A STRONG FRACTIONAL CUTTING-PLANE ALGORITHM FOR RESOURCE-CONSTRAINED PROJECT SCHEDULING

Jayaram K. Sankaran

Senior Lecturer, MSIS Department
University of Auckland
Private Bag 92019
Auckland, New Zealand.
E-mail: j.sankaran@auckland.ac.nz

Dennis L. Bricker

Associate Professor, Department of Industrial Engineering
The University of Iowa
4110 EB, Iowa City, IA 52242, USA.
E-mail: dennis-bricker@uiowa.edu

Shuw-Hwey Juang

Adjunct Assistant Professor, Department of Industrial Technology
The University of Southwestern Louisiana
229 Rougeou Hall, Lafayette, LA 70504, USA.
E-mail: juang@usl.edu

Abstract: We develop and test a strong fractional cutting-plane algorithm for the classical non-preemptive precedence- and resource-constrained project scheduling problem. While our basic approach is to formulate the problem as a 0-1 IP and solve it by LP-based branch-and-bound, we enhance the algorithm considerably through (a) an improved IP reformulation, (b) problem preprocessing techniques, and (c) on-the-fly tightening of the LP relaxation by generating strong and valid inequalities that are violated by the current (fractional) LP optimum. We also report results from an exploratory computational implementation of the algorithm.

Significance: Activities of a PERT/CPM project may require sharing of one or more resources (e.g., labor). An algorithm is described for finding the schedule which minimizes the completion time or *makespan* of the project.

Keywords: Project management: resource-constrained project scheduling. Integer programming: fractional cutting-plane algorithms.

Received "leave blank" Accepted "leave blank&quot

1. INTRODUCTION

The subject of this paper is a strong, FCPA (fractional cutting-plane algorithm) for resource-constrained project scheduling. The precise problem is as follows. We are given a set of activities, each of which lasts an integral number of time periods. Once commenced, an activity continues without interruption until its completion, ie., pre-emption is not allowed. Throughout its duration, each activity consumes, at a specified rate, each of several resources whose availabilities during each time period are known. An activity cannot commence until all of its predecessors have been completed and in every time period, the consumption of each resource should not exceed the availability. The objective is to minimize the completion time or *makespan* of the project.

Both exact algorithms and heuristics have been developed for this problem; since the present paper focuses on developing an exact algorithm, we confine our review of the literature to algorithms. Further, for reasons of space, our review is selective; we refer the reader to Ozdamar and Ulusoy (1995) for a more comprehensive review.

Fisher (1973) developed an algorithm for the resource-constrained project scheduling problem using a Lagrangian relaxation in which the resource constraints are relaxed. Although the Lagrangian subproblem is the dual of a network flow problem (as we show in Section 2), Fisher solved it by implicit enumeration. His relaxation yielded tight lower bounds on the instances of job shop scheduling for which he reported computational results.

Patterson and Huber (1974) described an approach in which the upper bound on project makespan is successively increased until a feasible solution is found. They also presented a maximum-bound approach in which they initially find an incumbent heuristically and then solve a sequence of 0-1 IPs, in each of which the makespan is bounded from above.

Stinson, Davis, and Khumawala (1978) described a branch-and-bound scheme which adapts the dominance rules of Schrage (1970), and employs a novel method for determining lower bounds on project makespan for partial schedules.

Christofides, Alvarez-Valdes, and Tamarit (1987) described a branch-and-bound algorithm for the problem and examined four different bounding strategies. One of these was the solution of the LP relaxation augmented by cutting-planes. Another was Lagrangian relaxation of the resource constraints. (Like Fisher, the authors use a non-polynomial but efficient procedure for solving the Lagrangian relaxation.) The algorithm uses disjunctive arcs to resolve conflicts that are created whenever it is necessary to schedule sets of activities whose total resource requirements exceed the resource availabilities in some periods.

Demeulemeester and Herroelen (1997) have also described an efficient enumerative algorithm for the present problem, in which branches that emanate from a parent node of the enumeration tree correspond to exhaustive and minimal combinations of activities, the delay of which resolves resource conflicts at the parent node.

Since the early 1980s, several researchers have successfully used strong FCPAs to solve many large pure and mixed IPs, in a variety of applications. These algorithms are replete with techniques for coefficient reduction, variable elimination, and on-the-fly generation of cutting-planes - the Lanchester prize-winning work of Crowder, Johnson, and Padberg (1983) for large and sparse 0-1 IPs is a prime example. However, a recent survey by Ozdamar and Ulusoy (1995) indicates that, for the resource-constrained project scheduling problem, no similar strong FCPA has been reported.

In this paper, we describe the development and testing of a strong FCPA for the project scheduling problem, with techniques for coefficient reduction, variable elimination, on-the-fly generation of cutting-planes, etc. We use LP-based branch-and-bound to solve the associated 0-1 IP. This is enhanced by us through: (1) an improved IP reformulation of the problem; (2) problem preprocessing techniques; and (3) on-the-fly tightening of the LP relaxation.

The general approach that is adopted in many of these strong FCPAs is to first employ a battery of techniques to preprocess the problem of concern in order to tighten its LP relaxation. Then, in contrast to the classical cutting-plane approach of Gomory (1963), these algorithms employ procedures for detecting valid inequalities for the convex hull of feasible solutions that are "strong" and are violated by the current fractional LP optimum. When it is no longer possible to identify additional valid violated inequalities, the algorithms resort to branch-and-bound.

