MANAGEMENT SCIENCE
Vol. 28, No. 10, October 1982
Printed in U.S.A.

RESOURCE-CONSTRAINED PROJECT SCHEDULING WITH TIME-RESOURCE TRADEOFFS: THE NONPREEMPTIVE CASE*

F. BRIAN TALBOT†

This paper introduces methods for formulating and solving a general class of nonpreemptive resource-constrained project scheduling problems in which the duration of each job is a function of the resources committed to it. The approach is broad enough to permit the evaluation of numerous time or resource-based objective functions, while simultaneously taking into account a variety of constraint types. Typical of the objective functions permitted are minimize project duration, minimize project cost given performance payments and penalties, and minimize the consumption of a critical resource. Resources which may be considered include those which are limited on a period-to-period basis such as skilled labor, as well as those such as money, which are consumed and constrained over the life of the project. At the planning stage the user of this approach is permitted to identify several alternative ways, or modes, of accomplishing each job in the project. Each mode may have a different duration, reflecting the magnitude and mix of the resources allocated to it. At the scheduling phase, the procedure derives a solution which specifies *how* each job should be performed, that is, which mode should be selected, and *when* each mode should be scheduled. In order to make the presentation concrete, this paper focuses on two problems: given multiple resource restrictions, minimize project completion time, and minimize project cost. The latter problem is also known as the resource-constrained time-cost tradeoff problem.

Computational results indicate that the procedures provide cost-effective optimal solutions for small problems and good heuristic solutions for larger problems. The programmed solution algorithms are relatively simple and require only modest computing facilities, which permits them to be potentially useful scheduling tools for organizations having small computer systems.

(PROJECT MANAGEMENT—RESOURCE CONSTRAINTS; PROGRAMMING—INTEGER ALGORITHMS, ENUMERATIVE; INDUSTRIES—CONSTRUCTION)

1. Introduction

Over the past decade attempts to actually solve multiple resource-constrained project scheduling problems of either the preemptive [13] or nonpreemptive [1], [4]–[6], [10], [11] varieties have been restricted to the case in which each job or activity in the project could be performed in only one prescribed way. Specifically, each job was chacterized by a unique duration and a singular complement of resource requirements that had to be met each time period the job was in process.

Only recently has a serious effort been made to formulate and solve the more general preemptive project scheduling problem where activities are described by continuous performance speed-resource functions [9], [15], and discrete time-resource functions [7], [8]. However, very little [2, pp. 173–183] has yet been reported for the important generalized nonpreemptive case where job performance is a function of discrete resource consumption. It is the purpose of this paper to present a formal model of the general nonpreemptive problem and to introduce a relatively simple yet effective procedure for optimally or heuristically solving it.

This paper thus discusses project scheduling problems comprised of jobs which once initiated, cannot be interrupted (preempted) by another job, and where resources

*Accepted by David G. Dannenbring; received January 20, 1980. This paper has been with the author 4 months for 2 revisions.

†University of Michigan.

consumed are discrete in nature. The model is more general than existing nonpreemptive approaches in that it permits each job to be accomplished in one of several different ways using a variety of resource types including per-period and project related resources. A solution to any of the project scheduling problems considered here is called a schedule. A schedule specifies *how* each job should be accomplished, that is, which resource-duration option should be selected, and *when* each job should begin and end. The solution procedures introduced derive schedules that are optimal or nearly optimal with respect to a given managerial objective.

Although a number of different objectives can be investigated using our approach, this paper focuses on two variations of the time-resource tradeoff problem: finding the schedule of jobs that minimizes project completion time, and determining the schedule of jobs that minimizes overall project costs. The first model was selected because it is the natural generalization of previous nonpreemptive resource-constrained project scheduling research [1], [4]–[6], [10], [11]. The resource-constrained time-cost tradeoff problem was selected because it bridges an important gap in the project management literature: the gap between discrete time-cost trade-off scheduling techniques and methods for nonpreemptive scheduling under resource constraints (this gap has been bridged for the preemptive case [7], [8]). Time-cost tradeoff methods have been based on the assumption that resources are available in unlimited quantities. Resource-constrained models, on the other hand, have not been developed to explicitly treat cost or profit as a scheduling objective while simultaneously permitting job durations to be affected by resource allocations. The proposed formulation and solution algorithm remove the above restriction, thus permitting resource-duration interations when monetary objectives are specified.

In the following section, the nature of the time-resource trade-off problem will be described and formally defined using a zero-one integer programming approach. Next, an implicit enumeration solution technique will be introduced for finding the schedule of jobs that minimizes project completion time. Modifications to this algorithm are subsequently described for solving the problem given monetary-related objective functions. Illustrations of these ideas follow, with solved examples for the objectives of minimizing project completion time and minimizing total project cost within the framework of a resource-constrained time-cost tradeoff problem. The last section of the paper presents computational results for a series of test problems.

