European Journal of Operational Research 64 (1992) 188–198
North-Holland

Scheduling a project to maximize its net present value: An integer programming approach

Kum Khiong Yang
Decision Sciences Department, National University of Singapore, Kent Ridge, Singapore 0511, Singapore

F. Brian Talbot
Operations Management Department, University of Michigan, Ann Arbor, MI 48109-1234, USA

James H. Patterson
Operations Management Department, Indiana University, Bloomington, IN 47405, USA

Received June 1990; revised August 1991

Abstract: We describe an integer programming algorithm for determining scheduled start and finish times for the activities of a project subject to resource limitations during each period of the schedule duration. The objective is to maximize the net present value of the project to the firm. A depth-first branch and bound solution procedure searches over the feasible set of finish or completion times for each of the activities of the project. Fathoming criteria based upon the concept of a network cut originally developed to solve the duration minimization version of this problem are extended in this paper to solve the net present value problem. These fathoming decision rules prevent many potentially inferior solutions from being explicitly evaluated. Computational experience reported demonstrates the efficacy of the approach.

Keywords: Scheduling; Net present value; Integer programming

1. Introduction

The application of optimization techniques to cost control has lagged far behind those directed at optimizing other measures of project performance. A review of the literature reveals that previous research has been focused primarily on the objective of minimizing project duration [1,2,5,9,10,17–20]. This is unfortunate when one considers that the single inducement leading to the involvement in any project is the project's potential of becoming a lucrative venture. Even if a manager cannot limit attention to the pecuniary aspects of a project alone, these aspects cannot be treated in cavalier fashion.

The procedure described in this paper determines activity completion times that maximize the net present value (NPV) of the project to the firm. Multiple limited-resources influence when activities can be performed, as in the resource-constrained, duration minimization version of this problem. In maximizing NPV, we assume that the start of each activity requires an initial capital investment that is recovered upon completion of the activity. Cash flows (cash payments and cash disbursements) occur during the performance of

Correspondence to: Prof. J.H. Patterson, Operations Management Department, Indiana University, Bloomington, IN 47405, USA.

each activity. A value at completion is determined for an activity by discounting capital requirements and associated cash flows to the end of the activity. The infrequent application of optimization techniques to cost control may be due in part to the difficulty of stating a suitable objective function that incorporates activity cash flow information in this way.

Russell [14] examines the unconstrained version of this problem by representing each optimization problem as a series of network flow problems. Russell then presents an effective procedure for solving the resulting series of network flow problems. By adding a project deadline constraint, Grinold [7] transforms Russell's formulation into an equivalent linear programming problem. Grinold's formulation results in a more efficient solution procedure for obtaining optimal solutions to the unconstrained problem.

Doersch and Patterson [4] present a binary linear programming formulation for maximizing project net present value. Their formulation considers capital rationing (availability) constraints, where progress payments can be reinvested in the project to increase the amount of capital available. Standard integer (binary) programming procedures solve the resulting formulated NPV problem. Smith-Daniels and Smith-Daniels [16] further extend the Doersch and Patterson formulation into a mixed integer linear programming model that includes material acquisition decisions. Their formulation incorporates the costs of material investments as part of the usage of capital. Formulations in [4] and [16] provide insights into modeling the maximization of NPV problem. There are, however, no reported experiments which describe computational procedures to solve either of these formulated problems. It is with this limitation or gap in reported computational experience in mind that we propose the specialized algorithm described in this paper.

Optimal solutions provided by our approach can be compared to heuristic procedures which have recently appeared in the literature [8,13,15] to solve this same problem. Comparison of net present value results can be made similar to the approach described in [3] for assessing heuristic performance in solving the duration minimization problem.

In Section 2 we present a conceptual formulation of the NPV maximization problem. This formulation serves as the basis for our enumeration scheme. The solution procedure is described in Section 3. It is a modification of the procedure proposed by Talbot [19] to solve the duration minimization problem. In Section 4 we describe the fathoming or schedule elimination rules used in our procedure. These cost-based rules trim many inferior solutions from the search, improving significantly the computational efficiency of our approach. Section 5 reports on computational experience with our procedure. The procedure described reliably solves problems in the range of 20–30 activities per project with due-dates close to the minimum resource-constrained duration.

2. Mathematical formulation

The value of an activity upon completion is given by

$$D_j = \sum_{t=1}^{d_j} F_{jt}\, e^{\alpha(d_j - t)} + C_j[1 - e^{\alpha(d_j)}]$$

where:

D_j = Terminal value of cash flows in activity j at its completion.

F_{jt} = Cash flows for activity j in period t, $t = 1, 2, \ldots, d_j$.

