A Branch–and–Bound Algorithm for the Project Duration Problem Subject to Temporal and Cumulative Resource Constraints

Christoph Schwindt

Institut für Wirtschaftstheorie und Operations Research
University of Karlsruhe
D–76128 Karlsruhe, Germany
e–mail: schwindt@wior.uni–karlsruhe.de

Abstract. In this paper we propose a branch–and–bound algorithm for the inventory–constrained project duration problem where minimum and maximum time lags between project events are prescribed. Any event may take items from or add items to one or several storages which are represented by cumulative resources. For each resource a minimum inventory has to be secured and a maximum inventory must not be exceeded at any point in time. This problem represents a generalization of several well–known resource–constrained project scheduling problems and is of great relevance to process flow scheduling. Our approach relies on the basic concepts devised by Neumann (1999). We perform an enumeration of alternatives to avoid stock shortage and stock excess in the optimal solution to the inventory–unconstrained problem. Performing a depth–first search, the search space is stepwise reduced by introducing disjunctive precedence constraints between disjoint sets of events. A feasible solution has been attained if all inventory conflicts are resolved squaring with the temporal constraints between the project events. The search space of an enumeration node represents the connected intersection of unions of polyhedral cones with the polyhedron of time–feasible solutions. The corresponding optimization problems are solved by a pseudo–polynomial fix–point algorithm which exploits the existence of a unique minimal point. The enumeration tree is pruned by lower bounds and fathoming rules. The algorithm can easily be adapted to the minimization of further regular objective functions like mean weighted flow time or maximum lateness. Finally, we sketch how the procedure can be truncated to an efficient heuristic.

Key words: Project management, scheduling, storages, temporal constraints, branch–and–bound.

1 Introduction

We consider a project for which n *events* numbered consecutively from 1 to n have been defined. Additionally we introduce the events 0 and $n+1$ which represent the beginning and the termination of the project, respectively. Thus, $V = \{0, 1, \ldots, n, n+1\}$ is the set of all project events. By $\mathcal{R}$ we denote the set of storages or *cumulative resources*. For each resource $k \in \mathcal{R}$ we suppose a *minimum inventory* $\underline{R}_k$ and a *maximum inventory* $\overline{R}_k$ to be given. $\underline{R}_k$ may be regarded as the safety stock, $\overline{R}_k$ as the item–capacity of storage k. Any event is associated with demands r_{ik} for resources $k \in \mathcal{R}$ termed *replenishment* if positive and *consumption* if negative. Let $S_i \geq 0$ be the time of occurrence of event i and let $S = (S_0, S_1, \ldots, S_n, S_{n+1})$ be the project *schedule* which has to be determined. By

$\mathcal{A}(S,t) := \{i \in V \mid S_i \leq t\}$ we denote the *active set* of events which occurred by time t. Then $r_k(S,t) := \sum_{i \in \mathcal{A}(S,t)} r_{ik}$ is the inventory of resource k at time t if the project is executed according to schedule S. Between the occurrences of events i and j *minimum time lags* d_{ij}^{min} and *maximum time lags* d_{ij}^{max} may be prescribed. Without loss of generality we assume all time lags to be integers. As pointed out in Neumann (1999), the relationships between the project events can be represented by a network where the events coincide with the nodes and the temporal constraints are represented by arcs $\langle i,j \rangle$ with weight $\delta_{ij} = d_{ij}^{min}$ and arcs $\langle j,i \rangle$ with weight $\delta_{ji} = -d_{ij}^{max}$, respectively. By $D = (d_{ij})_{i,j \in V}$ we denote the matrix of longest path lengths d_{ij} from nodes i to nodes j. D represents the transitive closure of all minimum and maximum time lags between events $i, j \in V$. With E being the set of arcs of the resulting network, the project duration problem (P) can be stated as follows:

$$
\text{(P)} \quad \left\{ \begin{array}{lll}
\text{Min.} & S_{n+1} & (1) \\
\text{s.t.} & S_0 = 0 & (2) \\
& S_j - S_i \geq \delta_{ij} \qquad (\langle i,j \rangle \in E) & (3) \\
& \underline{R}_k \leq r_k(S,t) \leq \overline{R}_k \qquad (k \in \mathcal{R}, t \geq 0) & (4)
\end{array} \right.
$$

Problem (1), (2), and (3) corresponds to the resource relaxation (P_T) of (P). Solutions to (P_T) are called time–feasible schedules, solutions to (P) are referred to as feasible schedules. By $\mathcal{S}_T$ and $\mathcal{S}$ we denote the set of all time–feasible and all feasible schedules, respectively.

