DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

ONDERZOEKSRAPPORT NR 9657

An Optimal Procedure for the Resource-Constrained Project Scheduling Problem with Discounted Cash Flows and Generalized Precedence Relations

by

Bert De Reyck

Willy Herroelen

Katholieke Universiteit Leuven

Naamsestraat 69, B-3000 Leuven

ONDERZOEKSRAPPORT NR 9657

An Optimal Procedure for the Resource-Constrained Project Scheduling Problem with Discounted Cash Flows and Generalized Precedence Relations

by

Bert De Reyck
Willy Herroelen

D/1996/2376/57

AN OPTIMAL PROCEDURE FOR THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM WITH DISCOUNTED CASH FLOWS AND GENERALIZED PRECEDENCE RELATIONS

Bert DE REYCK

Willy HERROELEN

October 1996

Operations Management Group
Department of Applied Economics
Katholieke Universiteit Leuven
Hogenheuvel College
Naamsestraat 69, B-3000 Leuven, Belgium
Phone: 32-16-32 69 66 or 32-16-32 69 70
Fax: 32-16-32 67 32
E-mail: Bert.DeReyck@econ.kuleuven.ac.be or Willy.Herroelen@econ.kuleuven.ac.be
WWW-page: http://econ.kuleuven.ac.be/tew/academic/om/people/bert
http://econ.kuleuven.ac.be/tew/academic/om/people/willy

AN OPTIMAL PROCEDURE FOR THE RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEM WITH DISCOUNTED CASH FLOWS AND GENERALIZED PRECEDENCE RELATIONS

Bert De Reyck • Willy Herroelen

Department of Applied Economics, Katholieke Universiteit Leuven

ABSTRACT

In this paper, we study the resource-constrained project scheduling problem (RCPSP) with discounted cash flows and generalized precedence relations (further denoted as RCPSPDC-GPR). The RCPSPDC-GPR extends the RCPSP to (a) arbitrary minimal and maximal time lags between the starting and completion times of activities and (b) the non-regular objective function of maximizing the net present value of the project with positive and/or negative cash flows associated with the activities.). To the best of our knowledge, the literature on the RCPSPDC-GPR is completely void. We present a depth-first branch-and-bound algorithm in which the nodes in the search tree represent the original project network extended with extra precedence relations which resolve a number of resource conflicts. These conflicts are resolved using the concept of a minimal delaying mode (De Reyck and Herroelen, 1996b). An upper bound on the project net present value as well as several dominance rules are used to fathom large portions of the search tree. Extensive computational experience on a randomly generated benchmark problem set is obtained.

1. Introduction

CPM (Critical Path Method; Kelley and Walker, 1959) and PERT (Program Evaluation and Review Technique; Malcolm et al., 1959) are devoted to minimizing the project makespan under the assumption that required resources are available in sufficient amounts, and that the technological precedence relations between any pair of activities i and j imply that activity i must be completed before activity j can be initiated. Over the years, the assumption of sufficiently available resources has been relaxed and many research efforts have been directed towards project scheduling with explicit consideration of resource requirements and constraints. More recent research has been directed at relaxing the strict precedence assumption of CPM/PERT. In accordance with Elmaghraby and Kamburowski (1992), we will refer to the resulting types of precedence relations as *generalized precedence relations* (*GPRs*). We distinguish between four types of GPRs: start-start (SS), start-finish (SF), finish-start (FS) and finish-finish (FF).

GPRs can specify a minimal or a maximal time lag between a pair of activities. A minimal time lag specifies that an activity can only start (finish) when the predecessor activity has already started (finished) for a certain time period. A maximal time lag specifies that an activity should be started (finished) at the latest a certain number of time periods beyond the start (finish) of another activity. GPRs can be used to model a wide variety of specific problem characteristics, including (Bartusch et al., 1988; De Reyck, 1995b; Neumann and Schwindt, 1995) activity ready times and deadlines, activities that have to start (terminate) simultaneously, non-delay execution of activities, (total or strong/weak partial) activity overlaps, fixed activity starting times, time-varying resource requirements and availabilities, time-windows for resources, inventory restrictions, setup times, overlapping production activities (process batches, transfer batches) and assembly line zoning constraints. The first treatment of GPRs is due to Kerbosch and Schell (1975), based on the pioneering work of Roy (1962). Other studies include Crandall (1973), Elmaghraby (1977), Wiest (1981), Moder et al. (1983), Bartusch et al. (1988), Elmaghraby and Kamburowski (1992), Brinkmann and Neumann (1994), Zhan (1994), De Reyck (1995a, 1995b), Neumann and Schwindt (1995) and Schwindt (1995), Neumann and Zhan (1995), De Reyck and Herroelen (1996b, 1996c, 1996d), Schwindt and Neumann (1996) and Franck and Neumann (1996).

Recently, a number of publications have dealt with various types of project scheduling problems in which cash flows are associated with the activities, and in which the objective is to schedule the activities in such a way that the net present value (*npv*) of the project is maximized. Generally, a series of cash flows may occur over the course of a project in two forms. Cash outflows include expenditures for labor, equipment, materials, etc.. Cash inflows correspond to progress payments for completed work. For a recent review and categorization of the various research efforts, we refer the reader to Herroelen et al. (1996a). We distinguish between procedures for the *unconstrained max-npv project scheduling problem*, which occurs when no constraints on the resource usage are imposed such that the activities are only subject to

precedence constraints, and procedures for the resource-constrained project scheduling problem with max-*npv* objective, also referred to as the *resource-constrained project scheduling problem with discounted cash flows* (RCPSPDC). Algorithms for the deterministic resource-unconstrained case have been presented by Russell (1970), Grinold (1972), Elmaghraby and Herroelen (1990), Herroelen and Gallens (1993), Kazaz and Sepil (1996) and Herroelen et al. (1996b), among which the latter seems to be the most efficient. Optimal algorithms for the resource-constrained case have been presented by Doersch and Patterson (1977), Smith-Daniels and Smith-Daniels (1987), Patterson et al. (1990a,1990b), Yang et al. (1992), Icmeli and Erengüç (1996) and Baroum and Patterson (1996). Heuristic approaches have been presented by Russell (1986), Smith-Daniels and Aquilano (1987), Padman et al. (1990), Padman and Smith-Daniels (1993), Zhu and Padman (1993), Icmeli and Erengüç (1994), Özdamar et al. (1994), Yang et al. (1995), Ulusoy and Özdamar (1995) and Sepil and Ortaç (1995).

In this paper, we present an optimal solution procedure for the resource-constrained project scheduling problem with discounted cash flows *and* generalized precedence relations (further denoted as RCPSPDC-GPR), thereby extending both the procedures presented in the literature for the resource-constrained project scheduling problem with generalized precedence relations (further denoted as RCPSP-GPR) as well as those for the resource-constrained project scheduling problem with discounted cash flows (further denoted as RCPSPDC). To the best of our knowledge, the literature on the RCPSPDC-GPR is completely void. In fact, all optimal and heuristic procedures for the RCPSP with GPRs have so far concentrated on minimizing the project makespan or optimizing other *regular* measures of performance (Brinkmann and Neumann, 1994; Zhan, 1994; Neumann and Zhan, 1995; De Reyck and Herroelen, 1996b, 1996c; Schwindt and Neumann, 1996; Franck and Neumann, 1996).

The remainder of this paper is organized as follows. Section 2 clarifies the terminology and the project representation used. In section 3, the temporal analysis of project networks with generalized precedence relations is briefly reviewed. Section 4 continues with a conceptual formulation of the RCPSP-GPR. In section 5 we review the optimal algorithm of De Reyck and Herroelen (1996d) for the (resource-) unconstrained max-*npv* project scheduling problem with GPRs, which will be used for the computation of upper and lower bounds on the project *npv* in the branch-and-bound procedure for the RCPSPDC-GPR, which will be described in section 6. Computational results are given in section 7. Section 8 is reserved for our overall conclusions.

2. Terminology and representation

Assume a project represented in activity-on-node (AoN) notation by a directed graph $G = \{V, E\}$ in which V is the set of vertices or activities, and E is the set of edges or generalized precedence relations (GPRs). The project is subject to a deadline D. The non-preemptable activities are numbered from 1 to n, where the dummy activities 1 and n mark the beginning and the end of the project. The duration of an activity is denoted by $d_i (1 \leq i \leq n)$, its starting time by

$s_i (1 \le i \le n)$ and its finishing time by $f_i (1 \le i \le n)$. The terminal cash flow value c_i (positive or negative) of activity i is obtained by compounding all the cash flows occurring during the execution of activity i to its completion time: $c_i = \sum_{t=1}^{d_i} f_{it}\, e^{\alpha(d_i - t)}$, where f_{it} (positive or negative) denotes the cash flow occurring during the t^{th} period activity i is in progress and α represents the discount rate. There are m *renewable* resource types, with r_{ikx} $(1 \le i \le n, 1 \le k \le m, 1 \le x \le d_i)$ the resource requirements of activity i with respect to resource type k in the x^{th} period it is in progress and $a_{kt} (1 \le k \le m; 1 \le t \le T)$ the availability of resource type k in time period $]t\text{-}1, t]$ (T is an upper bound on the project length). If the resource requirements and availabilities are not time-dependent, they are represented by r_{ik} $(1 \le i \le n, 1 \le k \le m)$ and $a_k (1 \le k \le m)$ respectively. The minimal and maximal time lags between two activities i and j have the form:

$$s_i + SS_{ij}^{\min} \le s_j \le s_i + SS_{ij}^{\max}; \qquad s_i + SF_{ij}^{\min} \le f_j \le s_i + SF_{ij}^{\max}$$
$$f_i + FS_{ij}^{\min} \le s_j \le f_i + FS_{ij}^{\max}; \qquad f_i + FF_{ij}^{\min} \le f_j \le f_i + FF_{ij}^{\max}$$

and can be represented in a *standardized form* by reducing them to just one type, e.g. the minimal start-start precedence relations, using the following transformation rules (Bartusch et al., 1988):

$$\begin{aligned}
s_i + SS_{ij}^{\min} \le s_j &\Rightarrow s_i + l_{ij} \le s_j \quad \text{with} \quad l_{ij} = SS_{ij}^{\min} \\
s_i + SS_{ij}^{\max} \ge s_j &\Rightarrow s_j + l_{ji} \le s_i \quad \text{with} \quad l_{ji} = -SS_{ij}^{\max} \\
s_i + SF_{ij}^{\min} \le f_j &\Rightarrow s_i + l_{ij} \le s_j \quad \text{with} \quad l_{ij} = SF_{ij}^{\min} - d_j \\
s_i + SF_{ij}^{\max} \ge f_j &\Rightarrow s_j + l_{ji} \le s_i \quad \text{with} \quad l_{ji} = d_j - SF_{ij}^{\max} \\
f_i + FS_{ij}^{\min} \le s_j &\Rightarrow s_i + l_{ij} \le s_j \quad \text{with} \quad l_{ij} = d_i + FS_{ij}^{\min} \\
f_i + FS_{ij}^{\max} \ge s_j &\Rightarrow s_j + l_{ji} \le s_i \quad \text{with} \quad l_{ji} = -d_i - FS_{ij}^{\max} \\
f_i + FF_{ij}^{\min} \le f_j &\Rightarrow s_i + l_{ij} \le s_j \quad \text{with} \quad l_{ij} = d_i - d_j + FF_{ij}^{\min} \\
f_i + FF_{ij}^{\max} \ge f_j &\Rightarrow s_j + l_{ji} \le s_i \quad \text{with} \quad l_{ji} = d_j - d_i - FF_{ij}^{\max}
\end{aligned}$$