α = Discount rate.

d_j = Duration of activity j.

C_j = Capital investment required by activity j.

Given a value upon completion for each activity, we can now provide a conceptual formulation of the NPV maximization, resource-constrained problem.

$$\text{Maximize} \sum_{j=1}^{N} q_{f_j} D_j + q_{f_N} B_{f_N} \tag{1}$$

Subject to

$$\max\{f_n,\ n = \text{all predecessors of } j\} + d_j \le f_j,$$
$$j = 1, \ldots, N, \tag{2}$$

$$\sum_{S_t} r_{jk} \le R^*_{kt}, \quad k = 1, \ldots, K, \quad t = 1, \ldots, \mathrm{DD}, \tag{3}$$

$$f_N \le \mathrm{DD}, \tag{4}$$

where:

D_j = Terminal value of cash flows in activity j at its completion.

DD = Deadline period for the last activity (N) of the project.

d_j = Duration of activity j.

q_t = Factor for discounting over t time periods to time 0.

B_t = Bonus for completion of project at time t. ($B_t < 0$ implies that B_t is a penalty.)

f_j = An integer variable representing the finish time of activity j.

r_{jk} = Amount of resource k required when activity j is active.

R^*_{kt} = Total amount of resource k available at time t.

S_t = The set of activities active at time t.

N = Number of activities in project. N is also the number of the unique ending activity in the project.

K = Number of resource classes.

The objective is to maximize the project net present value (1). The first constraint set (2) maintains the precedence relationships among activities. All activities that technologically succeed other activities are constrained to begin after the immediate predecessor activities are completed. The second constraint set (3) limits the usage of renewable resources in each period of the schedule duration. Resources are classified as either renewable or non-renewable [11] in our approach. Renewable resources (which include equipment and manpower) are available in limited quantities each period. Through a suitable definition of r_{jk} and R^*_{kt}, the second constraint set (3) insures that the use of renewable resources does not exceed the amount available in each period. The final constraint (4) limits project completion to a negotiated project deadline, DD.

Each activity must be completed within its specified duration in the above formulation. Integer requirements, in addition to the constraint set (2), prevent activities from being interrupted or split once begun.

3. Solution procedure

At the start of the enumeration process, activities that are in an immediate precedence relationship are labeled (numbered) such that if activity m precedes activity n, then $m < n$. By always selecting for assignment the lowest numbered activity j that has not been assigned a feasible completion time, we insure that an activity is considered for resource and time assignment only if all of its predecessor activities have first been scheduled.

Once the due-date for the project (DD) is specified [1] we determine the latest possible finish time for each activity, u_j. This is accomplished as follows. The critical path late finish time (LF_j) for each activity j is determined using standard critical path procedures. The upper bound on the completion time for each activity u_j is then set equal to $u_j = \text{LF}_j + (\text{DD} - \text{LF}_N)$.

Resource restrictions are maintained in our procedure using two compact arrays, a resource requirement array containing r_{jk} and a resource remaining array containing R_{kt}. We initialize R_{kt} to R^*_{kt}, the amount of resource k available in period t. When we assign an activity a resource and precedence feasible completion time t^*, f_j is set equal to t^*, and R_{kt} is reduced by r_{jk} for $k = 1, \ldots, K$, and $t = t^* - d_j + 1, \ldots, t^*$.

The following eight-step procedure describes the algorithm.

Step 1. Initialize the incumbent objective function value and incumbent activity completion times $F_{i=1,\ldots,N}$ to 0. Assign the first activity to its earliest completion time, $f_1 = d_1$, and reduce R_{kt} by r_{1k} for $k = 1, \ldots, K$, and $t = 1, \ldots, d_1$. Set the objective function value equal to the terminal value of activity 1, discounted d_1 periods. Set $j = 2$.

Step 2. Assign activity j to its earliest feasible completion time. This is accomplished as follows. First, determine t^*, the latest finish time for all

[1] If there is no limit on the latest project completion time or due-date, one can set DD equal to the sum of the activity durations, d_j, for all activities, $1, 2, \ldots, N$. Setting the due date equal to the sum of the activity durations is likely, however, to increase significantly the amount of computation time required to solve a problem (see discussion in Section 5). One is better off in those instances in which a project deadline is not given to estimate a project deadline DD than to use the sum of the activity durations for the schedule horizon. In those instances in which the solution obtained is equal to DD, the deadline can be further increased and the problem resolved with this new bound.

immediate predecessors of activity j. $t^* = \max\{f_n$, all n which immediately precede activity $j\}$.