In the following, we discuss the decomposition of (P) into a sequencing and a scheduling problem. The *sequencing problem* SEQUENCE consists of the determination of a nonempty set of schedules $\mathcal{Q}$ including at least one feasible schedule $S \in \mathcal{S}$ which solves the corresponding project duration problem on $\mathcal{Q}$ to optimality. The *scheduling problem* SCHEDULE($\mathcal{Q}$) is the problem to find a schedule $S \in \mathcal{S} \cap \mathcal{Q}$ minimizing the objective function on $\mathcal{Q}$. The basic principle of our branch–and–bound algorithm is to enumerate a finite number σ of solutions $\mathcal{Q}_1, \ldots, \mathcal{Q}_\sigma$ to SEQUENCE with $\mathcal{Q}_\rho \subseteq \mathcal{S}_T$ for all $\rho = 1, \ldots, \sigma$ and $\bigcup_{\rho=1}^{\sigma} \mathcal{Q}_\rho \supseteq \mathcal{S}$. Since a feasible schedule S^+ is optimal exactly if

$$
S_{n+1}^+ = \min_{\rho=1,\ldots,\sigma} \min_{S \in \mathcal{Q}_\rho} S_{n+1},
$$

solving the corresponding scheduling problems SCHEDULE($\mathcal{Q}_\rho$) ($\rho = 1, \ldots, \sigma$) yields an optimal schedule exactly if the feasible region $\mathcal{S}$ is nonempty.

In order to limit the size of the enumeration tree, the covering $\{\mathcal{Q}_1, \ldots, \mathcal{Q}_\sigma\}$ of $\mathcal{S}$ should include as few sets $\mathcal{Q}_\rho$ as possible. On the other hand, the scheduling problems SCHEDULE($\mathcal{Q}_\rho$) should be solved efficiently.

In Sections 2 and 3 we devise algorithms for the sequencing and the scheduling problem, respectively. Solutions $\mathcal{Q}$ to SEQUENCE represent connected unions of convex polyhedra whose number generally grows exponentially in the number n of project events. Nevertheless, the corresponding problems SCHEDULE($\mathcal{Q}$) can be solved by an algorithm whose time complexity is linear in n and in an upper bound $\overline{d}$ on the minimum project duration. In Section 4 we discuss the branching strategy, lower bounds on the minimum project duration, and fathoming rules for a further pruning of the enumeration tree. Section 5 is devoted to the truncation of the proposed branch–and–bound algorithm to a filtered beam search heuristic. Section 6 deals with an experimental performance analysis.

2 Solving the Sequencing Problem

For the basic concepts of *(minimal) lacking sets* and *(minimal) exceeding sets* of events we refer to Neumann (1999). Recall that a schedule S observes the inventory constraints exactly if (1) for all minimal lacking sets F at least one event $j \in F$ with $r_{jk} < 0$ does not occur before an event $i \in V \setminus F$ which replenishes resource k and (2) for all minimal exceeding sets F at least one event $j \in F$ with $r_{jk} > 0$ does not occur before an event $i \in V \setminus F$ which consumes resource k.

Now let F be a lacking (exceeding) set. An inclusion–minimal set B with

$$\sum_{i \in F \setminus B} r_{ik} \geq \underline{R}_k \quad (\sum_{i \in F \setminus B} r_{ik} \leq \overline{R}_k) \qquad \text{for all } k \in \mathcal{R} \tag{5}$$

is termed *minimal delaying alternative* for F. The concept of minimal delaying alternatives has been used by several authors in the field of project scheduling subject to renewable resource constraints (e.g. by Demeulemeester and Herroelen 1992, Icmeli and Erengüç 1996, De Reyck and Herroelen 1998, or Schwindt 1998). The delay of events $j \in B$ up to the occurrence of a replenishing (consuming) event $i \in V \setminus F$ resolves all inventory conflicts induced by minimal lacking or exceeding sets F', respectively, with $F' \cap B \neq \emptyset$ and $i \in V \setminus F'$. The number of all minimal delaying alternatives B for a lacking or exceeding set F may be exponential in the cardinality of F. It can be computed starting with the one–element subsets B of F which are augmented (e.g. by events with higher number, in order to avoid redundancy) until condition (5) is satisfied. B is inclusion–minimal exactly if no event $j \in B$ can be removed such that (5) still holds.