In our approach for the resource-constrained project scheduling problem, we first improve upon the conventional, 'aggregate' 0-1 IP formulation by disaggregating the precedence constraints. It is well known (especially regarding facility location problems) that, although disaggregation increases the size of the problem, in the context of LP-based branch-and-bound, it generally hastens problem solution by reducing the duality gap. In fact, we show that, when the resource constraints are relaxed, the resulting constraint matrix in the disaggregate formulation of the project scheduling problem is totally unimodular, a property not enjoyed by the aggregate formulation. The importance of this property is, of course, that if the data is integral (as in our case), we can be assured that the LP solution of the relaxation is also integral!

Besides providing an improved reformulation, we enhance the basic LP-based branch-and-bound algorithm through several problem preprocessing techniques in order to tighten the LP relaxation. We first apply a heuristic algorithm to the problem in order to obtain an incumbent and, in the manner of Patterson and Huber (1974), set up the IP so that every feasible solution (if any) is strictly better than the incumbent. Then we deduce lower and upper bounds on activity completion times based on precedence and resource considerations. Following this, we attempt to (i) determine whether the IP is infeasible, (ii) fix variables at zero, and (iii) identify redundant constraints. If we cannot determine that the IP is

infeasible, we try to reduce any coefficients of variables in those resource constraints that cannot be identified as redundant.

The third enhancement of the basic approach is on-the-fly generation of valid inequalities that are violated by the current LP fractional optimum. As a consequence of the aforestated result about total unimodularity of the disaggregated precedence constraint matrix, one or more resource constraints must necessarily be invoked for generating valid violated inequalities. These inequalities are of two kinds: 'minimal cover' inequalities that are derived from individual 'knapsack' resource constraints coupled with precedence constraints, and 'clique' inequalities that are analogous to the well-known clique inequalities for the set packing and node packing problems (see Padberg, 1973; Nemhauser and Sigismondi, 1992).

The paper is laid out as follows. In Section 2, we introduce the notation of the paper and present both the aggregate and disaggregate formulations. In Section 3, we briefly discuss our preprocessing techniques. In Section 4, we describe our procedures for generating valid inequalities that are violated by the current fractional LP optimum. In Section 5, we present the complete algorithm and discuss computational results. In Section 6, we offer our conclusions. The interested reader is referred to Sankaran, Bricker, and Juang (1998) for a comprehensive description of our study.

2. NOTATION AND PROBLEM FORMULATION

We assume that the activities and their precedence relationships are represented as an AON (Activities-On-Nodes) network. In the network, there is a node for each activity and for each pair of activities m and n such that m must *immediately* precede n (i.e., with no required intervening activity), there is an arc from node m to node n, which we denote by "$m \rightarrow n$". (For the remaining part of the paper, unless otherwise stated, when we say that one activity precedes another, we do *not* presume that the one immediately precedes the other.)

We now define some of our notation.

N: The number of activities, which in turn are numbered from 1 to N. We also define fictitious activities, numbered 0 and N+1, which respectively denote the starting and finishing of the project. One can always number the activities so that if m precedes n, then $m < n$; we assume the node numbers possess this property.

K: The number of resources.

g: A valid upper bound, minus 1, on the minimum project makespan, which might be determined by a heuristic algorithm. (The ensuing IP formulation will include as feasible only schedules that are strictly better than the incumbent, i.e., whose makespans are g or less.)

A(k,t): The (integral) availability of resource k in period t, for all k = 1,...,K and t = 1,...,g.

r(j,k): The rate of consumption of resource k by activity j (assumed constant throughout the duration of the activity), for all j = 1,...,N and k = 1,...,K.

J(k): The set of activities which consume positive amounts of resource k, i.e., $\{j \in \{1,...,N\}: r(j,k) > 0\}$.

D(m,n): The length of the longest path between activities m and n in the precedence network, defined for every pair m and $n \in \{0,...,N+1\}$ such that m precedes n. Thus, D(m,n) is the precedence-based lower bound on the time lapse between the completions of m and n. Dijkstra's algorithm (Nemhauser and Wolsey, 1988, p. 56) may be used to compute $D(\cdot,\cdot)$.

The following parameters pertain to a generic activity j (j = 0,...,N+1).

d(j): The duration of activity j (assumed integral). (We set d(0) = d(N+1) = 0.) For each m such that $m \rightarrow j$, d(j) is also referred to as the 'length' of the arc (m,j).

EFT(j): The earliest finish time of activity j as obtained from the precedence relations alone. We arbitrarily set EFT(0) = 0.

LFT(j): The latest finish time of activity j as obtained from the precedence relations alone, assuming LFT(N+1) = g.

e(j): A current valid lower bound on the finish time of j. Thus, for all j, $e(j) \geq EFT(j)$.

L(j): A current valid upper bound on the finish time of j. Thus, for all j, $L(j) \leq LFT(j)$.

We initially set e(j)=EFT(j) and L(j)=LFT(j). Both bounds e(j) and L(j) may be tightened during the preprocessing phase. Without loss of generality, we assume that both e(j) and L(j) are integral, and for all m, $n \in \{0,...,N+1\}$ such that $m \rightarrow n$, $e(\cdot)$ and $L(\cdot)$ satisfy

$$e(m) \leq e(n) - d(n) \text{ and } L(m) \leq L(n) - d(n). \quad \ldots \quad (1)$$

We now define the decision variables.

x(j,t) = 1 if activity j (j = 0,...,N+1) is completed at time t, otherwise x(j,t) = 0. Because d(j) is assumed to be integral, we define x(j,t) only for integral values, namely values of t in the interval [e(j),L(j)].