To ensure that the dummy start and finish activities correspond to the beginning and the completion of the project, we assume that there exists at least one path with nonnegative length from node 1 to every other node i and at least one path from every node i to node n which is equal to or larger than d_i. If there are no such paths, we can insert arcs $(1,i)$ or (i,n) with weight zero and d_i respectively. $P(i) = \{j \mid (j,i) \in E\}$ is the set of all *immediate predecessors* of node i, $Q(i) = \{j \mid (i,j) \in E\}$ is the set of all its *immediate successors*. If there exists a path from i to j, then we call i a (not necessarily *immediate*) *predecessor* of j and j a *successor* of i. If the length of the longest path from i to j is positive or all arcs of a longest path are associated with a lag of zero, node i is called a *real* (immediate) predecessor of node j, and j is called a *real* (immediate) successor of i. Otherwise it is a *fictitious* one.

3. Temporal analysis in project networks with generalized precedence relations

A schedule $S = \{s_1, s_2, ..., s_n\}$ is called *time-feasible*, if the starting times satisfy all GPRs. The minimum starting times representing a time-feasible schedule form the *early start schedule* $ESS = \{es_1, es_2, ..., es_n\}$. The calculation of an ESS can be related to the test for existence of a time-feasible schedule. The earliest start of an activity i can be calculated by finding the longest path from node 1 to node i. We also know that there exists a time-feasible schedule for G iff G has no cycle of positive length (Bartusch et al., 1988). Therefore, if we calculate the distance matrix $D = [d_{ij}]$, where d_{ij} denotes the maximal distance (path length) from node i to node j, a positive path length from node i to itself indicates the existence of a cycle of positive length and, consequently, the non-existence of a time-feasible schedule. The calculation of the distance matrix D can be done by standard graph algorithms for longest paths in (cyclic) networks, for instance by the Floyd-Warshall algorithm (for details, see Lawler, 1976), which takes $O(n^3)$ time.

4. The RCPSP-GPR

4.1. Definition

The resource-constrained project scheduling problem with generalized precedence relations (RCPSP-GPR) can be conceptually formulated as follows:

Minimize s_n [1]

Subject to

$s_i + l_{ij} \le s_j \quad \forall (i,j) \in E$ [2]

$\sum_{i \in S(t)} r_{ik} \le a_{kt} \quad k = 1,2,\ldots,m \qquad t = 1,2,\ldots,T$ [3]

$s_1 = 0$ [4]

$s_i \in \mathrm{N} \quad i = 1,2,\ldots,n$ [5]

where N denotes the set of natural numbers, $S(t)$ is the set of activities in progress in time period $]t\text{-}1, t]$ and T is an upper bound on the project duration, for instance $T = \sum_{i \in V} \max\left\{d_i, \max_{j \in Q(i)}\{l_{ij}\}\right\}$.

Note that it is not always possible to derive a feasible solution. The upper bound T indicates the maximal value for the project makespan if a feasible solution exists. The objective function given in Eq. 1 minimizes the project duration, given by the starting time (or finishing time, since $d_n = 0$) of the dummy activity n. The precedence constraints are denoted in standardized form by Eqs. 2. Eqs. 3 represent the resource constraints. The resource requirements and availabilities are assumed to be constant over time, although this assumption can be relaxed using GPRs without having to change the solution procedures. Time-varying resource requirements can be modelled

by splitting up the activities in a number of subactivities with a different *constant* resource requirement for each of the resource types. The subactivities should then be connected with minimal and maximal zero-lag finish-start precedence relations, which ensure a non-delay execution of all the subactivities of each activity. Time-varying resource availabilities can be handled by creating dummy activities which absorb a certain amount of each resource type for which a constant availability (equal to the maximum availability over time of that resource type) can then be assumed. These dummy activities should then be assigned a fixed starting time using a minimal and maximal time lag between the dummy activity representing the start of the project and the dummy activity in question (which corresponds to a ready time and a deadline which are equal). Eq. 4 forces the dummy start activity to begin at time zero and Eqs. 5 ensure that the activity starting times assume nonnegative integer values. Once started, activities run to completion (no preemption).

The RCPSP-GPR is known to be strongly NP-hard, and even the decision problem of testing whether a RCPSP-GPR instance has a feasible solution is NP-complete (Bartusch et al., 1988). Optimal procedures for the RCPSP-GPR have been presented by Bartusch et al. (1988) and De Reyck and Herroelen (1996b). Heuristic procedures have been presented by Zhan (1994), Brinkmann and Neumann (1994), Neumann and Zhan (1995), Franck and Neumann (1996) and Schwindt and Neumann (1996).

5. The unconstrained max-npv project scheduling problem with GPRs

The unconstrained max-*npv* project scheduling problem with GPRs involves the scheduling of project activities subject to GPRs in order to maximize the net present value (*npv*) of the project, under the assumption that no constraints on the usage of resources are imposed. Each activity has a terminal cash flow c_i, which can be positive or negative. A conceptual formulation of the unconstrained max-*npv* project scheduling problem can be formulated as follows:

$$\text{Maximize} \quad \sum_{i=2}^{n-1} c_i \, e^{-\alpha(s_i+d_i)} \qquad [6]$$

Subject to

$$s_i + l_{ij} \leq s_j \qquad \forall (i,j) \in E \qquad [7]$$

$$s_1 = 0 \qquad [8]$$

$$s_n \leq D \qquad [9]$$

$$s_i \in \mathrm{N} \qquad i = 1,2,\ldots,n \qquad [10]$$

The objective function in Eq. 6 maximizes the *npv* of the project. The constraint set given in Eq. 7 maintains the GPRs among the activities. Eq. 8 forces the dummy start activity to begin at time zero and Eq. 9 limits the project duration to a negotiated deadline. Eqs. 10 ensure that the activity starting times assume nonnegative integer values.

We will briefly review the optimal procedure developed by De Reyck and Herroelen (1996d), which will be used for the computation of upper and lower bounds on the project *npv* in the branch-and-bound procedure for the RCPSPDC-GPR, to be described in the next section. A detailed description of the procedure is given in Appendix 1. More detailed information and extensive computational experience can be found in De Reyck and Herroelen (1996d).

We start in STEP 1 by computing the constraint digraph using the transformation rules discussed in section 2 (time complexity $O(|E|)$). The distance matrix is computed using the Floyd-Warshall algorithm (time complexity $O(n^3)$). If the project is not time-feasible, i.e. if there is an activity i for which $d_{ii} > 0$, the algorithm stops. Otherwise, in STEP 2, the *early tree*, which spans all activities (nodes) scheduled at their earliest start time, is computed as follows ($O(n^2)$). For every activity i, a predecessor j is determined for which $d_{1j} + d_{ji} = d_{1i}$, upon which activities j and i are linked. For every activity i, there always exists a predecessor activity j satisfying $d_{1j} + d_{ji} = d_{1i}$, since dummy activity 1 will always satisfy this constraint for any given activity i. In other words, if we would link activity 1 to every other activity, we would get a valid early tree. However, this early tree contains very little information about the activities in the project and their precedence relations (e.g. critical paths) and would lead to a large number of 'unnecessary' recursion steps later on in the procedure. Therefore, we link every activity i to the highest numbered predecessor j ($j<i$) for which $d_{1j} + d_{ji} = d_{1i}$ holds (using a reverse search scheme).

The *current tree* is calculated in STEP 3 of the algorithm ($O(n^2)$) by delaying, in reverse order, all activities i with a negative cash flow and no successor in the early tree as much as possible within the early tree, i.e. without affecting the start times of the successor activities in the constraint digraph. Each such activity i is then linked to its successor j restricting a further delay of activity i, except for the case where activity j is itself a predecessor of activity i in the current tree (which is possible because activity networks with GPRs can contain cycles), which would lead to the creation of a cycle in the current tree. In that case, activities i and j remain fixed at their current starting times because only a simultaneous delay of both activities would ensure that the time-feasibility of the project network is not violated. Simultaneous delays will be examined in STEP 4.

If any activity i has been delayed while calculating the current tree, STEP 3 has to be repeated, since it is possible that delaying activity i will allow for an additional delay of another activity j ($j>i$). Searching in reverse order makes sure that no other activity $j<i$ will be delayed, but the delay of activities $j>i$ cannot always be avoided.

After STEP 3 has been repeated a sufficient number of times (worst-case scenario: n times, making the time complexity of STEP 3, including its repetitions, $O(n^3)$), the procedure will enter a recursive search, in which partial trees *PT* (with a negative *npv*) will be identified that may be shifted forwards in time in order to increase the *npv* of the project. When such a partial tree is

found, the algorithm computes the maximal shift of the partial tree by identifying the maximal possible increase in the starting times of the activities belonging to the partial tree without violating any of the precedence constraints, keeping all activities not belonging to PT at their current starting times. Therefore, we look for a new arc with minimal displacement, i.e. an arc (k,l) ($k \in PT, l \notin PT$) with minimal value for $d_{1l} - d_{1k} - d_{kl}$. We disconnect the partial tree from the remainder of the current tree and we add the arc (k,l) to the current tree, thereby relinking the forward-shifted partial tree to the current tree. The completion times of the activities in the partial tree are updated as follows: $\forall j \in PT$: $d_{1j} = d_{1j} + \min_{\substack{k \in PT \\ l \notin PT}} \{d_{1l} - d_{1k} - d_{kl}\}$. If a shift has been found and implemented, the recursive procedure is restarted until no further shift can be accomplished. Then, the optimal schedule with its corresponding *npv* is reported.