The question to be answered next is how to select event i up to whose occurrence all events $j \in B$ have to be shifted. One possibility is to enumerate the replenishing (consuming) events $i \in V \setminus F$. Since in general it is not possible to decide upon the constraints $S_j \geq S_i$ $(j \in B)$ which lead to an optimal schedule, all events $i \in V \setminus F$ have to be examined separately. In what follows, we propose the use of disjunctive precedence constraints $A \prec B$ between disjoint sets $A \subseteq V \setminus F$ and B which consider all delaying events i simultaneously:

$$\min_{j \in B} S_j \geq \min_{i \in A} S_i \tag{6}$$

with $A := \{i \in V \setminus F \mid r_{ik} > 0\}$ for lacking F and $A := \{i \in V \setminus F \mid r_{ik} < 0\}$, otherwise.

Our algorithm for computing a solution $\mathcal{Q}$ to SEQUENCE is based on the resource relaxation (P_T). We initialize $\mathcal{Q}$ by $\mathcal{S}_T$. If $\mathcal{Q}$ is empty, then stop: no feasible solution could be found. Otherwise, we determine an optimal solution S^+ to SCHEDULE($\mathcal{Q}$). If S^+ obeys the inventory constraints, then stop: a feasible solution has been determined. Else, we determine the earliest point in time t at which an inventory conflict occurs and compute a minimal delaying alternative B to $\mathcal{A}(S^+, t)$. $\mathcal{Q}$ is cut by the disjunctive precedence constraint (6). These steps are repeated until either a feasible solution has been found or the procedure terminates since the search space $\mathcal{Q}$ has been restricted to void.

How to solve the scheduling problems SCHEDULE($\mathcal{Q}$) will be discussed in the next section.

3 Solving the Scheduling Problem

Let $\mathcal{Q}$ be given by ν disjunctive precedence constraints $A_\mu \prec B_\mu$ with $\mu = 1, \ldots, \nu$. SCHEDULE($\mathcal{Q}$) can be stated as follows:

$$\left.\begin{array}{lll} \text{Min.} & S_{n+1} & \\ \text{s.t.} & S_0 = 0 & \\ & S_j - S_i \geq \delta_{ij} & (\langle i,j\rangle \in E) \\ & \min_{j\in B_\mu} S_j \geq \min_{i\in A_\mu} S_i & (\mu = 1, \ldots, \nu) \end{array}\right\} \tag{7}$$

The algorithm for solving (7) requires an upper bound $\overline{d}$ on the minimum project duration.

Proposition 1.

Let the feasible region $\mathcal{S}$ of (P) be nonempty. Then

$$\overline{d} = \sum_{j\in V} \max\{0, \max_{\langle i,j\rangle \in E} \delta_{ij}\} \tag{8}$$

represents an upper bound on the minimum project duration.

SCHEDULE($\mathcal{Q}$) can be solved as follows. Starting with the unique minimal point $ES = (d_{0i})_{i\in V}$ of $\mathcal{S}_T$, we check whether or not the disjunctive precedence constraints $A_\mu \prec B_\mu$ ($\mu = 1, \ldots, \nu$) are met. Assume that the current schedule S^+ does not satisfy $A_\mu \prec B_\mu$. Then any schedule $S \in \mathcal{Q}$ observes the inequalities $S_j \geq \min_{i\in A_\mu} S_i^+$ such that S_j^+ can be increased up to value $\min_{i\in A_\mu} S_i^+$ for all $j \in B_\mu$. Subsequently, we restore the time–feasibility of S^+ by setting $S_h^+ := \max\{S_h^+, \max_{j\in B_\mu}(S_j^+ + d_{jh})\}$ for all $h \in V$. These steps are performed until either all disjunctive precedence constraints are satisfied or S_{n+1}^+ exceeds the upper bound $\overline{d}$. In the latter case, $\mathcal{Q}$ has been proved to be empty. The following theorem establishes the correctness and provides the time complexity of this approach.

Theorem 1.