Step 3. Search the resource remaining array from period $t^* + 1$ to u_j for the earliest interval d_j periods long where $R_{kt} \geq r_{jk}$, for $k = 1, \ldots, K$. Suppose this interval is from $t^+ - d_j + 1$ to t^+. Then f_j is set equal to t^+, and r_{jk} is subtracted from R_{kt} for $k = 1, \ldots, K$, and $t = t^+ - d_j + 1, \ldots, t^+$. Update the objective function by adding the terminal value of activity j discounted by the present value discount factor, q_t, where t equals f_j. Otherwise, go to Step 5 if a feasible interval d_j periods long is not found within the interval $t^* + 1$ to u_j.

Step 4. A feasible completion time has just been assigned to activity j. If $j = N$, go to Step 7. Otherwise, set $j = j + 1$ and go to Step 2.

Step 5. A resource feasible completion time less than or equal to u_j cannot be assigned to activity j. Backtrack. If $j = 1$, go to Step 8. Otherwise, attempt to reassign activity $j - 1$ to its earliest feasible completion time greater than f_{j-1}. Set $j = j - 1$.

Step 6. Update the resources available in the resource remaining array by adding r_{jk} to R_{kt} for $k = 1, \ldots, K$, and $t = f_j - d_j + 1, \ldots, f_j$. Update the objective function by subtracting the terminal value of activity j discounted by the present value discount factor, q_t, where t equals f_j. Set $t^* = f_j - d_j + 1$. Go to Step 3.

Step 7. Activity N has just been assigned a feasible completion time, f_N. Add the appropriate bonus (or penalty) B_t, discounted to time $t = 0$ to the objective function value. Compare the current objective function value to the incumbent. If the current objective function value is greater than the incumbent, replace the incumbent objective function value and incumbent activity completion times $F_{i=1,\ldots,N}$ with those given by the current solution.

Because of *potential* positive effects on the objective function, further attempts to right-shift activity N to a feasible completion time greater than f_N is required [2]. Subtract the current bonus (or penalty) from the objective function value. Go to Step 6.

[2] This is a characteristic of the NPV problem that is not found when we attempt to minimize project duration. This is *one* of the characteristics which accounts for the NPV problem being inherently more difficult to solve.

Step 8. Optimality is achieved when we attempt to backtrack past activity one. (In 0–1 integer programming terminology, this is equivalent to failing to complete a partial solution in which the left-most variable has been complemented and underlined [6].) The incumbent solution is optimal.

One significant difference between a depth-first search algorithm for the NPV problem and those used in minimizing project duration concerns the updating of activity latest finish times. In solving the project duration minimization problem, we update (reduce) the latest finish time u_j of each activity j whenever we discover an improved schedule (shorter project duration). This reduces the remaining solution space in which a shorter project duration schedule potentially resides. When solving the net present value maximization problem, the optimal net present value can potentially occur at any of the original activity completion times from EF_j to u_j. Unfortunately, then, we cannot update the latest finish time u_j of each activity when a shorter project duration is found. Because of the inability to tighten the upper bounds or latest finish times dynamically when a shorter project duration is found, effective fathoming rules assume an even more significant role than they do then the upper bounds can be updated as in the duration minimization version of this problem. We describe next fathoming rules used to solve the NPV maximization problem.

4. Use of fathoming rules

In the absence of any fathoming criteria, the solution procedure described above amounts to a full or an *explicit* enumeration of all possible job finish times from EF_j to u_j for each job. Two fathoming rules are suggested here that eliminate from explicit consideration solutions that are known to be inferior. Such inferior solutions are *implicitly* enumerated in our approach. The two fathoming rules are based upon the concept of a network cut suggested by Talbot [19], and are implemented in the framework suggested by Geoffrion and Marsten [6] (see especially the flowchart on page 470 of their paper).

4.1. Review of network cuts for minimizing project duration

A brief review of the role of network cuts in solving the duration minimization version of this problem strengthens and motivates our discussion of the fathoming rules used in solving the NPV maximization problem. Figure 1 shows a hypothetical problem with different partial solutions obtained at successive stages of enumeration. We will focus our discussion on the three activities shown, plus the partition the cut provides for two remaining activity sets – those activities that precede the cut, and those that follow it. Fathoming is achieved for the duration minimization problem by considering the solution space of resources remaining after the cut as we attempt to right shift activities in the cut set. [3]

We initially identify all potential cuts in the network. Each cut corresponds to an integer time period c, and identifies when in the solution process schedule elimination or fathoming rules are applied. The term cut is used because it identifies how the time period c partitions or cuts activities on a Gantt chart. An integer time period c qualifies as a cut if the following two conditions hold:

Condition 1: There exists a job j^* with $\text{ES}_{j^*} = c + 1$.