6. The RCPSPDC-GPR

6.1. Definition

The resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations (RCPSPDC-GPR) can be conceptually formulated as follows:

$$\text{Maximize} \sum_{i=2}^{n-1} c_i \, e^{-\alpha(s_i+d_i)} \qquad [11]$$

Subject to

$$s_i + l_{ij} \le s_j \qquad \forall (i,j) \in E \qquad [12]$$

$$\sum_{i \in S(t)} r_{ik} \le a_{kt} \qquad k = 1,2,\ldots,m \qquad t = 1,2,\ldots,T \qquad [13]$$

$$s_1 = 0 \qquad [14]$$

$$s_n \le D \qquad [15]$$

$$s_i \in \mathrm{N} \qquad i = 1,2,\ldots,n \qquad [16]$$

The objective function in Eq. 11 maximizes the *npv* of the project. The constraint set given in Eq. 12 maintains the GPRs among the activities. Eqs. 13 represent the resource constraints. Eq. 14 forces the dummy start activity to begin at time zero and Eq. 15 limits the project duration to a negotiated deadline. Eqs. 16 ensure that the activity starting times assume nonnegative integer values. As an extension of the RCPSP-GPR or the RCPSPDC, the RCPSPDC-GPR is clearly NP-hard in the strong sense. If, for instance, only a positive cash flow is associated with the dummy end activity, the problem reduces to the RCPSP-GPR since the objective then is to minimize the completion time of the dummy end activity, i.e. minimizing the project makespan. If all precedence relations are of the zero-lag finish-start type, the problem reduces to the RCPSPDC. Also the corresponding feasibility problem (the decision problem of testing whether a RCPSPDC-GPR instance has a feasible solution) is NP-complete.

To the best of our knowledge, an algorithm for project scheduling with resource constraints, discounted cash flows as well as GPRs has not yet been presented in the literature. Algorithms for the RCPSPDC with zero-lag finish-start precedence constraints only do exist, the most efficient of which seems to be the branch-and-bound procedure of Icmeli and Erengüç (1996). Their algorithm extends the procedure of Demeulemeester and Herroelen (1992, 1996) originally developed for the RCPSP, by adapting the branching strategy to cope with the max-*npv* criterion. The authors use the procedure of Grinold (1972) for the unconstrained max-*npv* project scheduling problem to compute upper and lower bounds on the project *npv*. In the next section, we will present a branch-and-bound procedure for the RCPSPDC-GPR based on the concepts developed in De Reyck and Herroelen (1996b) for the RCPSP-GPR and on the procedure for the unconstrained max-*npv* project scheduling problem with GPRs of De Reyck and Herroelen (1996d).

6.2. A branch-and-bound procedure

6.2.1. The search tree

The nodes in the search tree represent the initial project network, described by a distance matrix $D = [d_{ij}]$, extended with extra (zero-lag finish-start) precedence relations to resolve a number of resource conflicts, which results in an *extended* distance matrix. Nodes which represent time-feasible (no violated maximal time lags) but resource-infeasible project networks and which are not fathomed by any node fathoming rules described below lead to a new branching. Therefore each (undominated) node represents a time-feasible, but not necessarily resource-feasible project network. Resource conflicts are resolved using the concept of *minimal delaying alternatives*, i.e. minimal sets of activities which, when delayed, release enough resources to resolve the resource conflict (and which do not contain any other delaying alternative as a subset). Each of these minimal delaying alternatives is then delayed (enforced by extra zero-lag finish-start precedence relations $i \prec j$, implying $s_i + d_i \leq s_j$) by each of the activities also belonging to the *conflict set* $S(t^*)$, the set of activities in progress in period $]t^*\text{-}1, t^*]$ (the period of the *first* resource conflict), but not belonging to the delaying alternative. Therefore, each minimal delaying alternative can give rise to several *minimal delaying modes*.

A similar delaying strategy was used by Demeulemeester and Herroelen (1992) for the RCPSP. As the RCPSP can be solved using semi-active timetabling to construct the partial schedules, activities belonging to the minimal delaying alternative can be delayed by the activity in $S(t^*)$ which terminates at the earliest time instant beyond the current decision point (further denoted as the *delaying activity*). In the RCPSP-GPR, this delaying strategy cannot be used because of the maximal time lags, which make semi-active timetabling inappropriate. These time lags make it impossible to determine which activity in $S(t^*)$ should be used as the delaying activity, because we cannot predict in advance which activity in $S(t^*)$ will terminate the earliest

in the feasible schedules that will be obtained by branching from the current project network. Therefore, in the RCPSP-GPR, we have to consider several possible *delaying modes* for each delaying alternative.

In general, the *delaying set* D, i.e. the set of all minimal delaying alternatives, is equal to $D = \left\{ D_d \middle| D_d \subset S(t^*) \text{ and } \forall \text{ resource type } k : \sum_{i \in S(t^*)} r_{ik} - \sum_{i \in D_d} r_{ik} \leq a_k \text{ and } \forall D_{d'} \in \mathrm{D} : D_{d'} \not\subset D_d \right\}$. The set M of minimal delaying modes equals: $M = \left\{ M_m \middle| M_m = \{k \prec D_d\}, k \in S(t^*) \setminus D_d, D_d \in D \right\}$. Activity k is called the *delaying activity*: $k \prec D_d$ implies that $k \prec l$ for all $l \in D_d$.

THEOREM 1. The delaying strategy which consists of delaying all minimal delaying alternatives D_d by each activity $k \in S(t^*) \setminus D_d$ will lead to the optimal solution of the RCPSPDC-GPR in a finite number of steps.

PROOF. See De Reyck and Herroelen (1996b).

6.2.2. Branching strategy

Each time-feasible minimal delaying mode with an upper bound *ub* (computed using the procedure for the unconstrained max-*npv* project scheduling problem with GPRs of De Reyck and Herroelen, 1996d) higher than an already obtained lower bound *lb* on the project *npv* is considered for further branching. If the node represents a project network in which a resource conflict occurs, a new branching occurs. If it represents a feasible schedule, the lower bound *lb* is updated and the procedure backtracks to the previous level in the search tree. Therefore, we have a depth-first search procedure, in which branching occurs until at a certain level in the tree, there are no delaying modes left to branch from. Then, the procedure backtracks to the previous level in the search tree and reconsiders the other delaying modes pending at that level. The procedure stops when it backtracks to level 0.

The computation of the upper bound *ub* on the project *npv* is not made upon creation of a node, but is deferred until a decision has been made to actually branch from it. The rationale behind this is that computing *ub* implies calculating the entire distance matrix, which is a time- and memory-consuming procedure. Supported by extensive computational tests, we defer the calculation of *ub* and the distance matrix until the node is actually selected for branching. As a result, another criterion will have to be used in order to select the node to branch from at a certain level. A node should be selected for branching when it entails a high chance of finding a feasible solution with a high *npv*. Therefore, delaying activities which carry negative cash flows is to be preferred to delaying activities with positive cash flows since the latter will negatively affect the *npv* of the project. In general, the higher (more positive) the cash flows of the delayed activities, the more likely the project *npv* will decrease. Therefore, we have chosen as a branching criterion the sum of the cash flows associated with the activities that have to be delayed, the

smallest sum being chosen first. Extensive experiments have revealed that a more sophisticated way of estimating the effect of the delay of activities on the project *npv* does not yield better results than choosing the node with the smallest sum of activity cash flows of the delayed activities.

6.2.3. Node fathoming rules

Nodes are fathomed when they represent a time-infeasible project network or when their *ub* does not exceed an already obtained lower bound *lb*. Nodes which are not fathomed and still represent an infeasible project network are considered for further branching. Three additional node fathoming rules (Theorems 2, 3 and 4) and a procedure which reduces the solution space and which can be executed as a preprocessing rule (Theorem 5) are added. These node fathoming rules are similar to the ones developed in De Reyck and Herroelen (1996b) for the RCPSP-GPR. Therefore, they will only be stated here without further explanation or proof.

THEOREM 2. If there exists a minimal delaying alternative D_d *with activity* $i \in D_d$ *but its real successor* $j \notin D_d$ $(d_{ij} \geq 0)$*, we can extend* D_d *with activity j. If the resulting delaying alternative becomes non-minimal as a result of this operation, it may be eliminated from further consideration.*

THEOREM 3. When a minimal delaying alternative D_d *gives rise to two delaying modes* M_{m_1} *and* M_{m_2} *with delaying activities i and j* respectively, *mode* M_{m_2} *is dominated by mode* M_{m_1} *iff* $d_{ij} + d_j \geq d_i$.

THEOREM 4. If the set of added precedence constraints which leads to the project network (in the form of an extended distance matrix) in node x contains as a subset another set of precedence constraints leading to the project network (extended distance matrix) in a previously examined node y in another branch of the search tree, node x can be fathomed.

THEOREM 5. If $\exists\, i, j \in V$ *and resource type k for which* $r_{ik} + r_{jk} > a_k$ *and* $-d_j < d_{ij} < d_i$*, we can set* $l_{ij} = d_i$ *without changing the optimal solution of the RCPSPDC-GPR.*

The detailed algorithmic steps of the branch-and-bound procedure are given in Appendix 2.

7. Computational experience

7.1. Benchmark problem set

Schwindt (1995) developed a problem generator ProGen/max which can randomly generate instances of various types of generalized resource-constrained project scheduling problems, based on the problem generator ProGen for the RCPSP developed by Kolisch et al. (1995). Two methods are proposed: DIRECT, which directly generates entire projects, and CONTRACT, which first generates cycle structures, upon which the (acyclic) contracted project network is generated. Several control parameters can be specified, as indicated in Table I.