2. Formulation of the Project Completion Time Minimization Problem

One can assume without loss of generality that a project can be depicted by an acyclic activity-on-node graph where activities, or jobs, are numerically labeled such that successor jobs always have higher numbers (labels) than all their predecessors. Associated with each job is a set of possible durations and the corresponding resource requirements which would permit the job to be completed in the stated durations. Each duration-resource combination is called a "job-operating mode" or simply a "mode." For example, it may be possible to complete job X in 5 days using a skilled person (mode 1), or in 7 days using an unskilled person (mode 2).

In keeping with the general nature of the model, three resource categories are included. Using the terminology introduced by Weglarz [14], and Slowinski [7], [8], these resources are defined as *renewable*, *nonrenewable* or *doubly constrained*. If resources are available in limited quantities each time period, the resources are considered renewable. An example of such a resource would be skilled labor: the number of skilled laborers available to work on the project each day is limited, although no constraint is placed on the number of days skilled labor may be used. Thus, the resource labor is renewed each period to a predetermined level, where the level may differ from one period to the next. If the total consumption of the resource over the life or part of the life of the project is constrained, it is called nonrenewable. Money is perhaps the best example of a nonrenewable resource: overall project costs are frequently limited to a fixed predetermined contract price. Also, cash flows within a project may be pegged to the attainment of milestones by particular dates. If the budgeted capital is not used before a specific time after which it is no longer available, it is a nonrenewable resource. Finally, resources are defined as doubly constrained if both their per-period and total availability are limited. Money is again probably the best example of a doubly constrained resource: cash flows per day as well as total cash expended on a project are often restricted.

These three resource categories are included in the model because a variety of time and resource-based objective functions, as well as constraint forms, may then be considered. This has been very ably demonstrated by Weglarz [15] and Slowinski [7], [8] for many different scheduling problems, but especially for the preemptive scheduling problem.

The general time-resource tradeoff problem can be formulated as an integer programming problem in which a zero-one integer variable $x_{jtm} = 1$ if job j operating in mode $m(1 \leqslant m \leqslant M_j)$ is assigned a completion time in period t; otherwise, $x_{jtm} = 0$. Jobs are labeled from 1 to N, with job N being the unique terminal task without successors. If such a job N does not naturally exist, then a dummy job N having one mode with zero duration and zero resource requirements is appended. Associated with each job j are its critical path determined early finish time, E_j, and late finish time, L_j. Both E_j and L_j are calculated in the usual way but using the set of minimum duration modes for all jobs. That is, on the forward pass through the network, early start and early finish (E_j) times are calculated, and on the reverse pass through the network late finish (L_j) and late start times are calculated (see [3, pp. 31–35] for details). In both passes the shortest duration mode for each job is used. Only the E_j and L_j are retained for future use, however. In determining the L_j, L_N is set equal to a known heuristic project completion time H, or, if H is not known, L_N is set equal to the sum of all maximum job durations.

Equations (1) to (5) define the problem when the objective is to minimize project completion time. Occurrence constraint set (2) insures that each job is completed exactly once. Constraint set (3) insures that precedence relationships are maintained. Here P is the set of all pairs of immediate predecessor jobs and d_{jm} is the duration of job j operating in mode m. Renewable resource restrictions are enforced by constraints (4). Resource k is available in R_{kt} units in time period t. Job j requires the use of r_{jkm} units of renewable resource k when operating in mode m. Nonrenewable resource limitations are imposed by (5). Here W_i is the amount of nonrenewable resource i available for the project, and w_{jim} is the amount of resource i consumed by job j in mode m. Doubly constrained resources would appear in both (4) and (5). For example, for a given job and mode we could set $r_{jkm} = w_{jim}/d_{jm}$ where k and i refer to the same resource type. The resource requirement r_{jkm} can now be considered the rate at which resource i is consumed by job j; thus both the rate and total usage of resource k are constrained.

$$\text{Minimize} \sum_{m=1}^{M_N} \sum_{t=E_N}^{L_N} t x_{Ntm} \tag{1}$$

$$\text{subject to} \sum_{m=1}^{M_j} \sum_{t=E_j}^{L_j} x_{jtm} = 1 \quad \text{for } j = 1, \ldots, N, \tag{2}$$

$$-\sum_{m=1}^{M_a} \sum_{t=E_a}^{L_a} t x_{atm} + \sum_{m=1}^{M_b} \sum_{t=E_b}^{L_b} (t - d_{bm}) x_{btm} \geqslant 0 \qquad \text{for all } (a,b) \in P, \tag{3}$$

$$\sum_{j=1}^{N} \sum_{m=1}^{M_j} \sum_{q=t}^{t+d_{jm}-1} r_{jkm} x_{jqm} \leq R_{kt}, \qquad k = 1, \ldots, K, t = 1, \ldots, H, \tag{4}$$

$$\sum_{j=1}^{N} \sum_{m=1}^{M_j} \sum_{t=E_j}^{L_j} w_{jim} x_{jtm} \leq W_i \qquad i = 1, \ldots, I. \tag{5}$$

