European Journal of Operational Research 49 (1990) 68–79
North-Holland

Computational experience with a backtracking algorithm for solving a general class of precedence and resource-constrained scheduling problems

James H. Patterson
Indiana University, Bloomington, IN 47405, USA

F. Brian Talbot
University of Michigan, USA

Roman Slowinski and Jan Węglarz
Technical University of Poznan, Poland

Abstract: In this paper computational results are presented with a very general, yet powerful backtracking procedure for solving the duration minimization and net present value maximization problems in a precedence and resource-constrained network. These networks are generally of the PERT/CPM variety, although it is not required that they be so. Among the advantages cited for our approach are low computer memory (storage) requirements and the ability to obtain improved solutions rapidly (heuristic properties). Since the resource-constrained project scheduling problem subsumes the job shop, flow shop, assembly line balancing, and related scheduling problems, our procedure can be used with little or no modification to solve a wide variety of problem types. Computational experience is reported for both mainframe and personal computer implementations.

Keywords: Project management, resource-constrained scheduling, branch and bound, integer programming, net present value

1. Introduction

A very general, yet powerful backtracking procedure for solving non-preemptive, resource-constrained project scheduling problems is described. Among the advantages cited for the depth-first search approach are:

(1) Its simplicity.

(2) Low computer storage requirements.

(3) The ability to obtain both heuristic and optimal solutions to resource-constrained, project scheduling problems.

Quite often, for example, improved solutions to a problem are found quite rapidly with our approach, both for the objective of minimizing project duration (makespan) as well as maximizing project net present

Received November 1989

value. The solution procedures described herein incorporate features of both the resource-constrained project scheduling problem and the Decision CPM problem. That is, we solve the resource-constrained version of this important scheduling problem for the case in which alternate technologies or *modes* exist for completing the activities of a project, each mode with potentially differing costs, resource requirements, and time durations for activity completion. To our knowledge, the procedures described herein are the only ones available for solving these types of scheduling problems for the case in which we desire to either minimize project duration or else maximize project net present value.

Types of problems modeled with our approach differ from the traditional Decision CPM problem in that while we do consider alternate modes for completing an activity, the precedence relations in the network are fixed. Hence, we do not specifically consider alternate precedence relationships for the activities of a network as does the Decision CPM model.

In the next section, we describe in detail the broad types of problems solvable with our approach. Section 3 briefly describes the backtracking algorithm used for the case in which we desire to minimize project duration, and in Section 4 we describe modifications required to solve the maximization of net present value problem. Section 5 reports on computational experience with our approach, and Section 6 gives several promising extensions to these procedures.

An earlier version of the procedures described in this paper, but without the computational results reported herein was reported in [12]. This paper expands upon these procedures, as well as reports on an extensive computational experiment designed to assess the efficacy of our approach. We assume reader familiarity with depth-first and breadth-first types of search procedures. Although reader familiarity is also assumed with reference [12], of necessity, we present in abbreviated form portions of this paper essential to the development of the computational tests described.