Table I. The control parameters of ProGen/max (Schwindt, 1995)

Problem size-based	Resource-based	Acyclic network-based	Cyclic network-based
# activities (n)	# resource types (m)	# initial and terminal activities	% maximal time lags
	min. / max. number of resources used per activity	maximal # predecessors and successors	# cycle structures
	resource factor (RF) (Pascoe, 1966)	order strength (OS)[1] (Mastor, 1970)	min. / max. # nodes per cycle structure
	resource strength (RS) (Kolisch et al., 1995)		coefficient of cycle structure density (Schwindt, 1995)
			cycle structure tightness (Schwindt, 1995)

We generated 5760 problem instances using the DIRECT method using the control parameters given in Table II. For each combination of control parameter values, 120 problem instances have been generated. The indication $[x,y]$ means that the value is randomly generated in the interval $[x,y]$, whereas x; y; z means that three settings for that parameter were used in a full factorial experiment. The parameters used in the full factorial experiment are the number of activities as a problem size-based measure, the order strength (OS) as an acyclic network-based

[1] Schwindt (1996) uses an estimator for the restrictiveness (Thesen, 1977) as a network complexity measure. However, De Reyck (1995c) has shown that this measure is identical to the order strength (Mastor, 1970), the flexibility ratio (Dar-El, 1973) and the density (Kao and Queyranne, 1982). We will use *order strength* when referring to this measure.

measure and the percentage of maximal time lags as a cyclic network-based measure. The cash flows for each of the activities are generated randomly from the interval $[-500,+500]$.

Table II. The parameter settings of the benchmark problem set

Control parameter	Value
# activities	10; 20; 30; 50
activity durations	[2,10]
# initial and terminal activities	[2,4]
maximal # predecessors and successors	3
OS	0.25; 0.50; 0.75
% maximal time lags	0%; 10%; 20%; 30%
# cycle structures	[0,10]
minimal / maximal # nodes per cycle structure	2 / 100
coefficient of cycle structure density	0.3
cycle structure tightness	0.5

7.2. The RCPSPDC-GPR results

The procedure has been programmed in Microsoft® Visual C++ 2.0 under Windows NT for use on a Digital Venturis Pentium-60 personal computer. The code itself requires 109Kb of memory, whereas 10Mb are reserved for the storage of the search tree. Solving all problem instances of the problem set to optimality (especially the 50-activity problem instances) is probably beyond the capabilities of current branch-and-bound procedures. Even for the classic RCPSP, the RCPSPDC or the RCPSP-GPR, problem instances with 50 activities are not amenable to optimal solution within acceptable computational effort. As an example, the RCPSP problem set of Kolisch et al. (1995), which consists of 480 instances with 30 activities has only recently been solved to optimality by Demeulemeester and Herroelen (1996). Therefore, given the higher complexity of the RCPSPDC-GPR, it is to be expected that a similar set consisting of 30-activity RCPSPDC-GPR instances will not be solved to optimality within acceptable computation times.

7.2.1. Basic results

We report in Table III the results of our procedure, when truncated after some seconds of running time. The reported values include the number of problems solved to optimality (for which the optimum was found and verified, including the problems proven to be infeasible), the number of unsolved problems (for which a feasible solution was not obtained within the given time limit) and the average CPU-time. Fig. 1 displays the significant effect of the problem size on the number of problems solved to optimality.

Table III. The results with the truncated version of the branch-and-bound procedure

Time limit	1 second	10 seconds	100 seconds
Problems solved to optimality	4155 (±72%)	4836 (±84%)	5244 (±91%)
Unsolved problems	24 (±0.4%)	13 (±0.2%)	7 (±0.1%)
Average CPU-time (in seconds)	0.40	2.10	12.08

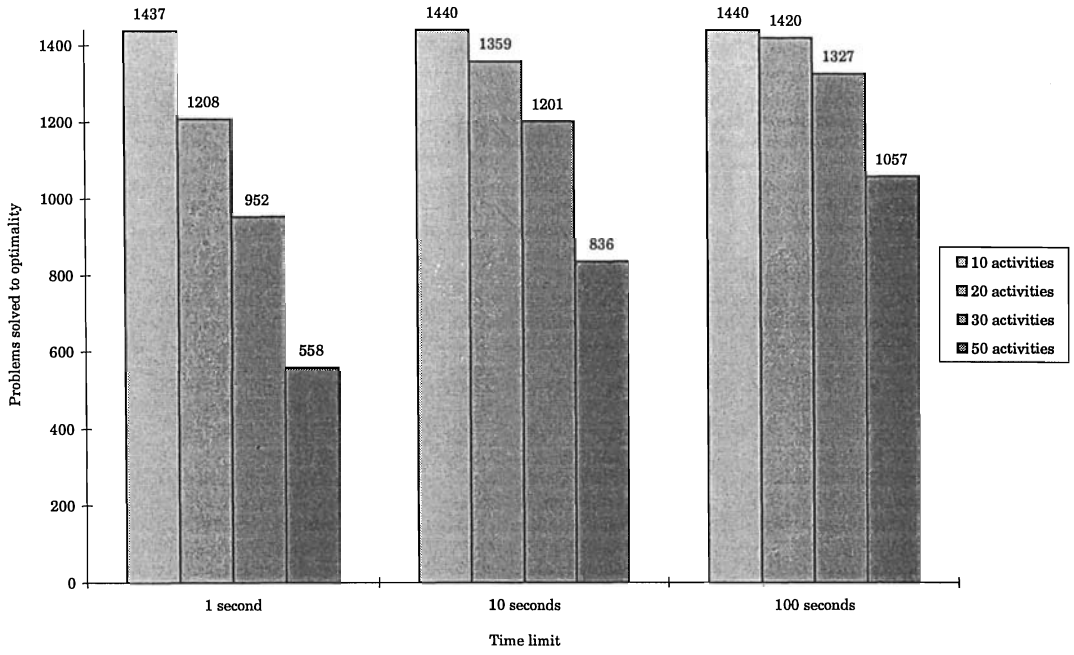

Fig. 1. The effect of problem size on the number of problems solved to optimality

7.2.2. The impact of *OS*

Fig. 2 displays the impact of *OS* on the RCPSPDC-GPR complexity. It was already established that, for the RCPSP-GPR, *OS* has a negative correlation with the problem hardness, that is, the higher *OS*, the easier the corresponding RCPSP-GPR instance (De Reyck and Herroelen, 1996c). However, for the unconstrained max-*npv* project scheduling problem, an opposite effect was observed (De Reyck and Herroelen, 1996d). Therefore, it would be interesting to see how these two effects interact to determine the effect of *OS* on the computational complexity of the RCPSPDC-GPR. In accordance with the results for the RCPSP-GPR, it is to be expected that *OS* will have a negative correlation with the number of nodes in the search tree. However, the time spent per node will increase when *OS* increases, due to the increased time needed to solve the unconstrained max-*npv* project scheduling problem. Fig. 2 clearly indicates that *OS* has no significant impact on the required CPU-time to solve the RCPSPDC-GPR instances. This means that, probably, both opposite effects of *OS* neutralize each other.

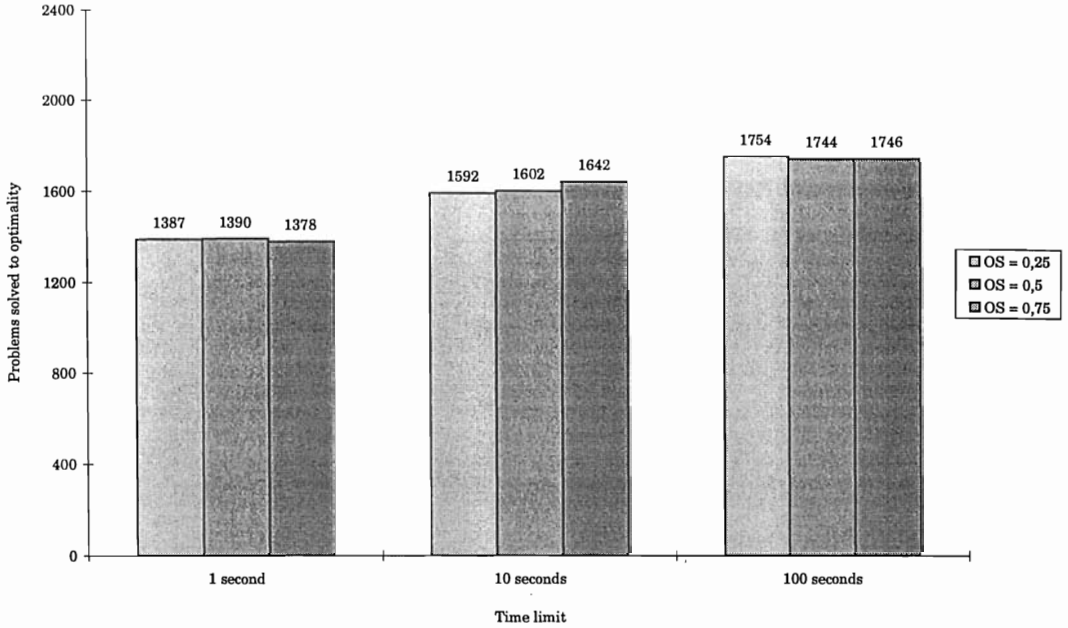

Fig. 2. The effect of *OS* on the number of problems solved to optimality

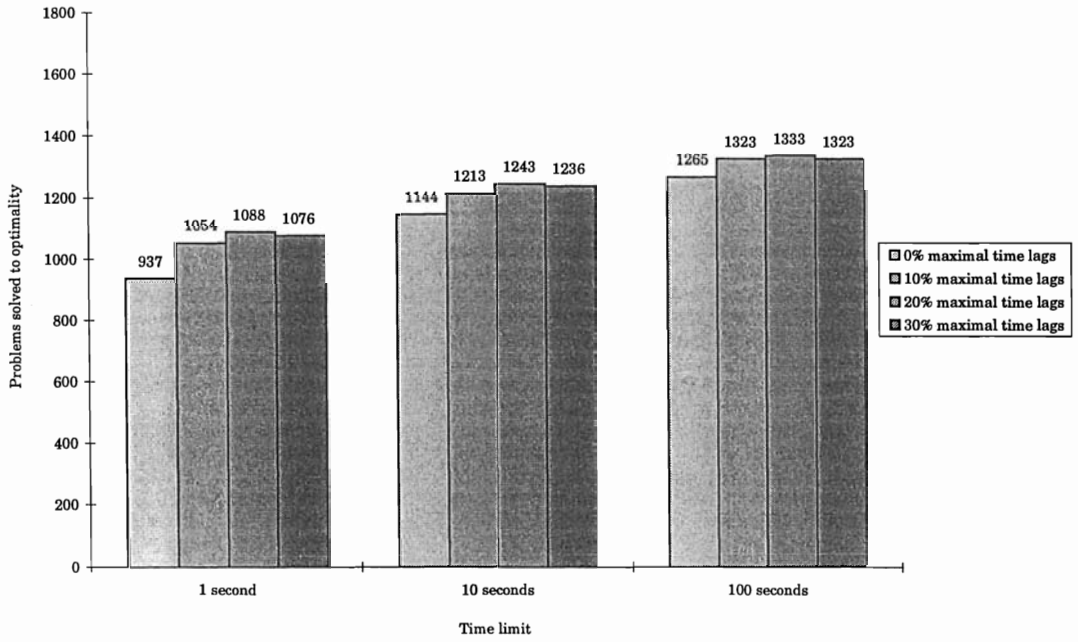

Fig. 3. The effect of % maximal time lags on the number of problems solved to optimality

7.2.3. The impact of the percentage of maximal time lags

The effect of the percentage of maximal time lags (Fig. 3) on the computational complexity of the RCPSPDC-GPR is neither monotonously increasing nor decreasing. On the contrary, a kind of bell-shaped curve seems to result. When maximal time lags are introduced, the number of problems solved to optimality increases up to a certain point, beyond which it decreases again.