The problem of scheduling the activities of the project so as to minimize the makespan of the project (subject to precedence and resource constraints) may be formulated as:

(P1:) $$Min \sum_{t=e(N+1)}^{L(N+1)} t.x(N+1,t) \quad \ldots \quad (2)$$

subject to

$$\sum_{t=e(j)}^{L(j)} x(j,t) = 1 \quad \text{for j = 0, ..., N+1.} \quad \ldots \quad (3)$$

For m, n ∈ {0, ..., N+1} such that m → n and t = e(n)-d(n), ..., L(m)-1,

$$\sum_{s=t+1}^{L(m)} x(m,s) + \sum_{s=e(n)}^{t+d(n)} x(n,s) \leq 1. \quad \ldots \quad (4)$$

For k = 1,...,K and t = 1,...,g,

$$\sum_{j=1}^{N} r(j,k).\left(\sum_{s=Max(e(j),t)}^{Min(t+d(j)-1,L(j))} x(j,s) \right) \leq A(k,t). \quad \ldots \quad (5)$$

x(j,t) ∈ {0, 1} ∀ t = e(j), ..., L(j), ∀ j = 0, ..., N+1.

The objective function, viz. (2), denotes minimization of the project makespan. The "convexity" or "generalized upper bound (GUB)" constraints, viz. (3), together with the integral constraints on x, ensure that for each j, activity j is completed in exactly one of the periods e(j), ..., L(j). The precedence constraints, viz. (4), ensure that for each pair m and n such that activity m must precede activity n, the start time of activity n is later than the finish time of activity m. For each such pair m and n, and each t such that e(n)-d(n) ≤ t ≤ L(m)-1, the corresponding constraint in (4) stipulates that if activity m finishes in period t+1 or later (i.e., if $\sum_{s=t+1}^{L(m)} x(m,s) = 1$), then activity n starts in period t+2 or later.

The resource constraints, viz. (5), ensure that the usage of a resource never exceeds its availability. Note that if activity j is ongoing in period t, then the finish time of the activity is at least t, and cannot exceed t+d(j)-1; hence the lower and upper limits of the summation in (5).

An important property of the formulation, P1, is that if the resource constraints are relaxed, then every basic feasible solution to the remaining set of constraints is 0-1. This fact is proved formally in Theorem 1.

THEOREM 1: Every basic feasible solution to (3)-(4) (plus non-negativity) is 0-1.

PROOF: To facilitate the proof, we make the non-singular transformation of the variables {x(j,t)} to variables {y(j,t)} as shown below.

For j = 0, ..., N+1 and t = e(j) ,..., L(j), let

$$y(j,t) = \sum_{s=e(j)}^{t} x(j,t) \quad \ldots \quad (6)$$

Thus, y(j,t) is 1 if activity j is completed by the end of period t (ie., if j finishes in period t or earlier); it is 0 otherwise. Then, (3) transforms to

$$y(j,L(j)) = 1 \ \forall \ j = 0, ..., N+1. \quad \ldots \quad (7)$$

Further, (4) transforms to

$$y(m,t) - y(n,t+d(n)) \geq 0 \quad \ldots \quad (8)$$

for m, n ∈ {0,...,N+1} such that m → n and t = e(n)-d(n), ..., L(m)-1.

Finally, the non-negativity of {x(j,t)} transforms to

$$y(j,t) - y(j,t-1) \geq 0 \text{ for } t = e(j)+1, ..., L(j), j = 0, ..., N+1, \text{ and}$$
$$y(j,e(j)) \geq 0. \quad \ldots \quad (9)$$

The constraint matrix corresponding to (7)-(9) is totally unimodular since the total unimodularity of the transpose follows readily from the result of Hoffman and Kruskal (see Nemhauser and Wolsey, 1988, Theorem 2.5, p. 542). Thus, every basic feasible solution y to (7)-(9) is integral, and the theorem now follows from the fact that for j = 0,...,N+1,

x(j,e(j)) = y(j,e(j)) and
x(j,t) = y(j,t) - y(j,t-1) for t = e(j)+1,...,L(j).

Theorem 1 has important implications for the solution of P1 through either Lagrangian relaxation or a cutting-plane approach. By transforming the x variables to the y variables defined in (6), the Lagrangian relaxation of P1 obtained by dualizing the resource constraints (5) can be solved efficiently by applying the network simplex algorithm to the dual. Theorem 1 implies that this Lagrangian relaxation has the *integrality property*, i.e., relaxing the integer restrictions in the Lagrangian relaxation still results in an integral solution. We let LP1 denote the LP relaxation of P1 (including the resource constraints). This Lagrangian relaxation therefore cannot yield better bounds than the optimal objective value of LP1 (Geoffrion, 1974). In particular, the Lagrangian relaxation of Fisher (1973) and of Christofides, Alvarez-Valdes, and Tamarit (1987) has the integrality property and does not require implicit enumeration for its solution.

With regard to a cutting-plane approach, Theorem 1 implies that any cut that makes infeasible a fractional solution to LP1 can be got only by invoking at least one resource constraint, possibly along with convexity and precedence constraints.

