINSON	STINSON

*Davis Procedure: Bounded Enumeration
Stinson Procedure: Branch and Bound
Talbot Procedure: Implicit Enumeration.

the predicted number of precedence feasible subsets surviving elimination is above 5,000, the implicit enumeration procedure of Talbot is likely to produce an optimal solution in the minimum computation time. In most other instances and when computer storage is not a limiting factor, the branch and bound approach of Stinson is likely to be preferred. (Whenever the predicted number of precedence feasible subsets surviving elimination exceeds 20,000 on a given problem and the average resource constrainedness exceeds 0.35, *all* procedures exhibit an expected increase in computation time, but the procedure of Stinson is less influenced by these particular values of problem characteristics and is the preferred solution approach.) These results are summarized in Table 3.

6. Extension of Solution Procedures to Problem Types Not Investigated in This Paper

Often, a given problem type does not satisfy all of the assumptions built-in to a design of each of these solution approaches. For example, it may be possible to begin an activity using one resource level, employ additional amounts of resources after the activity has been started, and then once again decrease the level of resource involvement as the activity is completed. Under this and other scenarios, a decision must be made in employing these solution procedures as to which one to modify to attempt solution. Several recommendations are provided below.

Activity Continuity Assumption

When a job or an activity can be split, this can be easily accommodated with each of the solution procedures by simply creating two or more activities to replace the split one. If it is then desirable to split the given task to effect a shorter duration schedule, the procedure attempting solution will do so. With the Branch and Bound procedure of Stinson and the Implicit Enumeration procedure of Talbot, such modification is likely to produce additional computation times with each procedure, and is likely to require additional storage with the Stinson procedure due to the effect of having more resulting jobs in the network to schedule. The Davis and Heidorn procedure, on the other hand, would require minimal (if any!) additional computational effort since each of the original activities is expressed as a series of unit duration tasks initially.

Constant Resource Usage by an Activity

This assumption is similar to the activity continuity assumption, in that the activity can be split at the durations at which the different resource usage levels could be employed. If resource levels for performing an activity can be considered to be a *variable* rather than a *given* in the problem, however, solutions using a procedure developed specifically for this problem by Talbot (1982) should be investigated.

Varying Resource Availabilities

Where the quantity of resources available over a schedule duration varies (caused by such factors as absenteeism, different lengths of the work week, vacations, holidays, etc.), the simplest modification to accomplish this would be in the Talbot implicit enumeration procedure—a resource availability array would simply need to be initialized at the availability rates per period. The Stinson and Davis and Heidorn programs, on the other hand, would both require extensive modifications to accommodate this variation of the problem, the Davis and Heidorn program in utilizing the resource based elimination criterion in eliminating precedence feasible subsets, and the very logic of the Stinson procedure in advancing the "clock" to the next scheduling period. This modification would most certainly increase the "size" of the branch and bound solution tree using Stinson's procedure.

7. Conclusion

Three optimization approaches for solving the single project, multiple-constrained resource, project scheduling problem were compared. The methods compared, all enumeration based, differ in such aspects as the order in which candidate problems are considered for evaluation and the methods used to identify and discard inferior partial schedules (solutions). The implicit enumeration procedure of Talbot requires far less computer storage than do the other two approaches investigated due to the manner in which candidate problems are considered. Further, computer storage required is always known in advance of problem solution. It is an effective problem solving technique whenever resource constrainedness in a problem is low, and would likely be the preferred solution approach where computer storage is a particularly limiting factor. Overall, the branch and bound solution procedure of Stinson which incorporates a provision for changing the search origin to identify attractive solutions earlier in the enumeration process produces solutions in the minimum amount of computation time, and likely would be the preferred solution approach in those instances in which computer memory is not limiting. Finally, in those instances in which the predicted number of precedence feasible subsets surviving elimination is low, the bounded enumeration procedure of Davis is likely to produce the optimal solution in the minimum amount of computation time.

Extensions of the procedures to treat problem types not considered in this investigation were also noted.[1]

[1]I would like to acknowledge the contribution of Edward W. Davis, University of Virginia, George E. Heidorn, Naval Postgraduate School (currently of IBM Research Labs), Joel P. Stinson, Syracuse University, and F. Brian Talbot, The University of Michigan, in the preparation of this paper. Without their cooperation and assistance in providing me with initial working versions of their solution procedures, this paper would not have been possible. The two data sets containing the test problems and the solution procedures (as well as the job control language for accessing both) were developed by Mr. Ron Howren of the School of Business Research Staff, University of Missouri–Columbia.