Let $\sigma : \mathbb{R}_{\geq 0}^{n+2} \to \mathbb{R}_{\geq 0}^{n+2}$ be the mapping with

$$\sigma(S) = \left(\max \left\{ \max_{i\in V}(S_i + d_{ij}), \max_{\substack{\mu=1,\ldots,\nu \\ j\in B_\mu}} \min_{i\in A_\mu} S_i \right\} \right)_{j\in V}.$$

1. σ possesses a fixpoint exactly if $\mathcal{Q} \neq \emptyset$. There is only one fixpoint S^+ with $S_0^+ = 0$. S^+ represents the unique minimal point of $\mathcal{Q}$.
2. For $\mathcal{Q} \neq \emptyset$, the fixpoint S^+ with $S_0^+ = 0$ coincides with the limit of the sequence (S^λ) with $S^1 = ES$ and $S^{\lambda+1} = \sigma(S^\lambda)$ for $\lambda \in \mathbb{Z}_{>0}$. S^+ satisfies $S_{n+1}^+ \leq \overline{d}$.
3. For $\mathcal{Q} \neq \emptyset$, there is a $\kappa \leq n\overline{d}$ with $\sigma(S^\lambda) = S^\lambda = S^+$ for all $\lambda \geq \kappa$.

The proof (cf. Schwindt 1998) relies on the monotonicity of the sequence of schedules S^λ and the existence of a unique minimal point S^+ of $\mathcal{Q}$. Notice that despite the pseudo–polynomial time complexity, the running time of the algorithm is independent of the scaling of the time lags. Moreover, the algorithm solves SCHEDULE($\mathcal{Q}$) for arbitrary regular (i.e. componentwise nondecreasing) objective functions $f(S)$.

4 Branching Strategy, Lower Bounds, and Fathoming Rules

The generation scheme of our branch–and–bound procedure is based on the algorithm for the sequencing problem SEQUENCE described in Section 2. Let $\mathcal{Q}(p)$ denote the search space belonging to enumeration node p at level ν. $\mathcal{Q}(p)$ is given by the conjunction of the temporal constraints (2) and (3) and all disjunctive precedence constraints $A_\mu \prec B_\mu$ $(\mu = 1.\ldots,\nu)$ defined along the path from the root at level 0 to node p. The enumeration is performed according to a depth–first search strategy. Initializing $\overline{d}$ as given by (8), we start with schedule $S^+ = ES$ and look for the earliest point in time t for which $\mathcal{A}(S^+,t)$ is lacking or exceeding. If S^+ is feasible, we set the current upper bound $\overline{d}$ to S^+_{n+1} and backtrack. Otherwise, we define a child node p' for each minimal delaying alternative B to $\mathcal{A}(S^+,t)$ and either determine optimal solutions $S^{p'}$ to the corresponding scheduling problems SCHEDULE($\mathcal{Q}(p')$) or establish $\mathcal{Q}(p') = \emptyset$. Instead of ES, the fix–point algorithm may start with S^+. In the latter case or if $S^{p'}_{n+1} > \overline{d}$, node p' is removed from the search tree. If all nodes p' have been deleted, we perform backtracking. Otherwise, we branch from one of the newly generated nodes p' which minimizes a nonnegative linear combination of $S^{p'}_{n+1}$ and the total inventory shortage and excess over time

$$\varphi(p') = \sum_{k\in\mathcal{R}} \int_0^{\overline{d}} \max\{0, r_k(S^{p'},t) - \overline{R}_k, \underline{R}_k - r_k(S^{p'},t)\}dt \tag{9}$$

and set $S^+ := S^{p'}$ and $p := p'$. Proceeding with the check of resource–feasibility for S^+, these steps are repeated until the feasible region has been completely explored.

The reason why (9) is taken into account when selecting the next enumeration node is that on the one hand there are generally several nodes p' with minimum objective function value $S^{p'}_{n+1}$ and on the other hand $\varphi(p')$ indicates the 'distance' of $S^{p'}$ from feasibility. Note that $S^{p'}$ is feasible exactly if $\varphi(p') = 0$.

In order to restrict the number of nodes which have to be explored during the enumeration, we use lower bounds and fathoming rules. Since $\mathcal{Q}(p)$ represents a superset of all search spaces $\mathcal{Q}(p')$ of descendant nodes p', the objective function value S^+_{n+1} of an optimal solution S^+ to SCHEDULE($\mathcal{Q}(p)$) represents a lower bound on the minimum project duration of all schedules which will be generated in the enumeration tree with root p. Let this lower bound be denoted by $lb_1(p)$. Since $lb_1(p)$ results from solving SCHEDULE($\mathcal{Q}(p)$), its computation does not require additional effort.