Condition 2: There does not exist a job $j > j^*$ such that $\text{ES}_j < \text{ES}_{j^*}$.

In the hypothetical problem shown in Figure 1, the cut occurs at integer time period $c = 20$. Activities 5, 6, and 7 have been cut by the time period c. These three activities make up the *cut set* at time period $c = 20$. From Conditions 1 and 2 above, preceding activities 1 to 4 (not shown) have latest activity finish times u_j either equal to or earlier than period c. Activities 8 to N, those activities that must follow those in the cut set, have earliest activity start times later than period c. Several other such cuts at time periods $c \neq 20$ potentially exist in a network. Here we illustrate just one. The same type of partitioning and fathoming arguments provided below apply to each cut in the problem.

The network cut shown in Figure 1 assists in identifying partial solutions that are inferior to previously enumerated results. This occurs as follows. For any current partial solution PS (that is, a specification of finish times for a subset of the jobs 1, 2,...), whenever the last activity of the cut set is assigned a feasible completion time, the resources remaining for scheduling the remaining activities (those which occur after the cut) are evaluated. This evaluation requires only an examination of the total resources used in each period after period c by activities in the cut set. [4] We wish to identify which current partial solutions are dominated by previous partial solutions, and which are not.

A partial solution is said to be dominated when it has the same or lesser resources remaining in each period after the cut period than does a previously enumerated partial solution. When the objective is to minimize project duration, a partial solution with a smaller remaining solution space *cannot lead* to a shorter project completion time than one given by the saved partial schedule. [5] The dominated partial solution is consequently fathomed, and backtracking occurs.

To better illustrate the use of network cuts in fathoming a partial solution, we continue with the example shown in Figure 1. We assume that chart (i) is the initial partial solution. This partial solution is obtained during the start of the enumeration process. The completion times of activities 5, 6, and 7 in chart (i), $\{f_5, f_6, f_7\} = \{17, 23, 21\}$, are saved as the *first saved* partial solution, $\text{PS}_{20}(1)$, at cut period $c = 20$.

Charts labelled (ii), (iii), and (iv) in Figure 1 show partial solutions as job 7 (the last job in the cut set) is successively right-shifted to later com-

[3] In an analogous fashion, we consider the NPV contribution of activities before, in, and after the cut in solving the NPV maximization version of this problem.

[4] Each network cut partitions the activities of a project into three mutually exclusive sets:

(i) Activities which precede the cut set. These activities have latest finish times either equal to or earlier than period c. Hence, these activities do not require the use of resources after period c.

(ii) Activities in the cut set. These are the only activities that compete with activities in (iii) for resources after period c.

(iii) Activities which follow the cut set. These activities have earliest start times later than time period c. These activities do not require the use of resources at or before period c.

[5] Proofs of these assertions are found in Talbot and Patterson [20].

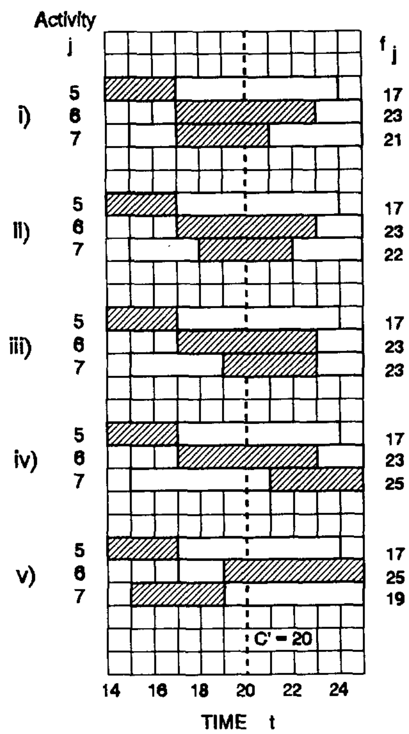

Figure 1. Gantt chart depicting use of network cuts

pletion times. In charts (ii) and (iii), activities 5 and 6 are assigned the same completion times as those in the first partial solution. The assignment of activity 7 to later completion times in these two cases uses more resources after period c without releasing resources during *any* period after period c. Since lesser resources are available for assigning the remaining activities (8 to N), partial solutions depicted in charts (ii) and (iii) are dominated by the saved partial solution $PS_{20}(1)$. Intuitively, the partial solutions depicted in these two charts are inferior since they do not provide any additional resources after the cut period for activities that can only begin after the cut. These partial solutions are thus dominated and we need not consider them further.