The initial rise in performance can be understood if we remember that, in the branch-and-bound procedure, several dominance rules and lower bounds are used which require the existence of maximal time lags in order to be applicable. This makes the procedure more effective and efficient when such time lags are introduced. However, when there are many maximal time lags, the increased problem complexity (there are less feasible solutions, making it harder to find good ones which can be used to dominate other nodes using lower bound arguments) leads to a decrease in efficiency and consequently, a decrease in the number of problems solved to optimality. A similar effect was found to exist for the RCPSP-GPR (De Reyck and Herroelen, 1996c).

7.2.4. The impact of *RF* and *RS*

The effect of *RF* (Fig. 4) is similar to the results reported by Kolisch et al. (1995) and De Reyck and Herroelen (1996a) for the RCPSP and to the findings of De Reyck and Herroelen (1996c) for the RCPSP-GPR. The higher *RF*, the more difficult the corresponding RCPSP(-GPR). An opposite effect can be observed for *RS* (Fig. 5), as was also observed by Kolisch et al. (1995) for the RCPSP and by De Reyck and Herroelen (1996c) for the RCPSP-GPR. The strong effects of *RF* and *RS*, and even more pronounced for *RS* than for *RF*, lead us to believe that the effect of resource-based measures on the computational complexity of the RCPSP-GPR is larger than the effect of network-based measures. A similar observation for the RCPSP has been made by De Reyck and Herroelen (1996a).

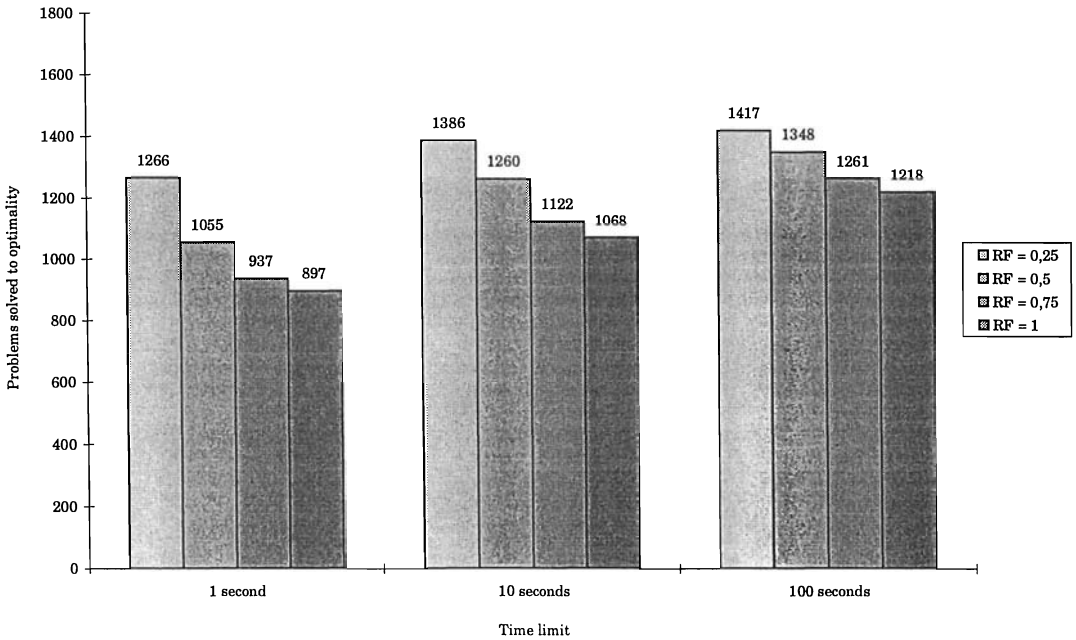

Fig. 4. The effect of *RF* on the number of problems solved to optimality

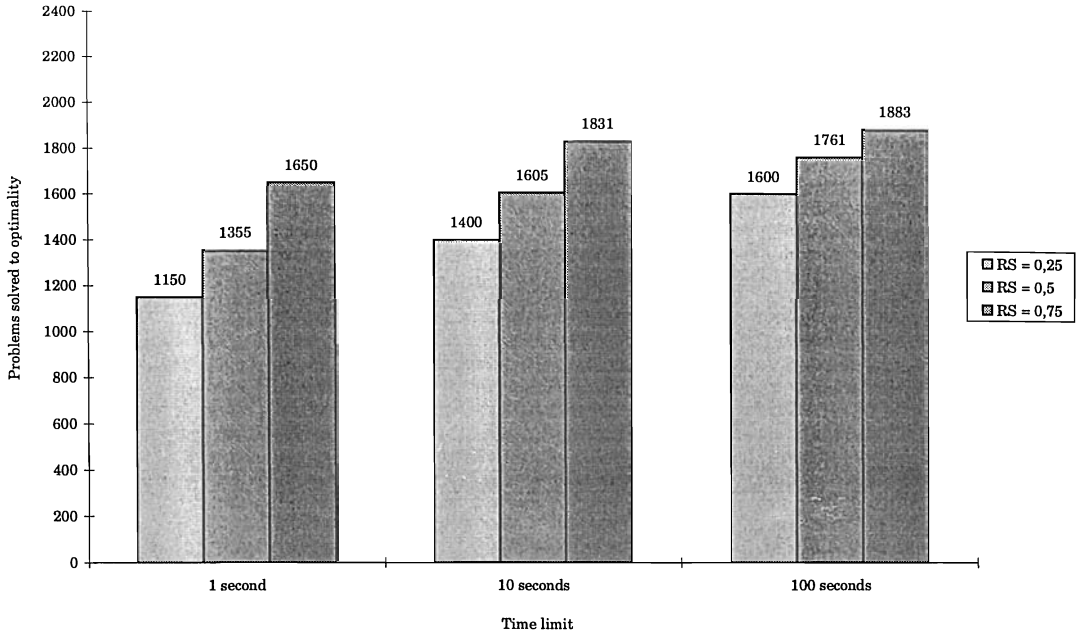

Fig. 5. The effect of *RS* on the number of problems solved to optimality

7.2.5. The impact of the cash flow distribution

In the experiment described above, the cash flows for each of the activities were randomly generated from the interval $[-500,+500]$. This means that, on the average, 50% of the activities will have a negative cash flow associated with it. In practice, the distribution of the cash flows may take very different forms, depending on the contract and payment structure of the project. In some projects, there may be few activities, if any, with a negative cash flow, whereas in other projects, all the activities except for the last activity of the project carry negative cash flows (for a clarifying review of the different types of contracts and payment structures, we refer the reader to Herroelen et al., 1996a). In order to examine the impact of different cash flow distributions on the complexity of the RCPSPDC-GPR, we randomly generated the cash flows of each of the activities from the interval $[0,+500]$, and assigned a negative cash flow to some activities by reversing the sign of the associated cash flow. The number of such activities was varied from 0% to 100% in steps of 10%.

De Reyck and Herroelen (1996d) examined the impact of the percentage of activities with a negative cash flow on the computational effort to solve the unconstrained max-*npv* project scheduling problem with GPRs. It was shown that projects with either few or many activities with negative cash flows constitute the easier instances, whereas problems with a mixture of activities with positive and negative cash flows constitute the most difficult ones. Fig. 6 shows the effect of the percentage of activities with a negative cash flow on the computational effort to solve a representative RCPSPDC-GPR instance (similar results are obtained for other instances). A different curve is shown for different values for the project deadline D (ranging from 21 to 35).

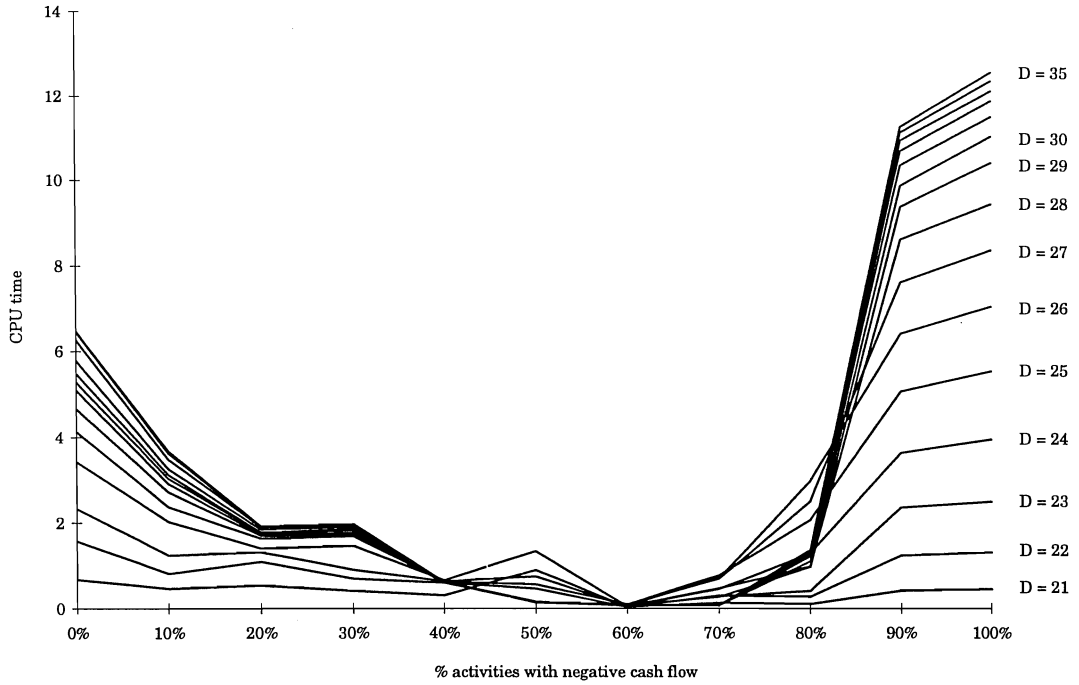

Fig. 6. The effect of the percentage of negative cash flow activities on the problem complexity