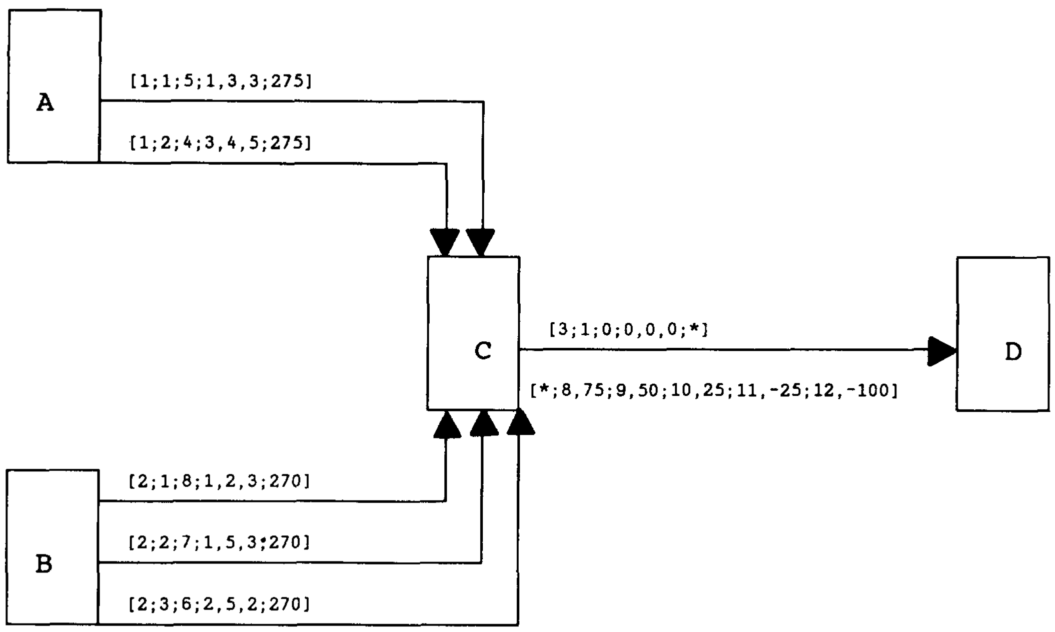
2. The project scheduling problem

2.1. Problem type investigated

In Figure 1 we show the multiple activity mode version of the resource-constrained project scheduling problem for three activities extracted from a larger project network model. In this example, the first two activities may be begun upon project initiation, resources permitting. The first activity can be completed using one of two different activity modes, with durations of 5 and 4 time periods and per period usage of resources as indicated. A progress payment of 275 is received upon the successful completion of this first activity. The cost of completing the first activity is either 150 or 200 depending upon the mode for activity completion selected. The second activity can be completed using one of three different activity modes, with resource requirements and activity durations indicated. Depending upon the mode selected for completing this second activity, a cash outflow of 208, 266, or 216 results. This latter activity (deciding between modes 2 and 3 for activity completion) illustrates the complex non-monotonic tradeoffs that exist among activity duration, activity cost, and resource consumption in the multiple mode, resource-constrained version of this problem. These types of relationships make the problem inherently more difficult to solve as much less of the structure of the problem can be used in developing bounds and improved fathoming criteria.

Completion by period 8 for both activities of this partial network implies a bonus of 75; completion in period 12 (and beyond) implies a penalty of 100 (-100). Per-period resources are available in the amounts 3, 6, and 7 each period of the schedule duration. The per period cost of these three renewable resources are 3, 4, and 5, respectively, as indicated in Figure 1.

Following the approach suggested by Slowinski and Weglarz [14,15,20,21], we classify resources as being *renewable*, *nonrenewable*, or *doubly constrained*. *Renewable* resources are available and are consumed on a per period basis. For example, skilled labor would be classified as a renewable resource if it is available in limited quantities and renews itself (is available) each period of the schedule duration. *Nonrenewable* resources are available on a total project basis. For example, cash may be considered nonrenewable if only a fixed amount is available in total for project completion. *Doubly constrained*



Key:

[Activity No.; Mode; Duration; Units of Resource 1 Reqd., Units of Resource 2 Reqd., Units of Resource 3 Reqd.; Performance Payment]

[*;Period of Completion, Bonus/Penalty Payment;•••; Period of Completion, Bonus/Penalty Payment]

Resource Limit:	[3, 6, 7]
Unit Cost of Resources:	[3, 4, 5]

Figure 1. A partial network depicting the multiple mode version of the resource-constrained project scheduling problem

resources are constrained on both a per period as well as on a total basis. Cash can be considered to be a doubly constrained resource if both total expenditures on a project are limited, as well as net cash outflows per period.

2.2. *Brief literature review*

Resource–duration interactions or the consideration of alternate operation or activity completion modes were first considered in the context of the CPM time–cost tradeoff problem. In this model, it is assumed that activity duration can be decreased monotonically (linearly) between the limits of the normal and crash activity durations simply by increasing expenditure levels. The allocation of renewable resources with resource–duration interactions has been examined by Elmaghraby [6]. The problem of considering resource–duration interactions under different operating modes for different combinations of renewable, nonrenewable, and doubly constrained resources for the preemptive version of this problem is reported by Slowinski [14] and by Weglarz [20,21]. The nonpreemptive version of this problem has been investigated by Talbot [18] for the objective of minimizing project duration.

2.3. *Notation*

Table 1 gives notation useful for describing the formulation and procedures developed in this paper.

Table 1
Notation

Symbol	Definition
$\alpha_j(\alpha_j')$	Due date of activity j
c_{jmd}	Cash flow of activity j, mode m, in its d-th period in progress ($d=1, 2,\ldots, D_{jm}$); if $c_{jmd} < 0$ there is a cash withdrawal, if $c_{jmd} > 0$ there is a cash inflow
c^*_{jmv}	Non-negative cash flow v periods after the completion of activity j ($v \geqslant 1$)
C_t	Net cash position in period t. C_0 is the cash available at the start of the project
D_{jm}	Activity duration associated with mode m of activity j; $D_{J1}=0$
E_j (L_j)	Critical path determined earliest (latest) completion time for activity j based upon shortest (longest) completion time mode for activities in the network
J	Unique ending (dummy) terminal activity
k	Index identifying renewable resources ($k=1, 2,\ldots, K$)
M_j	Number of modes associated with the completion of activity j ($m=1, 2,\ldots, M_j$)
P_j (S_j)	The set of immediate predecessor (successor) activities of activity j
P (S)	The set of all pairs of immediate predecessor (successor) relationships. $(a, b) \in P$ denotes activity a is an immediate predecessor of activity b
R_{kt}	The amount of renewable resource k available in period t
r_{jmk}	Per period amount of renewable resource k required to perform activity j using mode m
T^*	Due date for project
w_t	Single payment, present value discount factor for period t at interest rate i, $\left(\frac{1}{1+i}\right)^{t-1}$
ω_j	Weight assigned to activity j
x_{jmt}	A zero-one variable which is equal to 1 if activity j, using mode m, is completed in period t; equals 0, otherwise