Further right-shifting of activity 7 leads to chart (iv). Chart (iv) is a potentially good partial solution because this assignment of job 7 frees resources in period 21 after the cut. The freed resources can *potentially* be used by activities 8 to N. The partial solution shown in chart (iv) is, therefore, not dominated by the saved partial schedule $PS_{20}(1)$. Augmentation proceeds to consider the assignment of finish times to activities 8,.... The solution depicted in chart (iv), $PS_{20}(2) = \{17, 23, 25\}$, is saved as the second partial solution at cut period $c = 20$.

Assume now that enumeration has proceeded to activity N. A feasible solution has consequently been found. As we continue with the enumeration, we finally backtrack to job or activity 7 again. At $f_7 = 25$, activity 7 cannot be right-shifted any further because it has been assigned a finish time equal to its upper bound latest finish time, $u_7 = 25$. Hence, we backtrack and consider activity 6. Activity 6 is right-shifted to its latest completion time of $u_6 = 25$. (This could allow job 7 to be completed in period 19.) With this assignment for activities 6 and 7, the partial solution shown in chart (v) results. This partial solution is compared to the saved partial solutions $PS_{20}(1)$ and $PS_{20}(2)$ depicted in charts (i) and (iv).

Comparing the start and finish times of activities 5, 6 and 7 in chart (v) with those of the two saved partial solutions shows that this is potentially a good partial solution. Activity 7 in chart (v) does not require any resources after period 20, while the same activity in the saved partial solutions (i) and (iv) requires resources after period 20. This current partial solution cannot yet be fathomed, and is saved as a third partial solution at cut $c = 20$. Augmentation continues once again to consider activities $8, \dots, N$.

Further discussion of the development of cuts (along with proofs that only potentially inferior solutions are eliminated) can be found in [20]. It has been our objective here to review how network cuts are used as fathoming rules to eliminate inferior solutions from consideration. Our arguments are based on the usage of resources by activities in the cut set after the cut period.

We now turn our attention to the NPV maximization problem and the extension of network cuts to solve this problem.

4.2. Extensions of network cuts for the maximization of project NPV

We will continue to use Figure 1 in illustrating the use of network cuts in fathoming partial solutions when maximizing the project net present value. Chart (i) in this figure continues to represent a current partial solution that is saved as the first partial solution obtained during enumeration. The contribution to the project NPV of activities 1, 2,..., 7 of this saved partial schedule is stored as $NPV_{(1-7)a}$. We continue with the assignment of activities 8 to N until an incumbent, feasible assignment for all activities is obtained. An enumeration of all possible completions of this saved partial solution is finished once we attempt to backtrack past activity 7. With the enumeration of all possible completions of the saved partial solution in chart (i) completed, the best NPV completion of the remaining activities for this saved partial solution is known. This maximum NPV contribution of the remaining activities is stored as $NPV_{(8-N)a}$.

Backtracking to activity 7 leads to chart (ii). Chart (ii) is obtained when activity 7 is right-shifted to a later feasible completion time. The fathoming rules for the NPV maximization problem are used whenever the last activity in a cut set is assigned a feasible completion time. The fathoming rules are described below.

Fathoming Rule 1 (F1). The NPV contribution of activities 1, 2,..., 7 of the partial solution shown in chart (ii), $NPV_{(1-7)b}$, is first compared to the net present value of the first saved partial solution, $NPV_{(1-7)a}$. If $NPV_{(1-7)b}$ is less than $NPV_{(1-7)a}$ (i.e. the NPV contribution of activities 1 to 7 of the current partial solution is less than that of the saved partial solution), the solution space of the current partial schedule available for the assignment of the remaining activities is compared to that of the saved partial schedule. Backtracking occurs if the remaining solution space of the current partial solution is a proper subset of the saved partial solution. Otherwise, we save the current partial solution as a potentially good partial solution. Or, we invoke the second fathoming rule, F2.

In invoking F1, we recognize that the potential contribution of the remaining activities to the net present value of a current partial solution is always less than that of a saved partial solution when the solution space of the current partial schedule for the assignment of the remaining activities is contained entirely within that of the saved partial schedule. The first fathoming rule thus works by eliminating any partial solution that is inferior to a saved partial solution, based on the fact that the NPV contribution of the current partial solution (activities 1 to 7) and the potential NPV contribution of its remaining activities are both less than that of the saved partial schedule.