For problem instances with tight inventory constraints the calculation of a second lower bound $lb_2(p)$ turns out to be more expedient. Let $\tilde{d}$ be a hypothetical upper bound on the minimum project duration. Then $LS_i := \tilde{d} - d_{i,n+1}$ represents the latest occurrence time S_i of event $i \in V$ such that $S_{n+1} \leq \tilde{d}$. A lower approximation on the stock $r_k(S,t)$ of resource $k \in \mathcal{R}$ at time t for all schedules $S \in \mathcal{Q}(p)$ is given by the (not necessarily time–feasible) schedule $\underline{S}^k$ with $\underline{S}^k_i = LS_i$ if $r_{ik} > 0$ and $\underline{S}^k_i = S^+_i$, otherwise. Analogously, schedule $\overline{S}^k$ with $\overline{S}^k_i = S^+_i$ if $r_{ik} > 0$ and $\overline{S}^k_i = LS_i$, otherwise, provides an upper approximation on the stock of resource k at time t. If

$$r_k(\underline{S}^k,t) > \overline{R}_k \text{ or } r_k(\overline{S}^k,t) < \underline{R}_k \tag{10}$$

holds for a resource $k \in \mathcal{R}$ at a time $t \geq 0$, there is no feasible schedule $S \in \mathcal{Q}(p)$ with project duration $S_{n+1} \leq \tilde{d}$, and $\tilde{d}+1$ represents a valid *lower* bound on the minimum project

duration of feasible schedules $S \in \mathcal{Q}(p)$. $lb_2(p)$ corresponds to the least value $\tilde{d} \geq lb_1(p)$ which cannot be disproved by (10). For $lb_2(p) > \overline{d}$, the problem instance has been shown to be unsolvable. If the distance matrix D is given, $lb_2(p)$ can be computed in $\mathcal{O}(|\mathcal{R}|n \log n)$ time.

We use two fathoming rules to further prune the enumeration tree. Again, let p be the node under consideration and let $A \prec B$ be the disjunctive precedence constraint defined at node p. First, an event i can be removed from set A if there is a distinct event $h \in A$ with $d_{hi} > 0$. Furthermore, all remaining events $i \in A$ satisfying $\max_{j \in B} d_{ji} \geq 0$ can be transferred from set A to set B. The first rule fathoms node p if A is void or if B does no longer represent a *minimal* delaying alternative. The second rule corresponds to an adaptation of De Reyck and Herroelen's subset dominance rule to disjunctive precedence constraints. If the whole search space $\mathcal{Q}(q)$ of an enumeration node q has been explored and if $\mathcal{Q}(q)$ forms a superset of $\mathcal{Q}(p)$, node p can be fathomed. This dominance rule can be implemented efficiently exploiting the properties that (1) $\mathcal{Q}(q)$ forms a superset of all descendants' search spaces and (2) performing a depth–first–search, the parents of nodes q with inclusion–maximal completely explored search spaces are ancestors of p.

5 Truncating the Enumeration

Since the optimization problem (P) is NP–hard in the strong sense, the computation of optimal schedules to large instances including hundreds of events generally is too time–consuming. This implies the need of efficient heuristic procedures which generate good feasible solutions within a reasonable amount of time. In what follows, we sketch the truncation of the branch–and–bound algorithm to a filtered beam search. By f and $b < f$ we denote the integers corresponding to the *filter width* and the *beam width*, respectively. After the generation of all child nodes p' of current node p, we perform one iteration of the fix–point algorithm solving SCHEDULE($\mathcal{Q}(p')$) where only the disjunctive precedence constraint $A \prec B$ introduced at node p' is taken into account. Then the nodes p' are ordered according to nondecreasing values $\varphi(p')$. For the first f nodes SCHEDULE($\mathcal{Q}(p')$) is solved whereas the remaining child nodes of p are excluded from further consideration. Subsequently, the nodes p' with b least values $\varphi(p')$ are added to the enumeration tree.

For real–life problem instances even a beam width $b = 2$ may be too large. Therefore, we introduce the integer random variable

$$\tilde{b} = \max\{1, round(\overline{b}\tilde{u})\}$$

where $\tilde{u}$ is uniformly distributed in interval $[0, 1]$ and the real $\overline{b} \geq 1$ is the selected maximum beam width. The expected value of $\tilde{b}$ equals $(\overline{b}^2 + 1)/2\overline{b}$. The filter width f and the maximum beam width $\overline{b}$ are chosen according to the number n of events and the desired size of the enumeration tree.