2.4. Problem formulation

The zero-one integer program given by (1)-(5) models the project scheduling problem described above where the objective is to minimize project duration (completion time or makespan) for the case in which alternate modes exist for performing a subset of the activities of the project.

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

subject to

$$\sum_{m=1}^{M_j} \sum_{t=E_j}^{L_j} x_{jmt} = 1 \quad \text{for } j = 1, 2, \ldots, J, \tag{2}$$

$$-\sum_{m=1}^{M_a} \sum_{t=E_a}^{L_a} t x_{amt} + \sum_{m=1}^{M_b} \sum_{t=E_b}^{L_b} (t - D_{bm}) x_{bmt} \geqslant 0, \quad \forall (a, b) \in P, \tag{3}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M_j} \sum_{q=t}^{t+D_{jm}-1} r_{jmk} x_{jmq} \leqslant R_{kt} \quad \text{for } k = 1, 2, \ldots, K, \quad t = 1, 2, \ldots, T^*, \tag{4}$$

$$c_{t-1} + \sum_{j=1}^{J} \sum_{m=1}^{M_j} \sum_{q=t}^{t+D_{jm}-1} c_{jm(D_{jm}+t-q)} x_{jmq} + c^*_{jmv} x_{jm(t-v)} = c_t \quad \text{for } t = 1, 2, \ldots, T^*. \tag{5}$$

The objective of minimizing project duration (1) is achieved by scheduling the unique terminal activity as early as possible. Constraint set (2) insures that each activity will be completed during one time period only and using only one activity mode. Precedence relationships are maintained by (3) and renewable resource restrictions are imposed by (4). Constraints (5) are identities for the nonrenewable resource cash. These constraints insure that an activity mode is selected only if sufficient cash is available during each period of its duration. For the non-capital constrained version of this problem, constraint set (5) can be omitted. Alternatively, the beginning cash available C_0 can be made arbitrarily large.

Minimizing project duration often carries with it a simultaneous increase in project net present value, as the early completion of individual activities (as well as the project) implies the early advancement of cash payments, generally resulting in increased net present value amounts. The cash inflows, c^*_{jmv} in (5), arise from the completion of key activities and add to the funds available for project completion. Where the advancement of such individual cash flow amounts does not imply net present value maximization, we instead solve the NPV maximization problem directly. This objective, which has been considered in a capital-only constrained version by Doersch and Patterson [7] can be easily accommodated in our approach by substituting (6) for (1):

$$\text{maximize} \quad \sum_{t=1}^{T} (c_t - c_{t-1}) w_t + c_0. \tag{6}$$

Some caution is necessary when employing the formulation given by (2)–(6). Where only cash outflows exist for an activity, it is possible that the associated activity will be delayed indefinitely in an attempt to maximize project net present value. In such instances, the model can include a constraint such as given by (7) insuring the completion of the project by a specified due date, T^*:

$$\sum_{t=E_j}^{L_j} t x_{J1t} \leqslant T^*. \tag{7}$$

Performance measures other than minimizing project duration or maximizing project net present value can be accommodated with our approach. For example, using (8) we can minimize the mean weighted activity delay:

$$\text{minimize} \quad \frac{1}{J} \sum_{j=1}^{J} \omega_j \sum_{m=1}^{M_j} \sum_{t=\alpha'_j+1}^{L_j} (t - \alpha'_j) x_{jmt}. \tag{8}$$

In (9), the total number of tardy activities is minimized:

$$\text{minimize} \quad \sum_{j=1}^{J} \sum_{m=1}^{M_j} \sum_{t=\alpha'_j+1}^{L_j} x_{jmt}. \tag{9}$$

Finally in (10), we minimize the weighted flow time of activities:

$$\text{minimize} \quad \frac{1}{J} \sum_{j=1}^{J} \omega_j \sum_{m=1}^{M_j} \sum_{t=\alpha'_j+1}^{L_j} (t x_{jmt} - \alpha_j). \tag{10}$$

Similarly, restricting attention to strictly serial activities with corresponding restrictions placed on the usage of identifiable resource classes allows us to easily model the job shop and the flow shop scheduling problems with our approach. Other extensions are possible and require few (if any) modifications to our procedure.