We now show how P1 represents the precedence constraints in a disaggregate manner. Summing up (4) over t = e(n)-d(n),...,L(m)-1, we get

$$\sum_{s>e(n)-d(n)} \big(s+d(n)-e(n)\big).x(m,s) \;+\; \sum_{s<L(m)+d(n)} \big(L(m)-s+d(n)\big).x(n,s) \;\le L(m)+d(n)-e(n)... \qquad (4')$$

Observe that for s ≤ e(n)-d(n), the coefficients of terms in the first sum are non-positive, while for s ≥ L(m)+d(n), the coefficients of terms in the second sum are non-positive. Therefore, (4') implies that

$$\sum_{s=e(m)}^{L(m)} s\,x(m,s) \;+\; \big(d(n)-e(n)\big) \sum_{s=e(m)}^{L(m)} x(m,s) \;-\; \sum_{s=e(n)}^{L(n)} s\,x(n,s) \;+\; \big(L(m)+d(n)\big) \sum_{s=e(n)}^{L(n)} x(n,s)$$

$$\le\; L(m) + d(n) - e(n),$$

i.e., (4') implies that

$$d(n) + \sum_{s=e(m)}^{L(m)} s\,x(m,s) \le \sum_{s=e(n)}^{L(n)} s\,x(n,s). \qquad ... \qquad (4'')$$

However, (4") is precisely the conventional, *aggregate* representation of the precedence constraints. By deriving (4") formally from (4), we have shown that P1 is as tight a formulation as the conventional formulation in which the precedence constraints are represented by (4"). In fact, the following simple counter-example shows that not every basic feasible solution to (3) and (4") is necessarily 0-1, and hence, P1 can actually be tighter than the conventional formulation.

Let N = 2. Assume that activity 1 precedes activity 2. Let d(1) = 2, and d(2) = 3. From the precedence relation between activity 1 and activity 2, we can deduce that e(1) = 2 and e(2) = 5. The relevant constraints, i.e., (3) and (4"), are:

x(1,2) + x(1,3) + ... = 1,
x(2,5) + x(2,6) + ... = 1,

and

2x(1,2) + 3x(1,3) ... + 3 ≤ 5x(2,5) + 6x(2,6) +...

A basic solution with x(1,2) = x(1,4) = 0.5 and x(2,6) = 1 is feasible in the constraints (3) and (4"), along with the non-negativity restrictions, but is a fractional solution which, according to Theorem 1, cannot be a basic feasible solution of the disaggregated constraints (4).

The disaggregated constraints (4) were also used by Christofides, Alvarez-Valdes, and Tamarit (1987) who generated them on-the-fly as cutting-planes. Juang (1994), on the other hand, incorporated the "precedence cuts" of Wilson (1990), which are logically equivalent to (4), into her model of job shop scheduling problems. By applying the transformation (6) above, she was then able to identify a hidden network structure, so that a Lagrangian relaxation of the resource constraints allowed the dual subproblem to be solved by a network algorithm to yield integer basic solutions.

We close this section by listing three illustrations of the fact that besides being a tighter IP formulation, P1 enables us to model a wider variety of problems than the conventional, aggregate formulation.

1. *The parameter which denotes the duration of an activity can vary not just with the activity but also with the finish period of that activity*. Thus, we may denote the duration of activity j not just as d(j), but as d(j,t) where t is the finish period of activity j.
2. *The parameter which denotes the consumption of a resource by an activity can vary not just with the resource and the activity, but also with the finish period of the activity.*

3. *Arbitrary minimal and maximal time-lags between activities may readily be incorporated. Further, these time-lags can depend on the completion times of the predecessor and successor activities.* Such time-lags are referred to as generalized precedence relations by De Reyck and Herroelen (1996).

3. PROBLEM PREPROCESSING

Researchers have shown that the efficacy of cutting-plane algorithms can be enhanced through problem preprocessing techniques (e.g., Crowder, Johnson, and Padberg, 1983; Johnson, Kostreva, and Suhl, 1985;). Typically, these include variable fixing, simple logical checks for problem infeasibility and constraint redundancy, and coefficient reduction. We now discuss such techniques in the context of the project scheduling problem.

3.1 IMPROVING LOWER AND UPPER BOUNDS ON ACTIVITY FINISH TIMES

By a consideration of resource constraints, we are generally able to tighten the lower and upper bounds on the activity completion times, e(j) and L(j), respectively, relative to their initial values of EFT(j) and LFT(j), which were determined by the precedence constraints alone. Our procedure is motivated by the critical sequence approach to developing lower bounds on project makespan reported by Stinson, Davis, and Khumawala (1978).

Our procedure requires a succession of forward and backward passes to tighten the bounds on each activity's completion time. We repeat the forward pass of all the activities until none of the lower bounds e(·) can be improved. Similarly, we repeat the backward pass of the activities until none of L(·) can be improved. When the procedure for improving e(·) and L(·) terminates, if there is an activity j such that e(j) > L(j), then P1 is infeasible, in which case the incumbent is optimal.

3.2 IDENTIFYING REDUNDANT CONSTRAINTS AND FIXING VARIABLES

Having improved the lower and upper bounds on activity finish times, we perform simple checks for the infeasibility of P1, fix variables at zero when possible, and try to identify redundant constraints (both resource and precedence restrictions).

3.3 COEFFICIENT REDUCTION

For given activities k and t, if we cannot determine through the above procedure that resource constraint (k,t) is redundant, we proceed to tighten the constraint by adapting the coefficient reduction techniques proposed by Johnson, Kostreva, and Suhl (1985) for the project scheduling problem. In such a process, we also make a final attempt to ascertain whether resource constraint (k,t) is redundant.