Clearly, the effect of the percentage of negative cash flow activities is U-shaped, meaning that, contrary to the unconstrained max-*npv* project scheduling problem, projects with few or many activities with a negative cash flow are the most difficult instances. This result is quite logical, since when activities with negative and positive cash flows are mixed, the optimal schedule may "disconnect" in the sense that some activities are scheduled as close as possible to time zero, whereas others are scheduled as close as possible to the project deadline, such that the problem decomposes into two less complex problems. This only occurs, however, when the project deadline is set high enough such that the optimal schedule can indeed split up into two separate parts. When the deadline is set close to the makespan of the optimal solution for the RCPSP-GPR, no such U-curve will result. Notice that instances with all negative cash flows are more difficult to solve than instances with all positive cash flows. An explanation for this can be found if we remember that, when solving the unconstrained max-*npv* project scheduling problem, we took the *ESS* (early tree) as a starting point, which is less efficient when many activities have a negative terminal cash flow.

7.2.6. The impact of the project deadline

Another conclusion we can draw from Fig. 6 is that when the project deadline increases, the solution space expands and the problem becomes inherently more difficult, which was already observed by Icmeli and Erengüç (1996) for the RCPSPDC. Fig. 7 shows the effect of the project deadline on the computational complexity of the RCPSPDC-GPR instance when it is increased

from 19 (the critical path length) to 37. Several curves are shown, each corresponding to a different percentage of activities with a negative cash flow. Using a deadline of 19 or 20, however, no (resource-)feasible solution can be obtained.

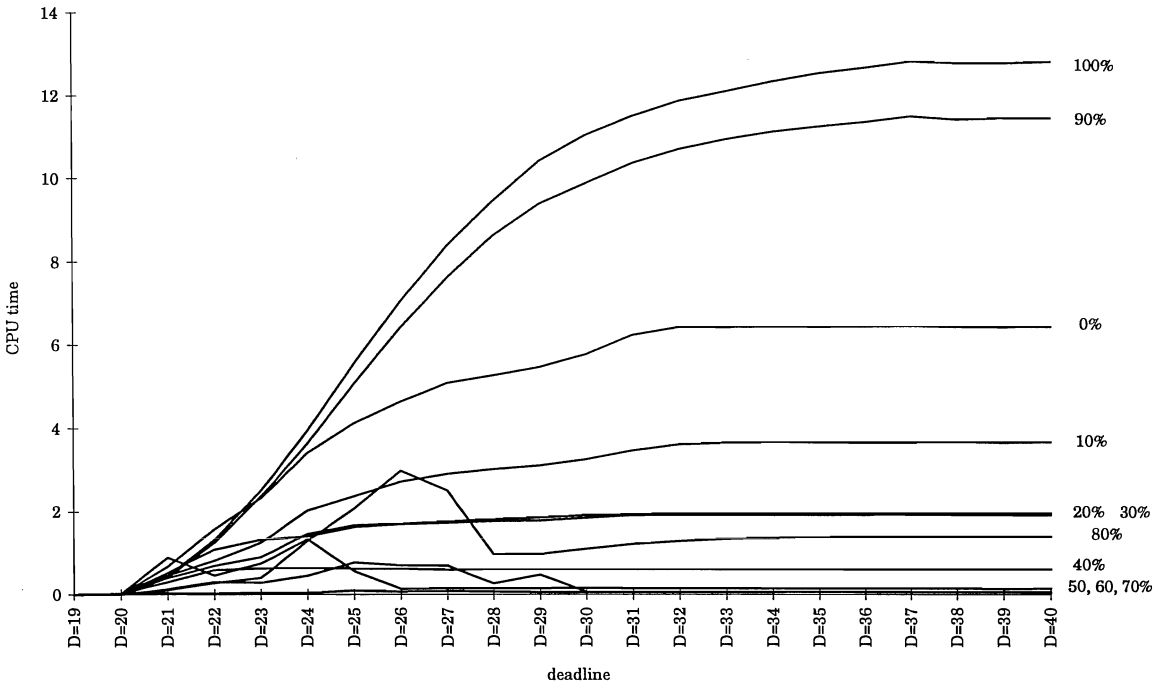

Fig. 7. The effect of the project deadline on the problem complexity

For some settings of the percentage of negative cash flow activities, a continuous increase in CPU-time (Fig. 8) can be observed, which levels out after the deadline reaches a certain value. For other settings of the percentage of negative cash flow activities, the required CPU-time decreases again beyond some critical value of the deadline (Fig. 9). The reason for these different results can be revealed if we look at the percentage of negative cash flows associated with each curve. When this percentage is either low or high (for the example: less or equal than 40% or higher or equal than 90%; representing the most difficult problem instances as can be seen in Fig. 6), this implies that the schedule will probably *not* disconnect into two separate parts, leading to a more complex problem which does not become easier to solve when the deadline is increased. An increased deadline extends the solution space, leading to a higher complexity, until the deadline reaches a value beyond which no expansion of the solution space is observed. The required CPU-time then levels off. When more or less 50% (for the example: 50% to 80%; representing the easiest problem instances) of the activities have a negative terminal cash flow, chances are that the optimal schedule disconnects into two separate parts, thereby reducing the problem complexity since less resource conflicts (and less severe ones) will result, provided that the project deadline is high enough to allow such a disconnected schedule. Therefore, when the deadline

reaches a critical value, the optimal schedule disconnects and the problem complexity decreases. This effect can be observed in Fig. 9.

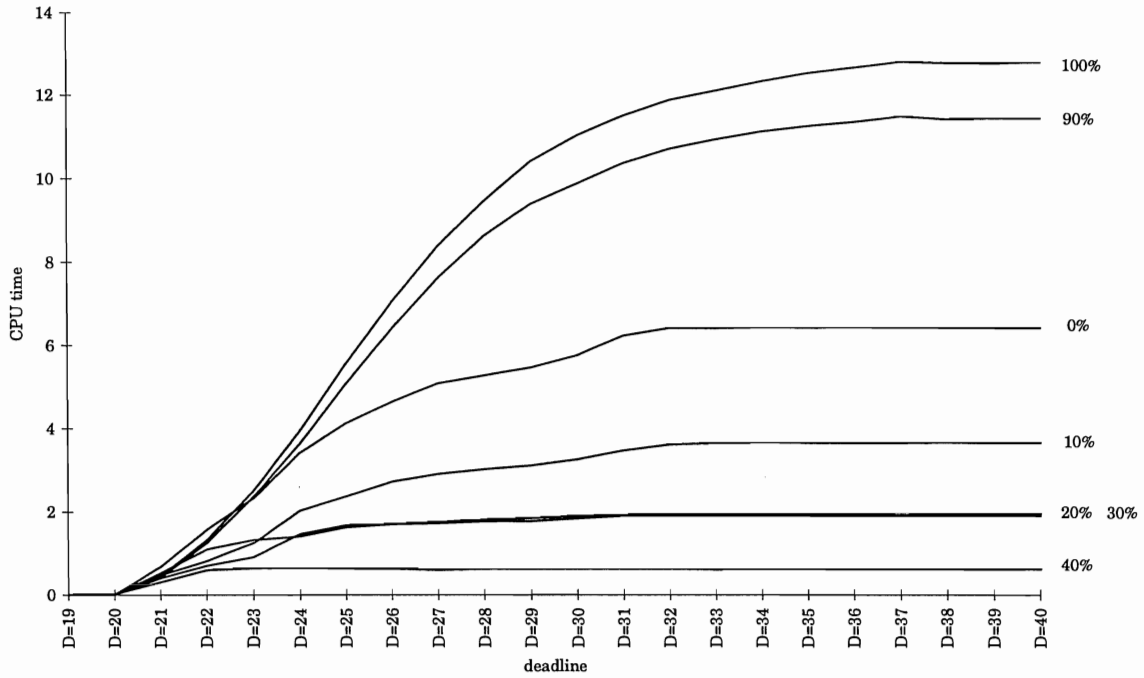

Fig. 8. The effect of the deadline on the complexity (hardest problem instances)

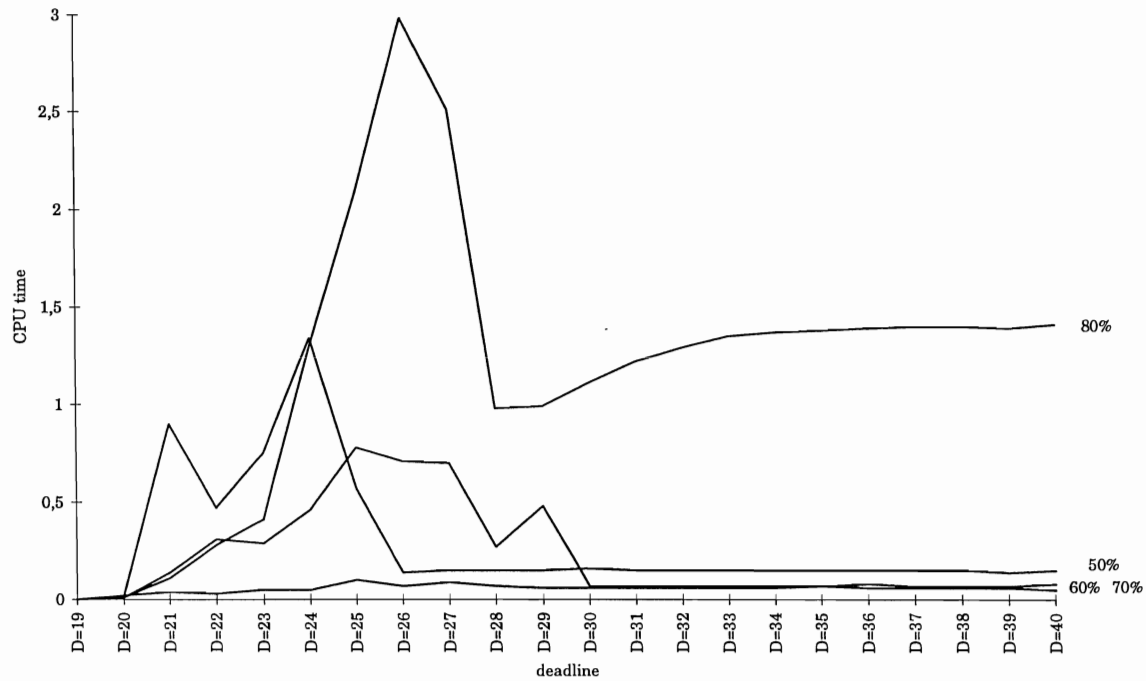

Fig. 9. The effect of the deadline on the complexity (easiest problem instances)