3. The backtracking algorithm for minimizing project duration

Since specific details are given in [12], we give here a more general outline of the solution procedure. For all types of problems considered with our approach, a depth-first search based branch and bound implicit enumeration procedure is used.

The solution methodology consists of two phases, a problem initialization phase, and an enumeration phase. In the initialization phase (described more fully in Section 5), activities are renumbered according to priority dispatch scheduling rules, activity modes are sorted using similar types of decision rules, and activities are assigned critical path based completion times. The relabeling and sorting conventions selected guide the search procedure in determining an initial solution, and affect the order in which activities and modes are considered for assignment during the enumeration phase as well. The late finish times are generated in this phase, and are used as upper bounds on the completion time of each activity.

The enumeration or second phase consists of a procedure for systematically generating the solution space, and for implicitly or explicitly evaluating all partial solutions that could be extended to an optimal solution.

The search is structured as a precedence tree which results when activities are assigned resource and precedence feasible completion times. Once activity j is assigned to a feasible completion time, the list of currently precedent feasible, unscheduled activities is updated to include the immediate successors of activity j. An attempt is then made to assign the next activity from this updated list of precedence feasible activities. If such an assignment is not possible within the upper bound limits for this activity, the algorithm backtracks along the same branch from which it extended to the most recently created node. Another assignment is then considered, and so on. For the objective of minimizing makespan, when a new solution is found, the time based upper bounds are adjusted to be one less than the incumbent solution. For cost-based objective functions, only the objective function bound is adjusted. Optimality is guaranteed when a solution is found equal to a known bound, or when backtracking proceeds to dummy node zero. Fathoming rules include consideration of the continuously updated upper bounds on the completion time of each activity, as well as those procedures employed in considering only precedence and resource feasible assignments for an activity.

4. Modifications to the backtracking algorithm for maximizing project net present value

The maximization of net present value problem is inherently more difficult to solve to optimally due to the absence of strong, easily determined bounds on the objective function value, and the corresponding weakening of the time-based upper bounds that are available for the minimization of makespan version of this problem. Specifically, if the net cash flow for an activity is negative, then the optimal solution may no longer have the property that all activities can be assigned to their early finish or left-shifted resource and precedence feasible completion times. The algorithm for the net present value maximization problem must therefore be modified to examine partial solutions with activity completion times between the early finish and late finish critical path based bounds. The enumeration procedure starts with the activities scheduled to complete at resource and precedence feasible left-shifted or early finish times, and examines right-shifting activities to later time periods to permit examination of the resource and objective function tradeoffs that occur during such schedule construction.

In the absence of good cost or time-based bounds, this right-shifting significantly increases computation times over those required to solve the project duration minimization problem. To partially circumvent this shortcoming, a heuristic was developed that defeats right-shifting until x percent of the computation time is expended for problem solution. Then, for the remaining $100 - x$ percent of the time available, right-shifting is permitted. The effect of this heuristic is to more quickly identify improved solutions to a problem, yet still be able to right-shift the majority of negative cash flow activities that have feasible completion times greater than their critical path determined early finish times. Values of x between 70 and 90 percent were evaluated. For the problems tested, 90 percent appeared to be a good choice for the value of x. It should be indicated that optimality is no longer guaranteed when invoking this solution heuristic, although improved heuristic solutions quite often result. We evaluated the solution procedure both with and without the solution-time-based heuristic for right-shifting in our computational experiments.

When sufficient cash is not available in our procedure to continue examining a branch of the branch and bound solution tree, fathoming occurs, and the algorithm begins backtracking. This cost based fathoming rule is used in addition to those described in the previous section for solving the NPV maximization version of this problem.

5. Computational experience

5.1. *Characteristics of problems solved*

Ninety-one problems were computer generated for purposes of testing the enumeration procedures described in Sections 3 and 4. The number of problems attempted (n) for each activity level is indicated in Table 2. Problems with the fewer number of activities give us an opportunity to evaluate the enumerative properties of our approach in detail, while problems with the larger number of activities give us the opportunity to assess the efficacy of our approach for solving problems of the type more frequently found in practice.

In addition to indicating the number of problems attempted for projects with identical numbers of activities, other characteristics of the problems solved such as the average or mean number of modes per activity, the average per period cash outflow, and so forth are given in Table 2. These problem characteristics are similar to those used previously in examining resource-constrained project scheduling problems [9,18].