3. Formulation of Problems with Monetary Objective Functions

The presented model permits one to consider several alternative objective functions in addition to simply minimizing project duration. In particular, least cost schedules can be obtained if the first term in (6) becomes the objective function, and (7) replaces (5). Given an arbitrarily large project completion time T, the global minimum cost solution can be found. Otherwise, the least cost solution will be obtained such that project completion time is less than or equal to the desired completion time T. Of course, the resource c in (6), which we are calling cost, may be any nonrenewable or doubly constrained resource.

$$\text{Min} \sum_{j=1}^{N} \sum_{m=1}^{M_j} \sum_{t=E_j}^{L_j} w_{jcm} x_{jtm} + \sum_{m=1}^{M_N} \sum_{t=E_N}^{L_N} S_t x_{Ntm}, \tag{6}$$

$$\sum_{m=1}^{M_N} \sum_{t=E_N}^{L_N} t x_{Ntm} \leq T. \tag{7}$$

By adding the second sum to (6) and dropping the constraint set (7), the problem becomes a resource-constrained version of the classical CPM time-cost tradeoff problem. Here S_t, which is usually approximated by a constant for all t, represents project overhead per period. By making S_t a function of time, however, performance penalties or bonuses may be considered explicitly.

If the project's time horizon is long, then the present value of cash flows rather than the absolute cost of the project may dominate in importance. In this case, the objective function would be to minimize the present value of expenditures of resource c as given by (8).

$$\text{Min} \sum_{j=1}^{N} \sum_{m=1}^{M_j} \sum_{t=E_j}^{L_j} v_{jtm} x_{jtm}. \tag{8}$$

Here, v_{jtm} is the present value of expenditures of resource c for job j operating in mode m if it finishes in time period t. Of course (8) can be replaced with a "maximize net present value " objective function if revenues are also taken into account.

Other objective functions and constraints can be incorporated into the model and solved using the proposed algorithm. The inclusion of lateness penalties and due dates or job performance constraints, such as concurrency requirements, are some of these possibilities. Since these variations have been considered in [6] and could easily be included in the algorithm, we will not repeat them here. It is of interest, however, to note that an important problem bridging the gap between the capital budgeting problem and the general time-resource tradeoff problem can be effectively formulated with our procedure. A brief comment on this situation is in order, because it is a frequently encountered problem that has not been adequately treated in the literature, and because it requires a new interpretation of what have been called nonrenewable resources.

To motivate this discussion, consider the general multiproject scheduling problem restricted by renewable, nonrenewable and doubly constrained resources. Assume that the objective is to schedule jobs in order to maximize the net present value of all projects, where *all* projects *must* be completed. Further assume that money is a nonrenewable or doubly constrained resource and that the completion of certain jobs (or projects) results in a *positive* cash flow. In this case, the performance of certain jobs increases or renews the nonrenewable resource money. This resource dependency between jobs poses no formulation problems: positive resource flows are simply indicated by negative coefficients for appropriate variables in (5) or in (8). However, the nomenclature "nonrenewable" somewhat disguises the broader interpretation of constraint set (5).

4. Solution Methodology for the Project Completion Time Minimization Problem

A two-stage solution methodology is developed which builds upon ideas presented earlier [11] for the basic resource-constrained scheduling problem. A specialized algorithm of this type was selected because of the inability of general purpose 0-1 computer codes to efficiently store and solve problems of the size we are considering. Stage one defines the problem as a compact integer programming problem, and stage two searches for the optimal solution using an implicit enumeration scheme that systematically improves upon generated heuristic solutions. Conceptually the problems are given by (1) to (8), but operationally no constraints or objective functions are explicitly formulated in our procedures. Rather, all resource and precedence data are stored in compact arrays that are interrogated during enumeration to insure solution feasibility. This results in a very efficient use of computer storage and also in a reduction of computational time.

Stage One

In stage one the network is first relabeled using one of the heuristic scheduling rules listed in Table 1. This process is similar to that successfully used in [11], although the rules are calculated differently. The effect of this relabeling procedure is to specify the order in which jobs will be considered for assignment (scheduled) during stage two of the algorithm. Resources and modes are also sorted for each job in stage one. Renewable resources are sorted such that the resource having the maximum frequency of highest per-period requirement relative to average resource availability has the smallest numerical label. Specifically, for each resource and job mode an index (9) is calculated.

$$\text{Index}_{jkm} = d_{jm} \text{ for the } k \text{ which maximizes } r_{jkm}/\bar{R}_k . \tag{9}$$

Otherwise, $\text{Index}_{jkm} = 0$, where

$$\bar{R}_k = \sum_{t=E_j}^{L_j} R_{kt}/(L_j - E_j + 1).$$

These indices are summed over all jobs and modes, then the resources are sorted in decreasing order of summed indices. The effect of this sort during the enumeration procedure is to identify resource infeasibility as early as possible. This sorting procedure can markedly reduce computational time when the number of renewable resources exceeds two.