Fathoming Rule 2 (F2). If $NPV_{(1-7)b}$ is *greater than* $NPV_{(1-7)a}$, and the remaining solution space of the current partial schedule is a subset of the remaining solution space of the saved partial schedule, an upper bound on the current partial

Table 1
Effect of Fathoming Rules Fl and F2 on computation times [a]

Problem [b]	No. of tasks	Problems with all positive activity cash flows				Problems with positive and negative activity cash flows			
		F1	F2	F1&F2	No. F [c]	F1	F2	F1&F2	No. F [c]
B, M & S	13	1.69	0.28	0.32	22.10	21.88	1.61	1.62	22.10
Davis	6	0.02	0.02	0.03	0.02	0.03	0.03	0.03	0.02
Davis	12	0.66	0.70	0.71	1.60	0.83	0.73	0.75	1.60
Davis	21	0.15	0.15	0.15	0.31	0.20	0.19	0.18	0.30
Davis	21	0.56	0.57	0.55	5.62	1.74	0.86	0.85	5.61
Davis	21	3.13	3.25	3.21	110.80	25.12	8.72	8.61	110.60
M&M	8	0.02	0.02	0.03	0.03	0.03	0.03	0.03	0.03
M&M	8	0.11	0.11	0.12	0.13	0.11	0.11	0.10	0.12
S, R & M	7	0.04	0.04	0.04	0.06	0.04	0.04	0.04	0.05
S, R & M	7	0.17	0.18	0.18	0.39	0.21	0.17	0.18	0.40
Average:		0.66	0.53	0.53	14.11	5.02	1.25	1.24	14.08

[a] IBM 4381 CPU time in seconds using FORT2VS compiler with optimization level 3.
[b] Problems modified with cash flow information from problem monograph by Patterson. Subsequent versions of the Davis 21-element, Moodie and Mandeville 8-element, and Shaffer, Ritter and Meyer 7-element problems have tighter resource limits with corresponding changes to cash flow amounts.
[c] Differences in CPU times without cuts are due to variability introduced through unequal loading of the CPU.

solution can be defined as the sum of $NPV_{(1-7)b}$ and $NPV_{(8-N)a}$. If this upper bound is less than the incumbent solution, the partial solution is fathomed and backtracking occurs. Otherwise we save the current partial schedule as a potentially good solution.

5. Computational experience

The enumeration scheme described in Section 3 has been implemented in four different solution modes: without any fathoming rules, with the first fathoming rule alone, with the second fathoming rule alone, and with both fathoming rules F1 and F2. When fathoming rules are invoked, several good partial solutions potentially have to be saved. Following the experience reported in [20], a maximum of the ten most recent good partial solutions are saved at each cut.

A set of ten problems (Table 1) has been solved using the four different solution modes for each problem. Each of the test problems is further solved as two separate instances – one in which each of the activities has a positive terminal activity value, and one in which each of the activities is associated with either a positive *or* a negative terminal activity value. In either instance, the net present value for the project is always positive. For each of these projects, the project due-date has been set equal to one period greater than the minimum resource-constrained duration. Since the algorithm without any fathoming rules consists of a complete (explicit) enumeration, the solution times in solving the two separate instances of the same test problem without the use of any fathoming rules are identical within the accuracy of the CPU timer used and the machine loading.

The ability of the fathoming rules to eliminate inferior partial solutions is clearly demonstrated by the large reduction in computation times shown in Table 1. When implemented alone, the first fathoming rule does not perform as well when both positive and negative terminal activity values exist for the activities. The reason for the lack of performance of the first fathoming rule in solving this instance of problems is predictable. Since the basic enumeration scheme proceeds by assigning activities to later and later completion times, the assignment of activities associated with negative terminal activity values to later completion times increases the net present value contribution of the activities before a cut, and hence disables the possible use of network cuts in fathoming potential inferior partial solutions.

The second fathoming rule implemented by itself performs extremely well in these experiments. For problems with both positive and negative terminal activity values, it clearly outper-

Table 2
Effect of varying horizon above resource-constrained duration (both Fathoming Rules employed) [a]

Problem [b]	No. of tasks	Problems with all positive activity cash flows			Problems with positive and negative activity cash flows		
		Minimum resource-constrained duration			Minimum resource-constrained duration		
		+1	+2	+3	+1	+2	+3
B, M & S	13	0.38	1.75	6.21	1.61	15.23	59.09
Davis	6	0.02	0.04	0.08	0.03	0.04	0.09
Davis	12	0.70	3.28	11.80	0.75	3.47	13.14
Davis	21	0.15	2.90	34.73	0.18	3.87	52.27
Davis	21	0.56	6.34	411.73	0.88	14.62	543.49
Davis	21	3.22	230.97	1913.05	8.68	729.72	1911.00
M&M	8	0.05	0.19	0.52	0.05	0.18	0.52
M&M	8	0.10	0.26	0.66	0.09	0.26	0.65
S, R & M	7	0.02	0.02	0.02	0.02	0.04	0.04
S, R & M	7	0.03	0.04	0.05	0.03	0.03	0.04
Average:		0.58	27.04	261.67	1.36	84.42	283.84

[a] IBM 4381 CPU time (in seconds) using FORT2VS compiler with optimization level 3.
[b] Problems modified with cash flow information from problem monograph by Patterson. See also Table 1.

forms the use of the first fathoming rule alone. While a great deal of care must be exercised in interpreting the averages given in Table 1, the trend is rather clear. The second fathoming rule is extremely effective in eliminating major portions of the search from explicit consideration.