5.2. *Characteristics of enumeration procedures*

Three FORTRAN-77 programs were written to evaluate the approaches described. Each procedure was tested under two operating conditions, providing an opportunity to assess the importance of such factors as the order in which activities are considered for resource and time assignment in the enumeration procedure. We also evaluated each of the procedures using two different time limits (1 and 10 minutes of CPU time on an IBM 4381 computer) for obtaining and verifying the optimal solution to a problem.

Experience in solving the single-mode version of these combinatorial problems [9] leads us to believe we would be much more likely to *obtain and verify* the optimal solution on those problems with the fewer number of activities, as solvability often decreases rapidly with an increase in the number of activities. We further conjecture that the procedures will rapidly obtain *improved* solutions for each of the problems attempted, regardless of problem size or objective function type. Finally, the ability to obtain and verify the optimal solution to a problem will likely decrease as we move from the objective of minimizing project duration to maximizing project net present value.

The first FORTRAN-77 program, TEP1, solves the duration minimization version of this problem (1). It specifically accommodates capital constraints (5), but does not allow for the infusion of additional capital.

The second enumeration procedure, TEP2, also solves the duration minimization problem. However, in this version, capital inflows (as well as outflows) are considered.

Finally, TEP3 solves the net present value maximization problem directly (6) allowing for both cash inflows as well as cash outflows. This latter program implements all of the modifications described in Section 4.

Examining the enumeration procedures in this way gives us the opportunity to assess the efficacy of our approach as a function of the increase in complexity of the problem being solved.

5.3. *Importance of the order for considering activities for resource and time assignment in the enumeration procedure*

Using the partial network shown in Figure 1, note that upon project initiation, we can either consider Activity 1 for resource assignment first, or else can consider Activity 2 first. In general, if there are k

Table 2
Summary problem characteristics [a]

	(A)	(B)	(C)	(D)	(E)	(F)	(G)	(H)	(I)	(J)
10-Activity problems (n = 25)										
Maximum	2.30	5.88	9.00	6.37	34					
Mean	1.76	4.14		4.86	18	0.2587	1.31	259.67	3	3333.03
Minimum	1.30	3.26	1.00	3.22	8					
20-Activity problems (n = 25)										
Maximum	2.20	5.28	9.00	6.18	55					
Mean	1.88	4.62		5.47	37	0.2832	1.57	276.62	5	4200.0
Minimum	1.50	3.91	1.00	4.73	22					
30-Activity problems (n = 25)										
Maximum	2.20	5.53	9.00	6.29	80					
Mean	1.91	4.70		5.49	54	0.2826	1.73	266.85	5	4800.0
Minimum	1.60	4.05	1.00	4.59	35					
50-Activity problems (n = 10)										
Maximum	2.26	5.68	9.00	6.13	125					
Mean	1.93	4.77		5.59	94	0.2948	1.87	278.07	4	23750.0
Minimum	1.60	4.40	1.00	4.93	59					
100-Activity problems (n = 5)										
Maximum	2.06	5.16	9.00	5.97	195					
Mean	1.96	4.88		5.68	173	0.2889	2.04	272.47	6	14333.0
Minimum	1.85	4.43	1.00	5.38	137					
500-Activity problems (n = 1)										
Maximum	1.98	4.90	9.00	5.72	836					
Mean	1.98	4.90		5.72	836	0.2909	2.06	271.90	5	78000.0
Minimum	1.98	4.90	1.00	5.72	836					

[a] (A) Mean number of activity modes.
(B) Mean activity duration.
(C) Minimum and maximum activity duration.
(D) Standard deviation of activity durations.
(E) Critical path length (based on minimum activity durations).
(F) Average fraction of resources used by activity mode.
(G) Network density (ratio of arcs to nodes).
(H) Mean per-period cash outflow.
(I) Number of positive cash payments.
(J) Mean magnitude of positive cash payments.

n = Number of problems in each set.

Note that the majority of the problem characteristics reported are themselves averages or means. For example, the mean number of activity modes for the 10-activity networks (1.76) is the average (over the 25 problems solved) of the average number of modes per activity in the 10-activity networks. Other problem characteristics are interpreted similarly.

activities available for resource and time assignment at a given time (that is, k activities for which all of their immediate predecessors are completed or have been scheduled), there are $k!$ possible orders for considering the activities in the enumeration procedure. Talbot [17] has shown in the single mode version of this scheduling problem that the order in which activities are considered for time and resource assignment can have a significant impact on the computer time required to obtain and verify an optimal solution to a problem.