Depending on the objective function selected, one of several mode sorts is used. For example, if the objective is to minimize project duration, modes are sorted by increasing duration. If the objective is to minimize project cost, then modes are sorted according to increasing total cost. Mode sorting procedures typically have a significantly far greater effect on computational time than resource sorting. The primary reason is that the order in which modes are sorted determines, along with job labeling, the initial heuristic project completion time found by the enumeration procedure, and hence, the initial bounds L_j.

Stage Two

Once a problem has been prepared for analysis, the algorithm passes to stage two. Here an implicit enumeration algorithm builds always-feasible partial solutions into complete schedules by considering jobs for assignment in increasing numerical order. (A "partial solution" or "partial schedule" indicates that fewer than N jobs have currently been feasibly scheduled.) The algorithm begins with an attempt to find the earliest feasible assignment for jobs 1, 2, 3, and so on, in order, until a job is identified that cannot be assigned without violating precedence or resource constraints, or without exceeding a bound such as its late finish time L_j. When such a job is identified, the algorithm backtracks to the most recently assigned job and attempts to reschedule it at a later time in order to free resources for use by the unassigned job. This process is *systematically* applied to all jobs and modes until an improved solution is found or until optimality is verified. If an improved solution is found, bounds are correspondingly tightened and the assignment procedure begins again with job 1. Ultimately, optimality is verified in one of two ways. Either all possible job assignments have been explicitly or implictly evaluated, or a solution is found which equals a proven theoretical bound such as the critical path.

The important role played by the job relabeling scheme in stage one of the algorithm is to specify the order in which jobs are considered for assignment in stage 2. Since jobs have been renumbered using a priority dispatch scheduling rule (selected from Table 1), the effect of considering jobs for assignment in increasing numerical order is simply to *attempt to* generate the corresponding heuristic solution to the problem. When stage 2 begins, in fact, it is usually more than just an attempt. If the initial upper bounds are so loose (due to L_N being set equal to the sum of all maximum job durations) as to not cause any backtracking, then the first solution to the problem *is* the heuristic solution derived from applying the priority dispatch scheduling rule. Thereafter, however, as the bounds become successively tighter following the identification of improved solutions, the numbering scheme forces the algorithm to always *attempt to* find the corresponding heuristic solution for all currently *un*assigned jobs. But this may be just an attempt: the now tighter bounds may prevent the realization of the heuristic solution. In other words, the enumeration process attempts to complete the existing partial solution of scheduled jobs with the heuristic solution of unscheduled jobs. If this is not possible, backtracking occurs. Thus, the selection of a relabeling rule is quite important. It determines the order in which jobs are considered for assignment, which in turn significantly affects the efficiency of the algorithm. Computational results later in the paper illustrate this point.

Now, a more detailed discussion of the foregoing general description will be given. We begin by considering the enumeration procedure for the objective of minimizing project completion time as given by (1) to (5). Modifications required in this procedure for other objective functions will follow.

Associated with each job j are integer variables Y_j and U_j, where $Y_j = t$ and $U_j = m$, if mode m of job j is assigned a completion time of t. Otherwise, Y_j and U_j are both unspecified (since partial schedules are always feasible). Enumeration begins by assigning mode 1 of job 1 to its earliest possible completion time. Renewable resources required, r_{1k1}, are subtracted from resources available, R_{kt}, for $k = 1, \ldots, K$ and $t = 1, \ldots, Y_1$. Similarly, nonrenewable resources consumed, w_{1im}, are subtracted from W_i for $i = 1, \ldots, I$.

Beginning with job 2 and continuing to job N, an attempt is made to assign the first mode of each job to its earliest precedent and resource-feasible completion time. Precedence and resource constraints are maintained by first finding $t^* = \max\{Y_p \mid p \in P_j^*\}$, where P_j^* is the set of immediate predecessors of j. To insure that nonrenewable resources are not overconsumed, availability array W_t is interrogated to determine whether $w_{jim} \leq W_i$. Then the resource availability array R_{kt} is scanned for $k = 1, \ldots, K$ and from $t^* + 1$ to L_j for the earliest contiguous interval d_{jm} units in length where $r_{jkm} \leq R_{kt}$. Suppose that interval begins at $t^* + \Delta$; job j then ends at $t' = t^* + \Delta + d_{jm} - 1$. Consequently, we would set $Y_j = t'$, $U_j = m$, and adjust resource availability arrays accordingly. R_{kt} would be reduced r_{jkm} units from $t = t^* + \Delta$ to t' for $k = 1, \ldots, K$, and W_i would be reduced by w_{jim} for $i = 1, \ldots, I$.