After problem preprocessing is complete, those resource constraints (k,t) which can not be proven redundant are generally transformed to

$$\sum_{j\in J(k,t)} mr(j,k,t).\left[\sum_{s=Max(t,e(j)):x(j,s)\neq 0}^{Min(L(j),t+d(j)-1)} x(j,s)\right] \leq \hat{A}(k,t)\,. \qquad (5')$$

The impact of preprocessing on P1 is that some of the variables may be prefixed at zero. Henceforth, we refer to the modified form of P1 as P, let PLP denote the LP relaxation of P, and let Conv(P) denote the convex hull of feasible solutions to P.

4. CONSTRAINT GENERATION

As with any FCPA, after problem preprocessing, we initially approximate Conv(P) by PLP and successively refine the approximation by adding valid inequalities for Conv(P) on-the-fly. Specifically, at each stage, we see whether the optimal solution to PLP is 0-1. If yes, it must also be optimal for P. Otherwise, we search for valid inequalities that are violated by the fractional solution. If one or more of these are found, we add them to PLP and solve the refined approximation of Conv(P). On-the-fly constraint generation ceases when: (a) the computed optimum to PLP is 0-1; (b) the computed optimum to PLP is fractional and we are unable to identify violated inequalities; or (c) the computed optimum to PLP is fractional and the improvement on the lower bound that is yielded since the preceding solution of the LP is negligible. In cases (b) and (c), we resort to conventional LP-based branch-and-bound to solve the problem to optimality.

As mentioned earlier, any procedure for generating valid constraints that are violated by a fractional solution (which we will refer to as $\overline{X}$) must invoke at least one resource constraint (see Theorem 1). We label a resource constraint as type 1 if all the constraint coefficients are unity and label it as type 2 otherwise. No violated valid inequality can be generated by considering a single type 1 resource constraint (possibly along with a non-overlapping set of generalized upper bounding (GUB) constraints, derived from (3) and (4)) -- this is because the constraint matrix corresponding to the type 1 constraint and the non-overlapping set of GUB constraints is totally unimodular. However, it is possible to generate a violated valid inequality with a type 2 constraint (Crowder, Johnson, and Padberg, 1983).

Hence, in the first phase of constraint generation, we consider each type 2 'knapsack' constraint in turn, along with a non-overlapping set of GUB constraints. Then, we try to identify valid violated minimal cover inequalities using results for the 0-1 knapsack problem with GUB constraints (Johnson and Padberg, 1981; Wolsey, 1990; Nemhauser and Vance, 1994; Sankaran, 1995). In the second phase of constraint generation, we attempt to identify valid violated 'clique'

constraints.

4.1 Minimal Cover Inequalities

Consider a single type 2 resource constraint (k,t). Define $\hat{J}(k,t)$ to be {j∈J(k,t): mr(j,k,t) > 0}. (Hence $\hat{J}(k,t)$ is the set of precisely those activities whose x(·,·) variables appear in resource constraint (k,t).) We then partition $\hat{J}(k,t)$ into p ordered pairs {(m_u,n_u): u = 1,...,p} and $|\hat{J}(k,t)|$ - 2p singletons. For each u, m_u precedes n_u. Also, let

$\tilde{J}(k,t) = \hat{J}(k,t) \setminus \left(\bigcup_{u=1}^{p} \{m_u, n_u\} \right)$. ($\tilde{J}(k,t)$ is the union of all the singletons.)

For $j \in \hat{J}(k,t)$, define $z(j,t) = \sum_{s=Max(t,e(j)):x(j,s)=0}^{Min(L(j),t+d(j)-1)} x(j,s)$. Then, {z(j,t)} must satisfy the following constraints.

$$\sum_{j \in \hat{J}(k,t)} mr(j,k,t).z(j,t) \le \hat{A}(k,t). \quad \ldots \quad (10)$$

For u = 1, ..., p,

$$z(m_u,t) + z(n_u,t) \le 1. \quad \ldots \quad (11)$$

For all $j \in \tilde{J}(k,t)$,

$$z(j,t) \le 1. \quad \ldots \quad (12)$$

Also,

$$\text{for all } j \in \hat{J}(k,t) \text{ and t, z(j,t) is 0-1.} \quad \ldots \quad (13)$$

Observe that (10) is a knapsack constraint in 0-1 variables while (11) and (12) denote a set of mutually exclusive GUB constraints. For a given fractional solution $\bar{X}$, let $\bar{z}(j,t)$ be the corresponding value of z(j,t). Then, we implement a separation algorithm for detecting violated cover inequalities for the 0-1 knapsack problem with GUB constraints (see Wolsey, 1990). If a violated cover inequality is found, we lift it and add it to the LP. The above procedure is executed in turn for all type 2 resource constraints. In our current implementation, we select the pairs {(m_u,n_u)} by maximizing $\sum_u \left[mr(m_u,k,t).\tilde{z}(m_u,t) + mr(n_u,k,t).\tilde{z}(n_u,t) \right]$ in a greedy way.