Talbot's experiments show that in general, for the single mode, makespan minimization version of this problem, and within precedence restrictions, considering activities in the order of minimum activity total slack, MINSLK, usually results in the minimum amount of computer time required for problem solution. Using this logic, we might consider the activities of Figure 1 in the order Activity 2 first, and then Activity 1 based upon the critical path based minimum activity total slack. (With the largest activity duration considered for Activity 2, Activity 1 possesses positive units of float or total slack. Activity 2 possess none.) For the maximization of net present value problem, it may seem more logical to consider the activities in the order Activity 1 and then Activity 2, as Activity 1 has a larger cash payment upon completion. However, in solving the 91 test problems, we found that solution times were generally less if the activities were again considered in the order of least total slack (critical path based, using the minimum activity duration for each activity), again with the renumbering or the relabeling of activities taking place within the precedence restrictions of a problem.

The 'within precedence restriction' decision rules for considering activities for resource assignment operate as follows. First, for all three implementations, we consider activities in a RANDOM order. This is accomplished by relabeling (renumbering) the activities in a random order, but such that a lower numbered activity always precedes a higher numbered one. We then consider the activities for time and resource assignment in their sorted (random) order in the enumeration procedure. Alternatively, we can sort the activities (again within precedence restrictions) in the order of minimum activity total slack or float. (In the multiple mode version of this problem, we use the *minimum* activity duration in determining the critical path based total float for an activity.)

Other priority dispatch, heuristic sorting or relabeling decision rules are, of course, possible (several in addition to sorting on maximum net present value were tested in our experiments). We found that the minimum activity slack (MINSLK) decision rule in general yields low computation times in comparison to the other decision rules tested. The RANDOM decision rule, on the other hand, considers activities in the order of activity number unless there is some reason a priori for numbering the activities of the network (other than assigning successors of an activity a higher number than the activity number). The RANDOM rule thus operates as if no consideration is given to the order in which to consider activities for resource assignment. This ordering scheme is used in many search type algorithms found in the literature for solving the single mode version of this resource allocation problem since little (if any) consideration is given to the labeling or numbering of the activities other than having a higher numbered activity always succeed a lower numbered one.

For each enumeration procedure tested, problems were solved using both activity enumeration priorities, MINSLK and RANDOM. Further, with the MINSLK rule, time limits of 1 and 10 minutes per problem are used to assess the affects of a greater amount of CPU time on solvability. With the RANDOM rule, all problems are solved with a 10 CPU minute limit per problem.

Paired t-tests are used to compare alternate methods for considering activities for resource assignment, as well as alternate solution times with each approach. These results are reported below.

5.4. Minimize project duration with no provision for additional cash inflows (TEP1)

The critical path solution (average for these 91 test problems based upon the minimum duration completion mode for each activity) is 59.5 time periods. On average, the solution found within 1 minute of CPU time per problem using the MINSLK rule for considering the activities in the enumeration procedure is 84.92 time periods, with optimal solutions being found *and verified* on 33% of the problems attempted. As might be expected, the preponderance of problems for which optimal solutions were found and were verified are in the 10–30 activity range, with verification not being obtainable for any of the 100 activity problems or above.

When the time limit is increased to 10 minutes CPU time per problem, the average schedule length decreases to 84.02 time periods, a 0.90 period improvement in makespan per problem. The percent of optimal solutions obtained also increases from 33% to 36.2% of the problems attempted with an increase in the time limit to 10 minutes per problem.

Our conjectures of problem difficulty increasing rapidly as a function of the number of activities in a network and the ability of the procedures to rapidly obtain improved solutions to a problem regardless of network size seems to have held in examining the test problems. With the 1 minute time limit per problem imposed, the improvement in schedule length of the final solution obtained over the first solution determined is approximately 6 time periods. An additional 9 minutes of CPU time allowed per problem improves the solution by only nine-tenths of a time period.

We also compared the MINSLK activity selection (enumeration) rule with a 1 minute time limit imposed per problem to the RANDOM rule with a 10 minute time limit imposed per problem. The results are quite dramatic. The first solution found using the RANDOM rule averages 103.75 time periods, and improves to only 89.50 time periods in the 10 minutes allowed per problem. This best solution obtained is only 0.68 periods less than the *first* solution obtained using the MINSLK rule, and is more than 4.50 periods greater than the best solution found using the MINSLK rule with only one-tenth of the CPU time. In all instances, these differences are significant with a p-value of 0.01 or less (often, $p < 0.0001$). Hence, incorporating properties of the optimal solution such as the completion times of activities with little or no critical path based total slack into the logic of the search procedure has a lot more to do with solvability or the quality of the final solution obtained than does the brute-force enumerative speed of a very fast digital computer. This is a very refreshing, yet not totally unexpected result which demonstrates that the order in which activities are considered for time and resource assignment has significance not only for the solution procedures reported here, but for existing enumerative procedures as well.