7.5. Conclusions

In this paper we present a branch-and-bound procedure for the RCPSPDC-GPR, the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations. The RCPSPDC-GPR extends the RCPSP to (a) arbitrary minimal and maximal time lags between the starting and completion times of activities and (b) the non-regular objective function which maximizes the net present value of a project with positive and/or negative cash flows associated with the activities. The procedure is a depth-first branch-and-bound algorithm in which the nodes in the search tree represent the original project network extended with extra precedence relations which resolve a number of resource conflicts. Resource conflicts are resolved using the concept of minimal delaying modes. Several dominance rules are used to fathom large portions of the search tree. Each project network in each node of the search tree is evaluated using a new optimal procedure for the (resource-) unconstrained project scheduling problem with generalized precedence relations, which can be used to calculate upper and lower bounds on the project net present value.

Extensive computational results are reported using a problem set consisting of 5760 instances with up to 50 activities, generated using ProGen/max, a new problem generator which can generate several types of generalized resource allocation problems (Schwindt, 1995). Solving all randomly generated problem instances (especially the 50-activity problem instances) to optimality is probably beyond the capabilities of current branch-and-bound procedures. The computational results obtained using a truncated version of the proposed branch-and-bound procedure indicate, however, that the algorithm is capable of solving many of the randomly generated problem instances to optimality. Moreover, the performance of the procedure is not significantly inferior to the procedure for the RCPSP-GPR of De Reyck and Herroelen (1996b), which is only suited for minimizing the project makespan or other *regular* measures of performance. This pleads for the validation of the truncated branch-and-bound procedure as a candidate for solving relatively large instances of the RCPSPDC-GPR against other suboptimal procedures such as priority-rule-based heuristics or local search.

Appendix 1

A procedure for the unconstrained max-npv project scheduling problem with GPRs

STEP 1. DISTANCE MATRIX CALCULATION

Compute the constraint digraph *cd* ($O(|E|)$).

Compute the distance matrix ($O(n^3)$)

If the project is not time-feasible (i.e. $\exists\, i \in V: d_{i,i} > 0$), STOP.

STEP 2. EARLY TREE CALCULATION

Compute the *early tree* as follows ($O(n^2)$): For each activity $i \in V \setminus \{1\}$, search for an activity $j \in V \mid j < i$ for which $d_{1,j} + d_{j,i} = d_{1,i}$. In case several such activities j exist, choose the one with the highest number smaller than i (search in reverse order starting from activity i-1). Link activities j and i in the early tree. Make the early tree the current tree.

STEP 3. CURRENT TREE CALCULATION

Compute a new current tree ($O(n^2)$) by delaying, in reverse order, each activity i with a negative cash flow and no successor in the current tree as much as possible (by increasing $d_{1,i}$), thereby linking it to the activity j preventing a further delay. Remove the link to any predecessor in the current tree. The delay of activity i is computed as $\min_{j \in V \setminus \{i\}} \{d_{1,j} - d_{1,i} - d_{i,j}\}$. If, however, activity j preventing a further delay of activity i is itself a predecessor of activity i in the current tree, activity i can neither be delayed nor linked to activity j. Rather, activities i and j are fixed at their current starting times. Make the so obtained tree the current tree.

If any activity has been delayed in this step, repeat STEP 3.

STEP 4.

$A = \varnothing$.

Do *RECURSION(1)* $\rightarrow$ *PT'*, *DC'* (parameters returned by the recursive function)

Report the optimal schedule $\{d_{1,1}, d_{1,2}, \ldots, d_{1,n}\}$ and net present value *DC'*. STOP.

RECURSION (NEWNODE)

Initialize $PT = \{newnode\}$, $DC = c_{newnode}$, $A = A \cup \{newnode\}$.

Do for each successor activity $i \notin A$ of *newnode* (in the current tree):

RECURSION (i) $\rightarrow$ *PT'*, *DC'*

If $DC' \geq 0$

set $PT = PT \cup PT'$ and $DC = DC + DC'$.

Else

Delete arc (*newnode*, *i*) from the current tree.

Find a new arc with minimal displacement, i.e. arc (*k*,*l*) ($k \in PT, l \notin PT$) with minimal value for $d_{1,l} - d_{1,k} - d_{k,l}$.

Add arc (*k*,*l*) to the current tree.

Update the completion times of the activities in *PT* as follows:

$\forall\, j \in PT: d_{1,j} = d_{1,j} + \min_{\substack{k \in PT \\ l \notin PT}} \{d_{1,l} - d_{1,k} - d_{k,l}\}$.

Go to STEP 4.

Do for each predecessor activity $i \notin A$ of *newnode* (in the current tree):

RECURSION (i) $\rightarrow$ *PT'*, *DC'*

$PT = PT \cup PT'$ and $DC = DC + DC'$.

Return.

Appendix 2

A branch-and-bound procedure for the RCPSPDC-GPR

STEP 1. INITIALISATION

Let lb = -9999 be a lower bound on the project *npv*.

Set the level of the branch-and-bound tree $p = 0$.

Compute the constraint digraph cd ($O(|E|)$).

Compute $d[0]$, the distance matrix at level 0 using the Floyd-Warshall algorithm ($O(n^3)$).

If the project is not time-feasible (i.e. $\exists i \in V: d[0][i][i] > 0$), STOP.

Preprocessing: reduce the solution space by adjusting $d[0]$ ($O(n^2 m)$):

$\forall (i,j) \mid i,j \in V$ and $\exists$ resource type $k: r_{ik} + r_{jk} > a_k$ and

case 1: $-d_j < d[0][i][j] < d_i$, set $l_{ij} = d_i$

case 2: $-d_i < d[0][j][i] < d_j$, set $l_{ji} = d_j$

Recompute $d[0]$ using the Floyd-Warshall algorithm ($O(n^3)$).

Compute an upper bound on the project *npv* using the algorithm for the unconstrained max-*npv* project scheduling problem described in Appendix 1 and go to STEP 3.

STEP 2. TEMPORAL ANALYSIS

Compute $d[p]$, the extended distance matrix at level p as follows ($O(n^2|D_d|)$):

$\forall i,j \in V:\ d[p][i][j] = d[p-1][i][j]$. $\forall i,j \in V$, $l \in D_d:\ d[p][i][j] =$

$\max\{d[p][i][j], d[p-1][i][k] + d_k + d[p-1][l][j]\}$, k being the delaying activity.

If lb > -9999, compute an upper bound ub on the project *npv* using the algorithm for the unconstrained max-*npv* project scheduling problem with GPRs described in section 5 and go to STEP 3.

If $ub \le lb$, erase the delaying mode and go to STEP 6.

STEP 3. RESOURCE ANALYSIS

Determine the *first* period in which a resource conflict occurs, i.e. the first period $]t^*\text{-}1, t^*]$ for which $\sum_{i \in S(t^*)} r_{ik} > a_k$ for some resource type k. $S(t^*)$, the set of activities in progress in period $]t^*\text{-}1, t^*]$, is called the *conflict set*.

If there is no conflict, let $lb = \max\{lb, ub\}$, erase the delaying mode and go to STEP 6.

Store the distance matrix.

STEP 4. DETERMINE MINIMAL DELAYING ALTERNATIVES AND MINIMAL DELAYING MODES

Increase the branch level of the search tree: $p = p + 1$.

Determine the minimal delaying set, i.e. the set of minimal delaying alternatives:

$$D = \left\{ D_d \;\middle|\; D_d \subset S(t^*) \text{ and } \forall \text{ resource type } k: \sum_{i \in S(t^*)} r_{ik} - \sum_{i \in D_d} r_{ik} \le a_k \text{ and } \forall\, D_{d'} \in \mathrm{D}: D_{d'} \not\subset D_d \right\}$$

Extend all minimal delaying alternatives using Theorem 2 and eliminate all non-minimal delaying alternatives. Determine the set of minimal delaying modes:

$$M = \left\{ M_m \;\middle|\; M_m = \{k \prec D_d\},\ k \in S(t^*) \setminus D_d,\ D_d \in D \right\}.$$

Eliminate all delaying modes satisfying Theorem 3.

Arbitrarily select a delaying mode M_m with corresponding delaying alternative D_d.

STEP 5. EVALUATE DELAYING MODES

For all delaying modes M_m

{

If the precedence constraints cannot be added, i.e. $\exists\, l \in D_d$: $k \prec l$ is infeasible, i.e. $d_k > -d[p][l][k]$ (k being the delaying activity), continue with the next delaying mode M_m.

Compute a penalty value as follows: $\Pi = \sum_{l \in D_d} c_l$

If the set of added precedence constraints of a previously examined node saved earlier is a subset of the set of added precedence constraints of the current node, continue with the next delaying mode M_m.

Temporarily store the delaying mode and its penalty value Π.

}

STEP 6. BRANCHING

If no delaying modes are left to branch from at level p, go to STEP 7.

Select the delaying mode M_m with the smallest penalty value Π (arbitrary tie-break).

Go to STEP 2.

STEP 7. BACKTRACKING

Decrease the branch level of the search tree: $p = p - 1$.

If $p \leq 0$, STOP with the optimal solution with an *npv* equal to *lb*

(if *lb* = -9999, then there exists no feasible solution).

Delete from the stack the information which has been previously saved on level p+1 for dominance testing.

Save the necessary information for node dominance testing on the stack, i.e. the list of added precedence constraints of the node reached upon backtracking.