If the assignment of mode m of job j is prevented because of either type of resource infeasibility, modes $m + 1, m + 2, \ldots, M_j$ are considered for assignment. If none of the modes of j is resource feasible, then the algorithm backtracks to job $j - 1$. Mode $m = U_{j-1}$ of job $j - 1$ is removed from solution. Array R_{kt} is increased by r_{j-1km} units for $k = 1, \ldots, K$ and $t = t'', t'' + 1, \ldots, Y_{j-1'}$, where $t'' = Y_{j-1} + 1 - d_{j-1m}$. W_t is increased by w_{j-1im} for $i = 1, \ldots, I$.

An attempt is now made to reassign job $j - 1$ mode m to the earliest feasible completion time *after* Y_{j-1}. That is, R_{kt} is scanned from $t = t'' + 1$ to L_{j-1} for the earliest contiguous interval d_{j-1m} units in length where $r_{j-1km} \leq R_{kt}$, for $k = 1, \ldots, K$. If such an interval is found, then Y_{j-1} is set equal to the new completion time, resource arrays are adjusted, and so on. If no such interval is found, an attempt is made to assign mode $m + 1$, $m + 2$, etc. This process of assignment and backtracking continues until job N is assigned a completion time, or until an attempt is made to backtrack below job 1. In the former case an improved (reduced) completion time for the project is found. In the latter, the incumbent best solution is optimal.

When an improved solution with a reduced project completion time $T^* = Y_N$ is found, it replaces the incumbent solution (if any). If Y_N equals a known lower bound on the solution, such as the critical path early completion time E_N, then Y contains the optimal schedule. If Y_N exceeds the best known bound, the late finish times L_j are reduced by the quantity $L_N - (Y_N - 1)$ and the augmentation process starts anew with job 1 mode 1. Again, optimality is assured when either an attempt is made to backtrack below job 1, or a project completion time is found that is equal to a known lower bound.

5. Algorithm Modifications Needed for Problems with Cost Related Objective Functions

The above discussion has been restricted to the case in which (1) is the objective function. If (6) or (8) were the objective function and constraints (7) were appended to the model, the above algorithm could still be used after incorporating the following minor modifications. In stage one during the calculation of L_j, the desired project completion time T replaces H, a known heuristic completion time for the project. In stage two, several changes in the solution procedure are made. First, the late finish times L_j are *not* updated when a new completion time is found for job N. But the cost C of the incumbent solution (if one exists) is replaced by the cost, as represented by (6) or (8), of the improved solution. The incumbent cost $\hat{C}$ then becomes a new upper bound on the solution. This upper bound is used in two ways. During augmentation, when mode m of job j is about to be evaluated for resource feasibility, a check is made to insure that the actual cost of the current partial solution for jobs 1 to $j - 1$ plus the cost of job j does not exceed C. Unfortunately, this is a weak test that guarantees feasibility but it does not exclude the explicit evaluation of many other nearly good solutions. Given the strong ordering scheme used for augmenting variables in the algorithm, it is possible to construct the following tighter cost bounds.

Define a vector D with elements D_j given by (10). D_j is thus

$$D_j = \sum_{g=j}^{N} \min\{ w_{gcm} \mid m = 1, \ldots, M_g \} \tag{10}$$

the cumulative minimum cost of all unassigned jobs when job j is being considered for assignment, where the nonrenewable resource c is cost. The cost bound is implemented before precedence restrictions are evaluated. If the actual cost of the partial solution of jobs 1 to $j-1$ plus D_j is greater than or equal to C, then backtracking to $j-1$ commences immediately.

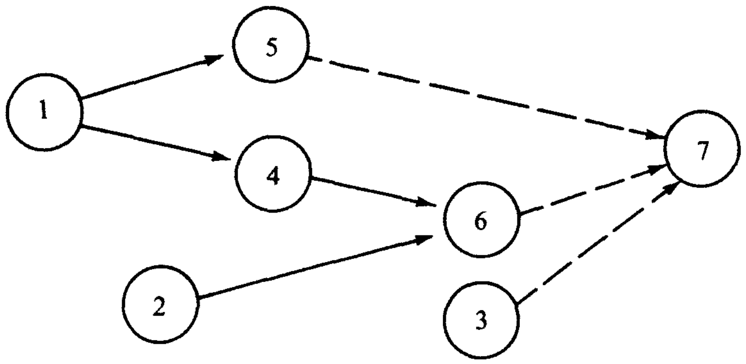
6. General Observations on Computational Strategies

In this paper as in [11], the general approach to the minimum project completion time problem is to start with a heuristic solution and to improve upon it systematically until the optimum is found. The attractiveness of this strategy is that the procedure may be stopped before reaching optimality when the tradeoff between computational time and potential solution improvement becomes unfavorable, yet a good heuristic solution to the problem will still be available. Very often, however, this is not the favored approach from a strictly computational point of view when an optimal solution is desired. If the initial heuristic solution greatly exceeds the optimal solution, then a fair amount of time may be expended in generating improved but nonoptimal solutions. An alternative strategy in this case, found to be useful in a zero-one approach to minimum duration project scheduling [4], is to search over a varying completion time horizon. Although it has not been implemented, the proposed algorithm could easily be modified to accommodate this approach. Regardless of the horizon strategy used, however, a successful application of this technique relies very heavily on heuristics—heuristics to obtain a good starting solution for bounding purposes (e.g., calculating L_j) and good heuristics to use in mode relabeling and resource and mode-sorting schemes. The computational section will report on results using the heuristics listed in Table 1.