5.5. *Minimize project duration allowing for additional cash inflows* (TEP2)

The results obtained allowing for the infusion of additional capital (e.g., progress payments) are similar to those obtained when cash flow payments are not subsequently reinvested back into a project. Using the MINSLK enumeration strategy, the percent of problems for which an optimal solution (minimum duration) was found and was verified increases from 33% to 35.2% as the solution time increases from 1 to 10 minutes CPU time per problem. Although statistically significant, the decrease in final schedule lengths achieved with the larger time limit is only 0.66 time periods on average (82.58 vs. 83.24 time periods). With the RANDOM search procedure, the average length of the final solution is 4.40 periods longer than with the MINSLK procedure when the MINSLK procedure is again allowed only one-tenth of the CPU time as the RANDOM rule.

Frequently, the objective of minimizing project duration is used in practice even though the reason for accepting a project has to do with its profitability or net present value to the firm. There are a variety of reasons why this is so (such as the difficulty of properly formulating the problem in an NPV context). We thus measured the cash flow associated with each problem in testing this version of the solution procedure, even though the objective centered on minimizing project duration. (These results would be similar to the results reported in the next section for the first solution obtained if we were to use a discount value of zero.) Using the MINSLK enumeration decision rule and a 1 vs. 10 minute time limit for problem solution, the average cash flow amount increases from \$21 962 to \$21 978 when the time limit is increased from 1 to 10 minutes. This is a very small increment, indeed.

Using the RANDOM enumeration decision rule and a 10 minute time limit per problem, the final cash flow (average) is \$21 776, or is well below the initial solution found using the MINSLK rule. The difference in cash flow results are significant beyond the $p = 0.003$ level. These results suggest that a wise choice in determining the order in which to consider activities in the enumeration procedure has as much impact in solving the cash flow version of this problem as it does in solving the project duration minimization problem directly. Further, the procedures developed, when given the MINSLK choice in the order in which to search for improved solutions, generally provide *very good* solutions early, with large increases in computation time not producing significantly better (in a practical sense) results.

5.6. *Maximizing project net present value* (TEP3)

As was suspected, the ability to obtain and verify an optimal solution decreases rapidly when the maximization of net present value criterion is used. In fact, without a constraint specifying some upper limit or due date for a project (7), we were not able to obtain and verify an optimal solution on any of the problems using either scheme for considering the order in which to assign activities completion times in the enumeration procedure. Without some limit on the date by which a project is due, the procedure has a great deal of difficulty determining when activities with a net negative cash flow should be scheduled. In fact, the procedure is searching for the *latest possible* period in which to schedule such activities, and without a specified upper limit, the search interval becomes quite large. While these results may on the surface appear to be discouraging, it is unusual that some due date would not exist for a project. When due date constraints are added (7) to the problem, solvability increases significantly for the problems attempted. Further, with right-shifting limited to the final ten percent of the solution time, improved solutions are found quite rapidly with our approach.

Using the MINSLK enumeration rule, the initial NPV solution (on average) is \$20014. This increases to \$20152 after 1 minute of enumeration time, and to \$20170 after 10 minutes of enumeration time with a discount factor of 1% per month. Thus, the additional 9 minutes of search time results in little improvement in the final results.

The results achieved using the RANDOM activity selection rule are as expected. The initial solution (on average) is \$19518 and this increases to only \$19802 in 10 minutes of CPU search time per problem. Once again, a clever strategy for guiding the search has a much more significant impact on the final solution obtained than does a less well thought out strategy using brute-force computation. In general, all results we achieved are significant beyond the $p = 0.01$ level, with the exception of the increase in NPV with the MINSLK rule and 10 vs 1 minutes for solution.

While searching for the optimal solution considering first those activities with the higher NPV amounts (again, within precedence restrictions) would appear to be a preferable strategy to searching based upon MINSLK, solving these 91 test problems in this fashion led to no significant improvement over using the MINSLK rule.

5.7. Personal computer implementation of the enumeration procedures

Because the enumeration procedures use a LIFO based search strategy and require relatively small amounts of computer memory, they are easily adapted for Personal Computer (PC) implementation. In fact, the only changes which had to be made between the mainframe and personal computer versions dealt with input and output conventions and resulting file manipulating requirements, plus changing the calls for the internal CPU clock. Normally, these same changes would also have to be made in adapting these procedures between mainframe versions. Once these changes are known, they can be made in about 20 minutes time.