When implemented independently, the second fathoming rule F2 eliminates many more solutions from explicit consideration than does the first fathoming rule, F1. Table 1 also shows, however, that when both fathoming rules are implemented together, they perform extremely well in eliminating inferior solutions from consideration. Since the amount of overhead required to implement both fathoming rules together is low, both rules are retained in our procedure.

In Table 2 we show the effect of varying the due-date beyond the one day extension to the minimum resource-constrained duration reported in Table 1. The increase in computation time required to solve a problem is significant when the potential schedule duration increases well beyond the minimum resource-constrained duration. This is because there are many more periods in which activities can be scheduled, and hence more potential solutions have to be both explicitly and implicitly evaluated. Also, as resources become more limiting, the effects of an increase in potential schedule duration are even more pronounced.

It is hence possible to identify before solution begins which problems are potentially very difficult to solve with our approach. Difficult problems are characterized by

(1) having resources that are quite limited, and

(2) having a due-date for project completion that is far removed from the minimum resource-constrained duration.

Fortunately, it is possible to determine in advance the gap between the project due-date and the minimum resource-constrained duration using one of several optimization approaches available for determining the minimum duration schedule. This results in an easy identification of problems which are potentially very difficult to solve.

In Table 3 we show the effect of varying the horizon in the absence of any fathoming rules at all.[6] A comparison of the results in Tables 2 and 3 shows the savings in computation time achieved using fathoming rules based upon network cut information. One would realistically never attempt to solve a problem with our approach without using the fathoming rules described.

Table 3
Effect of varying horizon above resource-constrained duration problems with positive and negative cash flows per activity (no fathoming rules employed)

Problem [b]	Solution times [a,c]			
	No. of tasks	Minimum resource-constrained duration		
		+1	+2	+3
B, M & S	13	22.09	211.63	1296.36
Davis	6	0.03	0.04	0.11
Davis	12	1.60	10.26	53.08
Davis	21	0.31	12.10	283.03
Davis	21	5.62	175.21	3157.71
Davis	21	110.79	712.35	28503.67
M&M	8	0.03	0.15	0.64
M&M	8	0.13	0.49	13.49
S, R & M	7	0.05	0.14	0.35
S, R & M	7	0.39	0.70	1.24
Average:		14.16	115.01	3357.14

[a] IBM 4381 CPU time (in seconds) using FORT2VS compiler with optimization level 3.
[b] Problems modified with cash flow information from problem monograph by Patterson. See also Table 1
[c] Because fathoming rules F1 and F2 are not employed, the only differences in timing results between problems with all positive and with positive and negative cash flows are attributable to unequal loading of the CPU.

Finally, in Table 4 we show changes in the net present value obtained by allowing for an extension of the schedule horizon (project deadline, DD) for two of the problems given in Table 1. These two problems were selected since each has activities near the end of the network with net cash outflows (or with negative capital inflows). The results are predictable. It is in a project manager's or contractor's best interest to delay the completion of these two projects as permitted by the project deadline. Hence, the timing and incidence of cash flows becomes a significant element of project negotiation and design. Stipulation of a net positive cash inflow in the last activity of a project (such as, for example, in painting the divider lines on a highway after other major portions of the highway are completed) can reduce the possibility of delaying the

[6] Since this amounts to a full enumeration of all possible schedules, it is not necessary that we solve both instances of each problem.

Table 4
Representative changes in net present value as a function of schedule horizon

Problem [a]	No. of tasks	Minimum resource-constrained duration		
		+1	+2	+3
B, M & S	13	146.91	147.01	147.11
M & M	8	235.93	242.61	243.54

[a] Problems modified with cash flow information from problem monograph by Patterson. See also Table 1.

completion of a project to achieve greater financial rewards. Such elements of project design should be considered at the time a project or a contract is awarded. The models described in this paper assist in identifying potential reasons for late project completion due to financial considerations.