4.2 Clique Inequalities

Suppose V is a subset of variables in P such that

$$\sum_{(j,t) \in V} x(j,t) \le 1 \quad \ldots \quad (14)$$

is valid for P. Then, we term V a *clique* and (14) a *clique constraint* owing to the graph-theoretic interpretation of such inequalities for the set packing problem (Padberg, 1979). Following Nemhauser and Sigismondi (1992), we use a simple greedy procedure for identifying valid clique inequalities that are violated by a given fractional solution to (3), (4), and (5').

Theorem 1 implies that to generate a violated clique inequality, at least one resource constraint must be invoked. Hence, initially in the clique generation procedure, we augment V by (n,t) only if, for all (m,s) ∈ V, the following holds:

B1: There exists (k,u) where u ∈ [s-d(m)+1,s] ∩ t-d(n)+1,t], and mr(m,k,u) + mr(n,k,u) > $\hat{A}(k,u)$.

If B1 is satisfied, then if activity m terminates in period s, activity n cannot terminate in period t (and vice versa) since there is a conflict for resource k in a period (namely, period u).

In practice, we augment V by $(\hat{n},\hat{t})$ where $(\hat{n},\hat{t}) \in$ Argmax{ $\bar{x}(n,t)$:$\bar{x}(n,t) > 0$ and B1 holds for all (m,s) ∈ V}. After V is augmented at least once in this manner, we continue to augment V as before with the difference that (n,t) is a candidate if $\bar{x}(n,t) > 0$ and either B1 or one of the following properties holds for each (m,s) ∈ V.

B2: m = n (in which case periods s and t are alternate finish times for the same activity).

B3: m precedes n and s+D(m,n) > t or n precedes m and t+D(n,m) > s (in which case precedence relations stipulate that if m terminates in period s, then n cannot terminate in period t and vice versa).

We initialize V as { $(\bar{n},\bar{t})$ } where $(\bar{n},\bar{t})$ maximizes $|\bar{x}(n,t)$ - 0.5|. If the search for a violated clique inequality fails, we make one more attempt by initializing V as { $(\tilde{n},\tilde{t})$ } where $(\tilde{n},\tilde{t})$ maximizes $\bar{x}(n,t)$ over all pairs (n,t) such

that $\bar{x}(n,t)$ is fractional.

5. COMPUTATIONAL EXPERIENCE

We have implemented our algorithm by coding it in FORTRAN and linking our subroutines with the solver LINDO (Schrage, 1989). We have also implemented four heuristics, namely, MINLST, MINLFT, SPT, and GRD, to find a good, initial solution (Boctor, 1990). The incumbent is initially the best of the four heuristic solutions available. We search for minimal cover cuts as described in Section 4, and only if unsuccessful, also search for a violated clique inequality.

	Problem ID	No. of activities	Way in which solved	No. of problem set-ups	Total no. of successful iterations of constraint generation	Initial duality gap (%)	Final duality gap (%)	Total no. of pivots	Total no. of branches in B&B	Total CPU time in seconds
1	BMS	14	LP:S	2	0	1.59	0.00	98	1	0.42
2	DAV1	7	P.P.	0	-	-	-	-	-	0.02
3	DAV2	13	C.G.:S	1	1	0.00	0.00	53	-	0.18
4	DAV3	22	P.P.	0	-	-	-	-	-	0.03
5	DAV4	22	P.P.	0	-	-	-	-	-	0.03
6	DAV5	22	B&B:I	1	1	-	-	135	7	0.55
7	M&M1	9	P.P.	0	-	-	-	-	-	0.001
8	M&M2	9	C.G.:I	1	1	-	-	17	-	0.07
9	MART	18	P.P.	0	-	-	-	-	-	0.04
10	SRM1	8	P.P.	0	-	-	-	-	-	0.01
11	SRM2	8	P.P.	0	-	-	-	-	-	0.01
12	DUPNT	23	B&B:I	1	3	-	-	1480	12	7.15
13	P35E	35	C&G:I	2	4	-	-	4638	3	46.24
14	P35F	35	P.P.	1	1	-	-	9290	11	65.52
15	A1A	22	B&B:I	2	7	-	-	11120	8	114.01
16	A2A	22	B&B:I	2	13	-	-	136543	230	1578.03
17	A1B	22	N.S.	1	5	-	-	>1,000,000	-	-
18	A2B	22	N.S.	2	9	-	-	>1,000,000	-	-
19	A4A	22	P.P.	0	-	-	-	-	-	0.04
20	A4B	22	N.S.	1	8	-	-	>1,000,000	-	-
21	A5A	22	C.G.:I	2	4	-	-	15441	5	169.57
22	A6A	22	B&B:I	1	5	-	-	131495	132	1900.44
23	A7A	22	P.P.	1	1	-	-	3458	4	19.87
24	A9A	22	L.P.:I	1	0	-	-	44	-	0.18
25	A9B	22	C.G.:I	2	9	-	-	70319	10	1071.42

Legend:
P.P. implies that the infeasibility of the last IP was diagnosed in the problem-preprocessing stage.
L.P:I (resp., L.P.:S) implies that the infeasibility (resp., optimal solution) of the last IP was ascertained by solving the non-augmented LP relaxation.
C.G.:I (resp., C.G.:S) implies that the infeasibility (resp., optimal solution) of the last IP was ascertained through constraint generation.
B&B:I implies that the infeasibility of the last IP was ascertained after invoking branch-and-bound.
N.S. implies that the problem was not solved within an upper limit of 1,000,000 pivots.