7. Solved Examples

The following example, adopted from Elmaghraby [2], illustrates the foregoing discussion. In Figure 1, the project is shown to consist of six jobs, each of which may be accomplished in one of two modes. Five renewable resources are required by each job. Resource one represents a higher level of skill than resource two, which is reflected in reduced durations for those modes using resource one. The resources available in each period of the project's duration are 1, 2, 6 and 8 for resources one, two, three and four respectively. Resource five will be discussed momentarily. The scheduling problem Elmaghraby considers is to find that combination of modes and job starting times which will minimize project completion time given these four resources restrictions. This problem is specified by (1) to (4), and the optimal solution is given in Figure 2.

In order to illustrate the notion of renewable and doubly constrained resources, we have modified Elmaghraby's problem by appending to each mode its per period and total costs of operation. Assuming that the resources are valued at \$100, \$30, \$10 and \$15 per period, the period cost of job 1 mode 1, for example, would be \$135 = [1(100) + 0(30) + 2(10) + 1(15)], and its total cost would be \$270 = 2(135). If only



Job (j)	Mode (m)	Duration (d_{jm})	Renewable Resource Requirements(r_{jkm}) 1	2	3	4	5 (k)	Total Cost (w_{j1m})
1	1	2	1	0	2	1	\$135	\$270
	2	3	0	1	2	1	65	195
2	1	1	1	0	3	2	160	160
	2	3	0	1	3	2	90	270
3	1	3	1	0	1	4	170	510
	2	4	0	1	1	4	100	400
4	1	5	1	0	1	3	155	775
	2	7	0	1	1	3	85	595
5	1	4	1	0	2	2	150	600
	2	6	0	1	2	2	80	480
6	1	1	1	0	3	4	190	190
	2	4	0	1	3	4	120	480
7	1	0	0	0	0	0	0	0
Resources Available Each Period (R_{kt})			1	2	6	8	\$300	

FIGURE 1. A Six-Job Project.

total cost is constrained, cost would be a nonrenewable resource. If resource five, cost per period, and total cost are constrained, then cost becomes a doubly constrained resource.

Figure 3 illustrates the minimum duration solution to the sample problem after restrictions have been placed on per-period and total cash expenditures. This problem is depicted by (1) to (5) where maximum per period cash flow $R_{5t} = \$300$, and maximum project cost $W_1 = \$2,020$.

Figure 4 shows the minimum cost solution to the sample problem represented by (2) to (6), where the latest acceptable project completion time is $T = 10$, and the maximum per period cash flow $R_{5t} = \$300$. Here, $S_t = 0$ for all t. Perhaps the most interesting case,.however, is when $S_t \neq 0$, since this casts the problem as a resource-constrained project scheduling time-cost tradeoff problem. For example, if the per period overhead cost of the sample problem is $S_t = \$250$ for all t, the overall minimum cost schedule is given in Figure 4. This can be verified quite readily by noticing that if the project were extended one period as given in Figure 3, then the marginal cost of overhead, \$250, would exceed the marginal reduction in scheduling costs, \$180 = 2200–2020.

```
              Job Schedule
Job Mode
 1    2     XXX
 2    2     XXX
 3    1     XXX
 4    1        XXXXX
 5    2        XXXXXX
 6    1             X
            123456789
               Time
```

Resource Profile

Time Period	Resources Consumed 1	2	3	4
1	1	2	6	7
2	1	2	6	7
3	1	2	6	7
4	1	1	3	5
5	1	1	3	5
6	1	1	3	5
7	1	1	3	5
8	1	1	3	5
9	1	1	5	6

FIGURE 2. Minimum Duration Solution to Sample Problem

```
                  Job Schedule
                                     Job
Job Mode                             Cost
 1    2     XXX                      $195
 2    1         X                     160
 3    2     XXXX                      400
 4    2        XXXXXXX                595
 5    2          XXXXXX               480
 6    1               X               190
            12345678901             $2020
               Time
```

Resource Profile

Time Period	Resources Consumed 1	2	3	4	5 (Cash Flow)
1	0	2	3	5	165
2	0	2	3	5	165
3	0	2	3	5	165
4	0	2	2	7	185
5	1	1	4	5	245
6	0	2	3	5	165
7	0	2	3	5	165
8	0	2	3	5	165
9	0	2	3	5	165
10	0	2	3	5	165
11	1	1	5	6	270

FIGURE 3. Minimum Duration and Minimum Cost Solution to Sample Problem with Money a Doubly Constrained Resource