6. Conclusion

We have described a procedure for solving the limited-resource project scheduling problem with the objective of maximizing project net present value. Computational experience with the procedure indicates significant reductions in the amount of computation time required to solve a problem using the fathoming rules described. The amount of computation time required for problem solution is in general lower the closer the due-date is to the minimum resource-constrained duration and the less limiting are the resources made available during each period of the schedule duration.

It is now possible to create a data set of optimal net present value schedules for projects involving as many as 20–30 activities per project. Such problems serve as benchmarks for comparing heuristic performance when solving the NPV maximization problem.

Extensions to our procedure that incorporate additional fathoming rules are certainly possible, opening up the possibility of solving larger problems than those reported here. The incorporation of additional fathoming rules is rather straightforward using the enumeration procedure described.

Finally, aspects of project design such as the incidence of cashflows in project activities can have serious implications in determining the 'optimal' time for project completion. Such design aspects are best considered in advance of project initiation.

References

[1] Christofides, N., Alvarez-Valdes, R., and Tamarit, J.M., "Project scheduling with resource constraints: A branch and bound approach", *European Journal of Operational Research* 29 (1987) 262–273.

[2] Davis, E.W., and Heidorn, G.E., "An algorithm for optimal project scheduling under multiple resource constraints", *Management Science* 17/12 (August 1971) 803–816.

[3] Davis, E.W., and Patterson, J.H., "A comparison of heuristic and optimum solutions in resource-constrained project scheduling", *Management Science* 21/8 (April 1975) 944–955.

[4] Doersch, R.H., and Patterson, J.H., "Scheduling a project to maximize its net present value: A zero-one programming approach", *Management Science* 23/8 (1977) 882–889.

[5] Demeulemeester, E., and Herroelen, W., "A decision support system for resource-constrained project scheduling", Department of Applied Economic Sciences, Katholieke Universiteit of Leuven, Leuven, Belgium, 1990.

[6] Geoffrion, A.M., and Marsten, R.E., "Integer programming algorithms: A framework and state of the art survey", *Management Science* 18/9 (1972) 465–491.

[7] Grinold, R.C., "The payment scheduling problem", *Naval Research Logistics Quarterly* 19/1 (1972) 123–136.

[8] Padman, R., Smith-Daniels, D.W., and Smith-Daniels, V.L., "Heuristic scheduling of resource-constrained projects with cash flows: An optimization-based approach", Working Paper 90-6, School of Urban and Public Affairs, Carnegie Mellon University, 1990.

[9] Patterson, J.H., "A comparison of exact approaches for solving the multiple constrained resource, project scheduling problem", *Management Science* 30/ 7 (1984) 854–867.

[10] Patterson, J.H., and Roth, G., "Scheduling a project under multiple resource constraints: A zero-one programming approach", *AIIE Transactions* 8 (1976) 449–456.

[11] Patterson, J.H., Slowinski, R., Talbot, F.B. and Weglarz, J., "An algorithm for a general class of precedence and resource constrained scheduling problems", in: R. Slowinski and J. Weglarz (eds.), *Studies in production and Engineering Economics, Volume 9: Advances in Project Scheduling*, Elsevier, Amsterdam, 1989, 3–28.

[12] Patterson, J.H., Slowinski, R., Talbot, F.B., and Weglarz, J., "Computational experience with a backtracking algorithm for solving a general class of precedence and resource constrained scheduling problems", *European Journal of Operational Research* 49 (1990) 68–79.

[13] Pinder, J.P., and Marucheck, A.S., "Maximizing project net present value: categorization of heuristic scheduling

performance", Working Paper, School of Business, University of Wisconsin, Madison, 1990.

[14] Russell, A.H., "Cash flows in networks", *Management Science*, 16/5 (1970) 357–372.

[15] Russell, R.A., "A comparison of heuristics for scheduling projects with cash flows and resource restrictions", *Management Science* 32/10 (1986) 1291–1300.

[16] Smith-Daniels, D.E., and Smith-Daniels, V.L., "Maximizing the net present value of a project subject to material and capital constraints", *Journal of Operations Management* 7/1 (1987) 33–45.

[17] Stinson, J.P., "A branch and bound algorithm for a general class of resource-constrained scheduling problems", in: *AIIE Conference Proceedings*, Las Vegas, NV, 1975 337–342.

[18] Stinson, J.P., Davis, E.W., and Khumawala, B.M., "Multiple resource-constrained scheduling using branch and bound", *AIIE Transactions* 10/3 (1978) 252–259.

[19] Talbot, F.B., "An Integer Programming algorithm for the resource-constrained, project scheduling problem", Ph.D. Dissertation, Pennsylvania State University, 1976.

[20] Talbot, F.B., and Patterson, J.H., "An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems", *Management Science* 24/11 (1978) 1163–1174.