TABLE 1: Illustrative computational results

When the optimum of PLP is fractional and either no more cuts can be found or the lower bound increases only slightly after the cuts are added, we invoke LINDO's branch-and-bound solver. When LINDO finds an integer solution, we update the incumbent and repeat the whole procedure. That is, once again, we define g (the upper bound on project

makespan), compute lower and upper bounds on activity finish times, identify problem infeasibility/redundant constraints, tighten constraints (coefficient reduction), and generate cutting-planes on-the-fly at the root node.

Table 1 presents computational results, using an IBM RS6000, of our algorithm on the first 25 problems in a set of 110 standard test problems frequently reported in the literature on resource-constrained project scheduling. We imposed an upper limit of 1,000,000 pivots. Three of the 25 problems were not solved within this limit.

A problem may be solved in one of several ways. These are itemized below. ('The last IP' refers to the IP which is formulated on the basis of that incumbent which is proven to be optimal at the conclusion of the algorithm.)

- P.P. The last IP is proven as being infeasible in the problem preprocessing stage.
- L.P.:I The last IP is proven to be infeasible by establishing that the non-augmented LP relaxation is infeasible.
- L.P.:S The optimal solution to the last IP is found by solving the non-augmented LP relaxation (ie., the optimal solution to the LP relaxation is 0-1).
- C.G.:I The last IP is identified as being infeasible, during the constraint generation phase (ie., the augmented LP relaxation is infeasible).
- C.G.:S The optimal solution to the last IP is found by solving the augmented LP relaxation (ie., the optimal solution to the augmented LP relaxation is 0-1).
- B&B:I The last IP is identified as being infeasible during the search (using branch-and-bound) for a solution that is better than the incumbent.

Note that in the algorithm, whenever the branch-and-bound phase encounters a feasible solution (one that is necessarily better than the incumbent), we once again set up the formulation *unless* it is shown to be infeasible during the problem preprocessing phase. (By 'setting up' the formulation, we mean actually loading it in LINDO by specifying the variables, constraints, and the objective function.) The numbers of branches, pivots, and iterations of successful constraint generation and augmentation are summed over all the formulations that are set up during the execution of the algorithm.

The two statistics, 'initial duality gap' and 'final duality gap', are defined only when the optimal solution of the LP relaxation of the last IP (either augmented with cuts generated on-the-fly or otherwise) is 0-1. In such a case, the initial duality gap is the duality gap between the *first* IP and its *non-augmented* LP relaxation, and the final duality gap is the duality gap between the *last* IP and its *fully augmented* LP relaxation. Both gaps are reported as a percentage of the optimal objective value of the IP (which is the same for both the first and the last IP's).

We illustrate the statistics in Table 1 with reference to problem instance P35F. First, with reference to the initial formulation, one iteration of constraint generation and augmentation was executed. Then, branch-and-bound was invoked to find a solution that was better than the incumbent, i.e., one that replaced the incumbent. Finally, in the problem preprocessing phase prior to the actual set-up of the second formulation, the incumbent was shown to be optimal, and hence, a second formulation was not set up.

The CPU-times displayed in Table 1 vary greatly even across problems with the same number of activities. The CPU-times are especially small when the formulation is not set up even once during the execution of the algorithm.

Based on our computational experience, we have gleaned a few pointers for future implementations of our algorithm.

1. Other efficient heuristics can be explored in the initial search for an incumbent -- the marginal computational cost of performing an extra heuristic search is much less than the effort that is required to solve the problem by branch-and-bound.
2. Optimizers such as MINTO (Nemhauser, Savelsbergh, and Sigismondi, 1994) are designed for implementing strong FCPAs such as ours. When we began the computational study, we had recourse only to LINDO which employs a standard, DFS variable-branching strategy; thus, special-ordered-sets (SOSs), such as constraints (3), could not be incorporated for branching purposes in LINDO (this is not the case with MINTO). Hence, the use of MINTO and like optimizers will likely enhance the performance of the algorithm.

   Bricker (1977) explained how branching on the y variables, that are obtained by transforming the x variables [refer (6) in Section 2], yields a search tree that is equivalent to branching by partitioning the SOSs, viz., constraints (3). Branching by partitioning the SOSs is commonly recognized as being superior to branching on the variables in the SOSs individually. Hence, branching on the y variables would be superior to branching on the x variables. Bricker noted that the former branching strategy would be especially useful for those models in which the x variables are implicitly non-negative. Unfortunately, that is not the case with the present problem; when formulating in terms of the y variables, we do need to include constraints of the form $y(j,t) \geq y(j,t-1)$ to ensure the non-negativity of the x(j,t) variables.
3. For the present problem, the LP relaxations are notoriously degenerate; hence, a code that uses interior-point methods to solve LP's might be more appropriate (LINDO employs the revised simplex method to solve linear programs).
4. In terms of reducing the duality gap through constraint generation, it might pay to include separation heuristics that are specifically designed for resource-constrained, project scheduling, and the total CPU time might even decrease as a result. Alvarez-Valdes and Tamarit (1993) described strong valid inequalities and facets for the underlying polytope, but

did not describe separation heuristics that can be used to identify violated inequalities.

6. CONCLUSION

We have presented a strong, fractional cutting-plane algorithm for resource-constrained project scheduling, replete with techniques for coefficient reduction, variable elimination, on-the-fly generation of cutting-planes, etc. A recent survey of the literature (Ozdamar and Ulusoy, 1995) indicates that no such algorithm for resource-constrained project scheduling had then been reported.