The EXE files on a 386-based, 20 MHz PC with a numeric coprocessor produce results which are on the order of 7 times slower than the mainframe versions (using an IBM 4381 as a basis of comparison). Thus, to get comparable results on a PC to those quoted above would require making runs with a 7 and a 70 minute time limit per problem. Recall, however, that the procedures very rapidly find improved solutions to a problem, especially when directed to search for solutions considering first those activities with the minimum amount of critical path based total slack. We found no significant differences in results obtained using all three FORTRAN implementations when allowing 5 minutes CPU time per problem on a PC and 1 minute CPU time on the mainframe IBM 4381. While very few problems could be solved to optimality (including verification) in 5 minutes CPU time on a PC, we regard the results obtained with the search procedures used to be extremely encouraging. Our conjecture at this time is that additional computer time allowed per problem will result in small improvements in the objective function values and some additional verifications of the optimality of the final solutions obtained, although the frequency of verification will most likely be very low.

6. Summary

We have presented computational results for several optimal seeking algorithms that, with no or with only minor modifications, are capable of solving a large class of precedence and resource constrained scheduling problems. Mainframe and personal computer implementations indicate that our approach provides extremely good heuristic solutions to realistic size problems, and optimal solutions to small problems are obtained quite often. Since this approach has been programmed to be fairly general in application, there is much that can be done to decrease the computation time required to solve a specific class of problems. Significant decreases in computation times can be achieved, for example, through the development of search, fathoming and bounding rules based upon *specific problem structures*. In order to keep our procedures as general as possible, few if any of these refinements for specific problem classes were programmed into our search procedures. In particular, there are many more opportunities in addition to those described in the paper to develop time and cost based bounds that can significantly decrease the computation time required to solve the net present value maximization version of this important scheduling problem.

References

[1] Blazewicz, J., Cellary, W., Slowinski, R., and Weglarz, J., *Scheduling Under Resource Constraints – Deterministic Models*, J.C. Baltzer AG, Basel, 1986.

[2] Davis, E.W., "Project scheduling under resource constraints – Historical review and categorization of procedures", *AIIE Transactions* 5(4) (1973) 297–312.

[3] Davis, E.W., "An exact algorithm for the multiple constrained-resource project scheduling problem", Unpublished Ph.D. Dissertation, Yale University, May 1968.

[4] Davis, E.W., and Heidorn, G.E., "An algorithm for optimal project scheduling under multiple resource constraints", *Management Science* 17(12) (1971) B803–B816.

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

[6] Elmaghraby, S.E., *Activity Networks: Project Planning and Control Network Models*, Wiley, New York, 1977.

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

[8] Hindelang, T.J., and Muth, J.F., "A dynamic programming algorithm for decision CPM networks", *Operations Research* 27(2) (1979) 225–241.

[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 Huber, W.D.,"A horizon-varying, zero-one approach to project scheduling", *Management Science* 20(6) (1974) 990–998.

[11] Patterson, J.H., and Roth, G.W., "Scheduling a project under multiple resource constraints: A zero-one programming approach", *AIIE Transactions* 8(4) (1976) 449–455.

[12] Patterson, J.H., Slowinski, R., Talbot, F.B., and Weglarz, J., Chapter 1 in: *Advances In Project Scheduling* (R. Slowinski and J. Weglarz, eds.), Elsevier, 1989.

[13] Pritsker, A.B., Watters, L.J., and Wolfe, P.M., "Multiproject scheduling with limited resources: A zero-one programing approach", *Management Science* 16(1) (1969) 93–108.

[14] Slowinski, R., "Two approaches to the problem of resource allocation among project activities—A comparative study", *Journal of the Operational Research Society* 31(8) (1980) 711–723.

[15] Slowinski, R., "Multiobjective network scheduling with efficient use of renewable and non-renewable resources", *European Journal of Operational Research* 7(3) (1981) 265–273.

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

[17] Talbot, F.B., "An integer programming algorithm for the resource-constrained project scheduling problem", Unpublished Ph.D. Dissertation, The Pennsylvania State University, November 1976.

[18] Talbot F.B., "Project scheduling with resource-duration interactions: The nonpreemptive case", *Management Science* 28(10) (1982) 1197–1210.

[19] 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.

[20] Weglarz, J., "On certain models of resource allocation problems", *Kybernetes* 9(1) (1980) 61–66.

[21] Weglarz, J., "New models and procedures for resource allocation problems", *Proceedings, 6th INTERNET Congress, Vol. 2*, VDI-Verlag GmbH, Düsseldorf, 1979, 521–530.

[22] Weglarz, J., Blazewicz, J., Cellary, W., and Slowinski, R., "An automatic revised simplex method for constrained resource network scheduling", *ACM Transactions on Mathematical Software* 3(3) (1977) 295–300.