Job Schedule

```
                                       Job
Job   Mode                             Cost
 1     2    | XXX                   |  $195
 2     1    |                   X   |   160
 3     2    | XXXX                  |   400
 4     1    |     XXXXX             |   775
 5     2    |       XXXX  XX        |   480
 6     1    |                    X  |   190
               1 2 3 4 5 6 7 8 9 0    $2200
                      Time
```

Resource Profile

Time Period	Resources Consumed 1	2	3	4	5 (Cash Flow)
1	0	2	3	5	165
2	0	2	3	5	165
3	0	2	3	5	165
4	1	1	2	7	255
5	1	1	3	5	235
6	1	1	3	5	235
7	1	1	3	5	235
8	1	1	3	5	235
9	1	1	5	4	240
10	1	1	5	6	270

FIGURE 4. Minimum Duration Solution to Sample Problem with Project Completion Time Less Than 11

8. Computational Results

To test the efficacy of this approach the enumeration procedure which solves the minimum project completion time problem was programmed in FORTRAN. This basic program was then modified, as described previously, to solve problems with cost-related objective functions. No special programming effort was exercised to optimize either the storage of data or program-running efficiency, since the basic approach already does a reasonably good job in both respects. Although for testing purposes two separate programs were developed (the basic and the modified program) with very little effort one program could be written to solve all the various problems discussed in this paper. Such a program could easily be written to run in less than 32k Bytes of memory for problems containing approximately thirty jobs with three modes each, and six renewable and ten nonrenewable resources over a planning horizon of 100 time periods. This efficient use of computer storage permits the approach to be adopted by potential users with only modest computer hardware.

A preliminary evaluation of the heuristic rules shown in Table 1 was accomplished by solving 100 10-job problems as project completion time minimization problems with each heuristic used as a node-numbering rule. Detailed descriptions of these problems will not be given here since they are completely listed in Talbot [12]. Briefly, jobs in each problem had from one to three modes of operation and each mode was constrained by three renewable resources. The total number of modes in each project ranged from eighteen to thirty. Network complexity (the ratio of the number of

TABLE 1

Node Relabeling Rules

Rule	Formula	Description
1. MAX $ADUR_j$	$\sum_{m=1}^{M_j} d_{jm}/M_j$	maximum average job duration
2. MAX DUR_j	$\mathrm{Max}\{d_{jm} \mid m = 1, \ldots, M_j\}$	maximum job duration
3. MIN L_j	L_j	minimum late finish time
4. MAX RD_j	$\left(\sum_{m=1}^{M_j} d_{jm}/M_j\right)\left(\sum_{m=1}^{M_j} \sum_{k=1}^{K} r_{jkm}/(M_jK)\right)$	maximum average resource demand
5. MIN E_j	E_j	minimum early finish time
6. MIN$(L-D)_j$	$\mathrm{Min}[L_j - \min\{d_{jm} \mid m = 1, \ldots, M_j\}]$	minimum (late finish time reduced by smallest duration)
7. RAN$_j$	–	nodes renumbered randomly
8. MIN$(L-D)_j$	$\mathrm{Min}\left\{L_j - \sum_{m=1}^{M_j} d_{jm}/M_j\right\}$	minimum (late finish time reduced by average duration)

precedence relationships to the number of jobs) ranged from 1.0 to 1.6. Critical path early finish times E_N ranged from 10 to 27, and optimal project completion times ranged from 12 to 32.

Table 2 contains five summary measures of heuristic performance for the minimum duration problem. "Best" refers to the smallest project completion time found by all heuristics. "Equal to optimal" is the percentage of problems for which the heuristic solution equaled the optimal solution. "Worst" is the percentage of problems for which the rule yielded the largest project completion time. Measures *d* and *e* refer to the percentage of problems for which the enumeration algorithm found and verified the optimal solution when the jobs were numbered with the respective heuristics.

Certainly no sweeping generalities can be inferred from an analysis of this small sample. It is interesting to note, however, that all rules outperformed the random rule number 7 on all measures except total time to optimality. Furthermore, the best overall rules, numbers 3, 6 and 8, are based on the minimum late finish time rule which has been found to be one of the best rules for the single-mode problem [11]. The minimum

TABLE 2

Heuristic Solution Results for Minimum Duration Problems

	Rule*							
Percentage of Problems on which Rule Was	1	2	3	4	5	6	7	8
a. Best	72	73	83	73	76	77	57	76
b. Equal to optimal	30	29	33	30	30	34	23	34
c. Worst	60	59	46	58	53	54	77	54
d. Fastest to optimal	37	41	52	49	37	58	37	55
e. Slowest to optimal	37	39	28	40	39	28	59	29

*Described in Table 1

average time to find and verify the optimal solution for all 100 problems was obtained with rule 8 at 0.392 seconds of CPU time for each problem on an AMDAHL 470 V/7 computer. The range of solution times for each problem using rule 8 was from 0.001 to 12.912 seconds with only two problems requiring more than 2 seconds.