We noted in Section 2 that an LP-based branch-and-bound approach is very robust for modeling project scheduling problems. Hence, it remains a very attractive proposition. We believe that a more sophisticated implementation of the algorithm proposed by this paper will be competitive with the enumerative algorithms reported in the literature.

Acknowledgements: We are very grateful to Prof. James Patterson for readily providing us with a standard collection of test problems, and to the referees, whose detailed suggestions helped greatly in improving the readability of this paper.

7. REFERENCES

Alvarez-Valdes, R., and Tamarit, J.M. (1993) "The project scheduling polyhedron: dimension, facets and lifting theorems", *European Journal of Operational Research*, Vol. 67, pp. 204-220.

Boctor, F.F. (1990) "Some efficient multi-heuristic procedures for resource-constrained project scheduling", *European Journal of Operational Research*, Vol. 49, pp. 3-13.

Bricker, D.L. (1977) "Reformulation of special ordered sets for implicit enumeration algorithms with applications in nonconvex separable programming", *AIIE Transactions*, Vol. 9, pp. 195-203.

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

Crowder, H., Johnson, E.L., and Padberg, M.W. (1983) "Solving large scale zero-one linear programming problems", *Operations Research*, Vol. 31, pp. 803-834.

De Reyck, B., and Herroelen, W. (1996) "Optimal and heuristic solution procedures for the resource-constrained project scheduling problem with generalized precedence relations", presented at INFORMS Atlanta Fall 1996.

Demeulemeester, E.L., and Herroelen, W.S. (1997) "New benchmark results for the resource-constrained project scheduling problem", *Management Science*, Vol. 43, pp. 1485-1492.

Fisher, M.L. (1973) "Optimal solution of scheduling problems using Lagrange multipliers: Part I", *Operations Research*, Vol. 21, pp. 1114-1127.

Geoffrion, A.M. (1974) "Lagrangean Relaxation for Integer Programming", *Math. Prog. Study 2*, pp. 82-114.

Gomory, R.E. (1963) "An algorithm for integer solutions to linear programs", in *Recent Advances in Mathematical Programming*, Graves and Wolfe (editors), McGraw-Hill, pp. 269-302.

Johnson, E.L., Kostreva, M.M., and Suhl, U.H. (1985) "Solving 0-1 integer programming problems arising from large scale planning models", *Operations Research*, Vol. 33, pp. 803-819.

Johnson, E.L., and Padberg, M.W. (1981) "A note on the knapsack problem with special ordered sets", *Operations Research Letters*, Vol. 1, pp. 18-22.

Juang, S.H. (1994) "Optimal solution of job shop scheduling problems - a new network flow approach", Unpublished Ph.D. dissertation, The University of Iowa, Iowa City, IA, USA.

Nemhauser, G.L., Savelsbergh, M.W.P., and Sigismondi, G.C. (1994) "MINTO: a mixed integer optimizer", *Operations Research Letters*, Vol. 15, pp. 47-58.

Nemhauser, G.L., and Sigismondi, G.C. (1992) "A strong cutting plane/branch-and-bound algorithm for node packing", *Journal of the Operational Research Society*, Vol. 43, pp. 443-457.

Nemhauser, G.L., and Vance, P.H. (1994) "Lifted cover facets of the 0-1 knapsack polytope with GUB constraints", *Operations Research Letters*, Vol. 16, pp. 255-263.

Nemhauser, G.L., and Wolsey, L.A. (1988) *Integer and combinatorial optimization*, John Wiley, New York.

Ozdamar, L., and Ulusoy, G. (1995) "A survey on the resource-constrained project scheduling problem", *IIE Transactions*, Vol. 27, pp. 574-586.

Padberg, M.W. (1973) "On the facial structure of set packing polyhedra", *Math. Prog.*, Vol. 5, pp. 199-215.

Padberg, M.W. (1979) "Covering, packing and knapsack problems", *Annals of Discrete Mathematics*, Vol. 4, pp. 265-287.

Patterson, J.H., and Huber, W. (1974) "A horizon-varying zero-one approach to project scheduling", *Management Science*, Vol. 20, pp. 990-998.

Sankaran, J.K. (1995) "Minimal cover inequalities for the 0-1 knapsack problem with special ordered sets", Working paper, MSIS Department, U. of Auckland, Private Bag 92019, Auckland, New Zealand.

Sankaran, J.K., Bricker, D.L., and Juang, S-W (1998) "A strong fractional cutting-plane algorithm for resource-

constrained project scheduling", Working paper, MSIS Department, U. of Auckland, Private Bag 92019, Auckland, New Zealand.

Schrage, L.E. (1970) "Solving resource-constrained network problems by implicit enumeration - nonpreemptive case", *Operations Research*, Vol. 18, pp. 263-278.

Schrage, L.E. (1989) *Linear, integer, and quadratic programming with LINDO*, 4th edition, The Scientific Press, San Francisco, CA.

Stinson, J.P., Davis, E.W., and Khumawala, B.M. (1978) "Multiple resource-constrained scheduling using branch-and-bound", *AIIE Transactions*, Vol. 10, pp. 252-259.

Wilson, J.M. (1990) "Generating cuts in integer programming with families of special ordered sets", *European Journal of Operational Research*, Vol. 46, pp. 101-108.

Wolsey, L.A. (1990) "Valid inequalities for 0-1 knapsacks and mips with generalized upper bound constraints", *Discrete Applied Mathematics*, Vol. 29, 251-261.