6 Computational Results

The branch–and–bound algorithm and the filtered beam search heuristic have been coded in ANSI C. The experimental performance analysis is based on problem instances which have been randomly generated with the project generator ProGen/max. The different parameters

which control the shape of the project network, the tightness of time lags, and the resource requests are described in Schwindt (1996). Cumulative resource constraints (4) are generated as follows. First, for each pair (i,k) of events $i \in V$ and resources $k \in \mathcal{R}$ we determine whether event i requests resource k or not. The corresponding *resource factor* control parameter RF defines the percentage of pairs (i,k) belonging to a request. Second, the respective resource requirements r_{ik} are drawn randomly from set $\{r^{min},\ldots,r^{max}\}$. The scarcity of resources is controlled by the *resource strength* RS. The minimum inventory $\underline{R}_k$ and the maximum capacity $\overline{R}_k$ of resource k are given by $\underline{R}_k = RS \min_{t\geq 0} r_k(ES,t) + (1-RS)\sum_{i\in V} r_{ik}$ and $\overline{R}_k = RS \max_{t\geq 0} r_k(ES,t) + (1-RS)\sum_{i\in V} r_{ik}$. $RS = 0$ implies that the resource profiles belonging to resource–feasible schedules are constant over time, whereas for $RS = 1$ the earliest start schedule ES is feasible and thus optimal.

Our testset consists of 360 projects with 10, 20, 50, and 100 events, respectively. All instances have 5 resources. The resource factor has been chosen to $RF = 0.75$, the requirements are uniformly distributed in set $\{-2,\ldots,2\}$, and the resource strength is $RS = 0.9$. Computational experience shows that smaller RS values result in a high percentage of unsolvable instances, in particular if the temporal constraints prevent the simultaneous occurrence of a large number of events.

The tests have been performed on an Intel Pentium PII personal computer with 333 MHz clock pulse operating under Windows NT 4.0. We imposed a time limit of 30 s. The instances including $n = 50$ and $n = 100$ events have been solved by a variant of the above algorithm where the minimal delaying alternatives (whose number grows exponentially with the cardinality of the active set) are replaced by one–element subsets of the active set. For the different values of n, Table 1 shows the percentages p_{opt} of instances for which optimality could be shown, the percentages p_{uns} of instances whose unsolvability could be proved, the percentages of instances which could be solved to feasibility but not to optimality, and the percentages p_{unk} of instances whose solvability remains unknown. Δ_{lb_2} denotes the mean relative deviation of the best solution found from lower bound lb_2 where the mean is taken w.r.t. all unsolved instances for which a feasible schedule has been determined. t_{cpu} is the mean running time in seconds.

	p_{opt}	p_{uns}	p_{feas}	p_{unk}	Δ_{lb_2}	t_{cpu}
$n = 10$	66.7%	33.3%	0.0%	0.0%	—	0.072
$n = 20$	48.9%	51.1%	0.0%	0.0%	—	0.074
$n = 50$	50.0%	45.6%	2.2%	2.2%	1.4%	1.700
$n = 100$	54.4%	34.4%	8.9%	2.2%	6.9%	3.896

Table 1: Computational results

Applying filtered beam search (filter width $f = 10$, maximum beam width $\overline{b} = 3$) to the unsolved instances with 100 events, the mean relative deviation of the best schedule from lb_2 is decreased to 5.3%.

References

[1] Demeulemeester, E. and W. Herroelen (1992). A branch–and–bound procedure for the multiple resource–constrained project scheduling problem. *Management Sci.*, 38, 1803–1818.

[2] De Reyck, B. and W. Herroelen (1998). A branch–and–bound procedure for the resource–constrained project scheduling problem with generalized precedence relations. *European J. Oper. Res.*, 111, 152–174.

[3] Icmeli, O. and S.S. Erengüç (1996). A branch and bound procedure for the resource constrained project scheduling problem with discounted cash flows. *Management Sci.*, 42, 1395–1408.

[4] Neumann, K. (1999). Project scheduling with limited cumulative resources: modelling and basic concepts and results. *Proceedings of IEPM '99*.

[5] Schwindt, C. (1996). Generation of resource–constrained project scheduling problems with minimal and maximal time lags. *Report WIOR–489*, University of Karlsruhe.

[6] Schwindt, C. (1998). A branch–and–bound procedure for the resource–constrained project duration problem subject to temporal constraints. *Report WIOR–544*, University of Karlsruhe, submitted to INFORMS Journal on Computing.