References

BALAS, E., "An Additive Algorithm for Solving Linear Programs with Zero-One Variables," *Oper. Res.*, 13, 3 (July–August 1965), 517–546.

———, "Project Scheduling With Resource Constraints," *Applications of Mathematical Programming Techniques*, Carnegie-Mellon University, Pittsburgh, 1970.

BRAND, J. D., W. L. MEYER, AND L. R. SHAFFER, "The Resource Scheduling Problem in Construction," Civil Engineering Studies, Report No. 5, Department of Civil Engineering, University of Illinois, Urbana, Ill., 1964.

DAVIS, E. W., "An Exact Algorithm for the Multiple Constrained-Resource Project Scheduling Problem," unpublished Ph.D. Dissertation, Yale University, 1969.

———, "Resource Allocation in Project Network Models—A Survey," *J. Indust. Engineering*, 17, 4 (April 1966), 177–188.

———, "Project Scheduling under Resource Constraints: Historical Review and Categorization of Procedures," *AIIE Trans.*, 5, 4 (December 1973), 297–313.

——— AND G. E. HEIDORN, "Optimal Project Scheduling under Multiple Resource Constraints," *Management Sci.*, 17, 12 (August 1971), B803–B816.

——— AND JAMES H. PATTERSON, "A Comparison of Heuristic and Optimum Solutions in Resource-Constrained Project Scheduling," *Managment Sci.*, 21, 8 (April 1975), 944–955.

FISHER, MARSHALL L., "Optimal Solution of Scheduling Problems Using Lagrange Multipliers. Part I," *Oper. Res.*, 21, 5 (September–October 1973), 1114–1127.

GORENSTEIN, S., "An Algorithm for Project Sequencing with Resource Constraints," *Oper. Res.*, 20, 4 (July–August 1972), 835–850.

GUTJAHR, A. L., "An Algorithm for the Assembly-Line Balancing Problem," MS thesis, Johns Hopkins University, 1963.

——— AND G. L. NEMHAUSER, "An Algorithm for the Line Balancing Problem," *Management Sci.*, 11, 2 (November 1964), 308–315.

JOHNSON, THOMAS J. R., "An Algorithm for the Resource-Constrained, Project Scheduling Problem," unpublished Ph.D. Dissertation, Massachusetts Institute of Technology, 1967.

KELLEY, J. E. AND M. R. WALKER, "Critical-Path Planning and Scheduling," Proceedings of Eastern Joint Computer Conference, December 1959, 160–173.

MALCOLM, D., J. ROSEBOOM, C. CLARK AND W. FAZAR, "Application of a Technique for Research and Development Program Evaluation," *Oper. Res.*, 7, 5 (September–October 1959), 646–669.

PATTERSON, JAMES H., "Project Scheduling: The Effects of Problem Structure on Heuristic Performance," *Naval Res. Logist. Quart.*, 23, 1 (March 1976), 95–123.

——— AND WALTER D. HUBER, "A Horizon-Varying Zero-One Approach to Project Scheduling," *Management Sci.*, 20, 6 (February 1974), 990–998.

——— AND G. ROTH, "Scheduling a Project under Multiple Resource Constraints: A Zero-One Programming Approach," *AIIE Trans.*, 8, 3 (December 1976), 449–456.

SCHRAGE, LINUS, "Solving Resource-Constrained Network Problems by Implicit Enumeration-Nonpreemptive Case," *Oper. Res.*, 18, 2 (March–April 1970), 225–235.

STINSON, JOEL P., "A Branch and Bound Algorithm for a General Class of Resource-Constrained Scheduling Problems," unpublished Ph.D. Dissertation, University of North Carolina at Chapel Hill, 1976.

———, EDWARD W. DAVIS AND BASHEER M. KHUMAWALA, "Multiple Resource-Constrained Scheduling Using Branch and Bound," *AIIE Trans.*, 10, 3 (September 1978), 252–259.

TALBOT, F. BRIAN, "An Integer Programming Algorithm for the Resource-Constrained Project Scheduling Problem," unpublished Ph.D. Dissertation, Pennsylvania State University, 1976.

———, "Resource-Constrained Project Scheduling with Time-Resource Tradeoffs: The Nonpreemptive Case," *Management Sci.*, 28, No. 10 (October 1982), 1197–1210.

——— AND JAMES H. PATTERSON, "An Efficient Integer Programming Algorithm With Network Cuts for Solving Resource-Constrained Scheduling Problems," *Management Sci.*, 24, 11 (July 1978), 1163–1174.

——— AND ———, "Optimal Methods for Scheduling Projects under Resource Constraints," *Project Management Quart.*, 10, 4 (December 1979), 26–33.