Erase the distance matrix and the lower bound of this node and go to STEP 6.

References

Baroum, S. and Patterson, J.H., 1996, "An exact solution procedure for maximizing the net present value of cash flows in a network", *Fifth International Workshop on Project Management and Scheduling*, 11 - 13 april, Poznan, 31-34.

Bartusch, M., Möhring, R.H. and Radermacher, F.J., 1988, "Scheduling project networks with resource constraints and time windows", *Annals of Operations Research*, 16, 201-240.

Brinkmann, K. and Neumann, K., 1994, "Heuristic procedures for resource-constrained project scheduling with minimal and maximal time lags: the minimum project-duration and resource-levelling problem", Technical Report WIOR-443, Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe.

Crandall, K.C., 1973, "Project planning with precedence lead / lag factors", *Project Management Quarterly*, 4, 18-27.

Dar-El, E. M., 1973, "MALB-A heuristic technique for balancing large single-model assembly lines", *AIIE Transactions*, 5, 343-356.

De Reyck, B., 1995a, "Project scheduling under generalized precedence relations - A review : Part 1", Research Report 9517, Department of Applied Economics, Katholieke Universiteit Leuven.

De Reyck, B., 1995b, "Project scheduling under generalized precedence relations - A review : Part 2", Research Report 9518, Department of Applied Economics, Katholieke Universiteit Leuven.

De Reyck, B., 1995c, "On the use of the restrictiveness as a measure of complexity for resource-constrained project scheduling", Research Report 9535, Department of Applied Economics, Katholieke Universiteit Leuven.

De Reyck, B. and Herroelen, W., 1996a, "On the use of the complexity index as a measure of complexity in activity networks", *European Journal of Operational Research*, 91, 347-366.

De Reyck, B. and Herroelen, W., 1996b, "A branch-and-bound algorithm for the resource-constrained project scheduling problem with generalized precedence relations", Research Report 9613, Department of Applied Economics, Katholieke Universiteit Leuven.

De Reyck, B. and Herroelen, W., 1996c, "Computational experience with a branch-and-bound algorithm for the resource-constrained project scheduling problem with generalized precedence relations", Research Report 9628, Department of Applied Economics, Katholieke Universiteit Leuven.

De Reyck, B. and Herroelen, W., 1996d, "An optimal procedure for the unconstrained max-*npv* project scheduling problem with generalized precedence relations", Research Report 9642, Department of Applied Economics, Katholieke Universiteit Leuven.

Demeulemeester, E. and Herroelen, W., 1992, "A branch-and-bound procedure for the multiple resource-constrained project scheduling problem", *Management Science*, 38, 1803-1818.

Demeulemeester, E. and Herroelen, W., 1996, "New benchmark results for the resource-constrained project scheduling problem", *Management Science*, to appear.

Doersch, R.H. and Patterson, J.H., 1977, "Scheduling a Project to Maximize its Present Value: A Zero-One Programming Approach", *Management Science*, 23, 882-889

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

Elmaghraby, S.E. and Herroelen, W., 1990, "The scheduling of activities to maximize the net present value of projects", *European Journal of Operational Research,* 49, 35-49.

Elmaghraby, S.E. and Kamburowski, J., 1992, "The analysis of activity networks under generalized precedence relations", *Management Science*, 38, 1245-1263.

Franck, B. and Neumann, K., 1996, "Priority-rule methods for the resource-constrained project scheduling problem with minimal and maximal time lags - an emprical analysis", *Fifth International Workshop on Project Management and Scheduling*, 11 - 13 april, Poznan, 88-91.

Grinold, R.C., 1972, "The payment scheduling problem", *Naval Research Logistics Quarterly*, 19, 123-136.

Herroelen, W. and Gallens, E., 1993, "Computational experience with an optimal procedure for the scheduling of activities to maximize the net present value of projects", *European Journal of Operational Research*, 65, 274-277.

Herroelen, W., Demeulemeester, E. and Van Dommelen, P., 1996a, "Project network models with discounted cash flows: A guided tour through recent developments", *European Journal of Operational Research*, to appear.

Herroelen, W., Demeulemeester, E. and Van Dommelen, P., 1996b, "An optimal recursive search procedure for the deterministic unconstrained max-*npv* project scheduling problem", Research Report, Department of Applied Economics, Katholieke Universiteit Leuven.

Icmeli, O and Erengüç, S.S., 1994, "A tabu search procedure for the resource-constrained project scheduling problem with discounted cash flows", Computers and Operations Research, 8, 841-853.

Icmeli, O and Erengüç, S.S., 1996, "A branch-and-bound procedure for the resource-constrained project scheduling problem with discounted cash flows", *Management Science*, 42(10).

Kao, E.P.C. and Queyranne, M., 1982, "On dynamic programming methods for assembly line balancing", *Operations Research*, 30, 375-390.

Kazaz, B. and Sepil, C., 1994, "Project scheduling with discounted cash flows and progress payments", *Journal of the Operational Research Society*, 47, 1262-1272.

Kelley, J.E., Jr. and Walker, M.R., 1959, "Critical path planning and scheduling", *Eastern Joint Computing Conference*, 16, 160-172.

Kerbosh, J.A.G.M. and Schell, H.J., 1975, "Network planning by the Extended METRA Potential Method", Report KS-1.1, University of Technology Eindhoven, Department of Industrial Engineering.

Kolisch, R., Sprecher, A. and Drexl, A., 1995, "Characterization and generation of a general class of resource-constrained project scheduling problems", Management Science, 41(10), 1693-1703.

Lawler, E.L., 1976, *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart and Winston, New York.

Malcolm, D.G., Roseboom, J.H., Clark, C.E. and Fazar, W., 1959, "Applications of a technique for R&D program evaluation (PERT)", *Operations Research*, 7(5), 646-669.

Mastor, A. A., 1970, "An experimental and comparative evaluation of production line balancing techniques", *Management Science*, 16, 728-746.

Moder, J.J., Phillips, C.R. and Davis, E.W., 1983, *Project management with CPM, PERT and precedence diagramming*, Van Nostrand Reinhold Company, Third Edition.

Neumann, K. and Schwindt, C., 1995, "Projects with minimal and maximal time lags: construction of activity-on-node networks and applications", Technical Report WIOR-447, Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe.

Neumann, K. and Zhan, J., 1995, "Heuristics for the minimum project-duration problem with minimal and maximal time lags under fixed resource constraints", *Journal of Intelligent Manufacturing*, 6, 145-154.

Özdamar, L., Ulusoy, G. and Bayyigit, M., 1994, "A heuristic treatment of tardiness and net present value criteria in resource-constrained project scheduling", Working Paper, Department of Industrial Engineering, Marmara University.

Padman, R., Smith-Daniels, D.E. and Smith-Daniels, V.L., 1990, "Heuristic scheduling of resource-constrained projects with cash flows: an optimization-based approach", Working Paper 90-6, Carnegie-Mellon University.

Padman, R. and Smith-Daniels, D.E., 1993a, "Maximizing the net present value of capital-constrained projects: an optimization-guided approach", Working Paper 93-56, Carnegie-Mellon University.

Padman, R. and Smith-Daniels, D.E., 1993b, "Early-Tardy Cost Trade-Offs in Resource Constrained Projects with Cash Flows: An Optimization-Guided Heuristic Approach", *European Journal of Operational Research*, 64, 295-311.

Pascoe, T.L., 1966, "Allocation of resources - CPM", *Revue Française de Recherche Opérationelle*, 38, 31-38.

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

Patterson, J.H., Slowinski, R., Talbot, F.B. and Weglarz, J., 1990b, "An algorithm for a general class of precedence and resource-constrained scheduling problems", in: Slowinski, R. and Weglarz J. (Ed.), *Advances in Project Scheduling*, Elsevier, Amsterdam, 3-28.

Roy, B., 1962, "Graphes et ordonnancement", *Revue Française de Recherche Opérationelle*, 323-333.

Russell, A.H., 1970, "Cash flows in networks", *Management Science*, 16, 357-373.

Russell, R.A., 1986, "A Comparison of Heuristics for Scheduling Projects with Cash Flows and Resource Restrictions", *Management Science*, 32, 291-300.

Schwindt, C., 1995, "ProGen/max: a new problem generator for different resource-constrained project scheduling problems with minimal and maximal time lags", Technical Report WIOR-449, Institut für Wirtschaftstheorie und Operations Research, Universität Karlsruhe.

Schwindt, C. and Neumann, K., 1996, "A new branch-and-bound-based heuristic for resource-constrained project scheduling with minimal and maximal time lags", *Fifth International Workshop on Project Management and Scheduling*, 11 - 13 April, Poznan, 212-215.

Sepil, C. and Ortaç, N., 1995, "Performance of the heuristic procedures for constrained projects with progress payments", Working Paper, Middle East Technical University.

Smith-Daniels, D.E. and Aquilano, N.J., 1987, "Using a Late-Start Resource-Constrained Project Schedule to Improve Project Net Present Value", *Decision Sciences*, 18, 617-630.

Smith-Daniels, D.E. and Smith-Daniels, V.L., 1987, "Maximizing the Net Present Value of a Project Subject to Materials and Capital Constraints", *Journal of Operations Management*, 7, 33-45.

Thesen, A., 1977, "Measures of the restrictiveness of project networks", *Networks*, 7, 193- 208.

Ulusoy, G. and Özdamar, L., 1995, "A Heuristic Scheduling Algorithm for Improving the Duration and Net Present Value of a Project", *International Journal of Operations and Production Management*, 15, 89-98.

Wiest, J.D., 1981, "Precedence diagramming methods: some unusual characteristics and their implications for project managers", *Journal of Operations Management*, 1, 121-130.

Yang, K.K., Talbot, F.B. and Patterson, J.H., 1992, "Scheduling a Project to Maximize Its Net Present Value: An Integer Programming Approach", *European Journal of Operational Research*, 64, 188-198.

Yang, K.K., Tay, L.C. and Sum, C.C., 1995, "A Comparison of Stochastic Scheduling Rules for Maximizing Project Net Present Value", *European Journal of Operational Research*, 85, 327-339.

Zhan, J., 1994, "Heuristics for scheduling resource-constrained projects in MPM networks", *European Journal of Operational Research*, 76, 192-205.

Zhu, D. and Padman, R., 1993, "Heuristic selection in resource-constrained project scheduling: experiments with neural networks", Working Paper 93-43, Carnegie-Mellon University.