To evaluate the effectiveness of this approach on larger problems and with cost-related objective functions, ten 20-job and ten 30-job problems were solved under a variety of resource and cost conditions. It was not always possible to solve these problems optimally within 16 seconds of CPU time, which was the maximum time the program was permitted to run for a given problem. However, in all cases good heuristic solutions were found within two seconds by at least one of the eight renumbering heuristics listed in Table 1 when appropriate mode sorting was also used.

9. Summary and Conclusions

This paper has presented a computationally tractable integer programming approach for solving a large class of nonpreemptive resource-constrained project scheduling problems in which job performance time is a function of resource allocation. The approach is general enough to permit the solution of problems with time or monetary objective functions under a variety of resource restrictions. In particular, the paper introduces and discusses in detail the only solution algorithms thus far reported in the open literature for solving the nonpreemptive project completion time minimization problem and the discrete time-cost tradeoff problem with multiple resource constraints. Computational experience is reported for a series of test problems which indicates that the methodology can provide optimal solutions to small problems and good heuristic solutions to larger problems in a cost-effective manner.[1]

[1]This work was supported by a grant from the Graduate School of Business Administration, the University of Michigan, during the Summer of 1979. The author wishes to thank Professor James Patterson, the University of Missouri-Columbia, for reading an early draft of this paper and for providing a number of valuable suggestions. Appreciation is extended to Professors Roman Slowinski and Jan Weglarz, both of the Technical University of Poznan, Poland, who introduced the author to new approaches for preemptive scheduling which in many ways influenced the current work. The author also wishes to acknowledge the constructive comments provided by the referees.

References

1. Davis, E. W. and Heidorn, G. E., "An Algorithm for Optimal Scheduling Under Multiple Resource Constraints," *Management Sci.*, Vol. 17, No. 12 (August 1971), pp. B-803–816.
2. Elmaghraby, S. E., *Activity Networks: Project Planning and Control by Network Models*, Wiley, New York, 1977.
3. Levy, J. and Wiest, J., *A Management Guide to PERT/CPM*, 2nd ed., Prentice-Hall, Englewood Cliffs, N.J., 1977.
4. Patterson, J. H. and Huber, D., "A Horizon-Varying, Zero-One Approach to Project Scheduling," *Management Sci.*, Vol. 20 No. 6 (February 1974), pp. 990–998.
5. ——— and Roth, G. W., "Scheduling a Project Under Multiple Resource Constraints: A Zero-One Programming Approach," *AIIE Trans.* Vol. 8, No. 4 (December 1976), pp. 449–455.
6. Pritsker, A. B., Watters, L. J. and Wolfe, P. M., "Multiproject Scheduling with Limited Resources: A Zero-One Programming Approach," *Management Sci.*, Vol. 16, No. 1 (September 1969), pp. 93–108.
7. Slowinski, R., "Two Approaches to Problems of Resource Allocation Among Project Activities—A Comparative Study," *J. Operational Res. Soc.*, Vol. 31, No. 8 (August 1980), pp. 711–723.
8. ———, "Multiobjective Network Scheduling with Efficient Use of Renewable and Non-renewable Resources," *European J. Operational Res.*, Vol. 7, No. 3 (July 1981), pp. 265–273.
9. ——— and Weglarz, J., "Solving the General Project Scheduling Problem with Multiple Constrained Resources by Mathematical Programming," Proc. 8th IFIP Conf. on Optimization Techniques, *Lecture Notes in Control and Information Sciences*, Vol. 7, Springer-Verlag, Berlin, 1978, pp. 278–289.

10. STINSON, J., "A Branch and Bound Algorithm for a General Class of Resource-Constrained Scheduling Problems," *AIIE Conf. Proc.*, Las Vegas, Nevada, 1975, pp. 337–342.
11. TALBOT, F. B. AND PATTERSON, J. H., "An Efficient Integer Programming Algorithm With Network Cuts For Solving Resource-Constrained Scheduling Problems," *Management Sci.*, Vol. 24, No. 11 (July 1978), pp. 1163–1174.
12. ———, "Project Scheduling With Resource-Duration Interactions: The Nonpreemptive Case," *Working Paper*, The Graduate School of Business Administration, The University of Michigan, January, 1980.
13. WEGLARZ, J., BLAZEWICZ, J., CELLARY, W. AND SLOWINSKI, R., "An Automatic Revised Simplex Method for Constrained Resource Network Scheduling," *ACM Trans. Math. Software*, Vol. 3, No. 3 (September 1977), pp. 295–300.
14. ———, "New Models and Procedures for Resource Allocation Problems," *Proc. 6 INTERNET Congress* 2, VDI-Verlag, Dusseldoof, 1979, pp. 521–530.
15. ———, "Control in Resource Allocation Systems," *Foundations Control Engineering*, Vol. 5, No. 3 (September 1980), pp. 159–180.