DEPARTEMENT TOEGEPASTE ECONOMISCHE WETENSCHAPPEN

ONDERZOEKSRAPPORT NR 9642

An Optimal Procedure for the Unconstrained Max-NPV Project Scheduling Problem with Generalized Precedence Relations

by

Bert De Reyck

Willy Herroelen

Katholieke Universiteit Leuven

Naamsestraat 69, B-3000 Leuven

ONDERZOEKSRAPPORT NR 9642

An Optimal Procedure for the Unconstrained Max-NPV Project Scheduling Problem with Generalized Precedence Relations

by

Bert De Reyck

Willy Herroelen

D/1996/2376/42

AN OPTIMAL PROCEDURE FOR THE UNCONSTRAINED MAX-NPV PROJECT SCHEDULING PROBLEM WITH GENERALIZED PRECEDENCE RELATIONS

Bert DE REYCK

Willy HERROELEN

August 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 UNCONSTRAINED MAX-NPV PROJECT SCHEDULING PROBLEM WITH GENERALIZED PRECEDENCE RELATIONS

Bert De Reyck • Willy Herroelen

Department of Applied Economics, Katholieke Universiteit Leuven

ABSTRACT

The unconstrained max-npv project scheduling problem involves the scheduling of the activities of a project in order to maximize its net present value. Assume a project represented in activity-on-node (AoN) notation, in which the activities have a known duration and are subject to technological precedence constraints. Throughout each activity, a series of cash outflows and receipts may occur, which allows for the computation of a terminal cash flow value (positive or negative) upon its completion. The project is to be scheduled against a fixed deadline in the absence of resource constraints. Several procedures have been presented in the literature to cope with this problem. In this paper, we describe how one of the most efficient optimal procedures can be adapted to cope with generalized precedence relations, which introduce arbitrary minimal and maximal time lags between the start and completion of activities. The procedure has been programmed in Microsoft® Visual C++ 2.0 under Windows NT for use on a personal computer. Extensive computational results are reported.

1. Introduction

Recently, a number of publications have dealt with various types of project scheduling problems with the objective of maximizing the net present value (*npv*) of the project, in which cash flows are associated with the activities of the project, and the objective is to schedule the activities in such a way that the net present value 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 take place in the form of progress payments for completed work. We assume that these cash flows can be associated with the completion of the project activities.

The research presented in the literature can be classified in different ways. For a recent review, we refer the reader to Herroelen et al. (1996a). We distinguish between, on the one hand, procedures for the *unconstrained max-npv project scheduling problem*, i.e. when no constraints on the resource usage are imposed such that the activities are only subject to precedence constraints, and, on the other hand, 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), Sepil and Kazaz (1994) and Herroelen et al. (1996b), the most efficient of which seems to be the procedure of Herroelen et al. (1996b). Optimal algorithms for the resource-constrained case have been presented by Doersch and Patterson (1977), Smith-Daniels and Smith-Daniels (1987), Patterson et al. (1989,1990), Yang et al. (1992), Icmeli and Erengüç (1995) 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 (1993a,1993b), 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 a model and an optimal solution procedure for the unconstrained max-npv project scheduling problem with generalized precedence relations (GPRs), which allows for arbitrary minimal and maximal time lags between the start and completion of activities. To the best of our knowledge, no procedure has been presented yet for the unconstrained max-npv problem with GPRs, neither for the minimal time lag case (precedence diagramming), nor for the minimal/maximal time lag case (generalized precedence relations). We describe how the procedure of Herroelen et al. (1996b) can be adapted to cope with generalized precedence relations.

The remainder of this paper is organized as follows. In section 2 we introduce the basic problem type under study, namely the unconstrained max-npv problem with GPRs. Section 3 continues with a brief description of time analysis in project networks with generalized precedence relations and defines some basic concepts which will be used in the description of the solution procedure. The solution procedure itself is described in section 4. In section 5, an example will be given to illustrate the algorithm. Section 6 reports extensive computational experience using a random problem generator which can generate project networks with generalized precedence relations (Schwindt, 1995). Section 7 is reserved for our conclusions and suggestions for future research.

2. The unconstrained max-npv problem with generalized precedence relations

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 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 given by $d_i (1 \le i \le n)$, its starting time by $s_i (1 \le i \le n)$ and its finishing time by $f_i (1 \le i \le n)$. The problem is unconstrained in the sense that no constraints are imposed on the use of resources. Throughout each activity, a series of cash outflows and receipts may occur, which allows for the computation of a terminal cash flow value (which may be positive or negative) upon its completion as follows:

$$c_i = \sum_{t=1}^{d_i} f_{it}\, e^{\alpha(d_i - t)}$$

where c_i represents the terminal value of all the cash flows occurring during the execution of activity i, f_{it} denotes the cash flow occurring during the t^{th} $(1 \le t \le d_i)$ period activity i is in progress and α is the discount rate.

The minimal and maximal time lags between two activities i and j are of the form:

$$s_i + SS_{ij}^{\min} \le s_j \le s_i + SS_{ij}^{\max}$$

$$s_i + SF_{ij}^{\min} \le f_j \le s_i + SF_{ij}^{\max}$$

$$f_i + FS_{ij}^{\min} \le s_j \le f_i + FS_{ij}^{\max}$$

$$f_i + FF_{ij}^{\min} \le f_j \le f_i + FF_{ij}^{\max}$$

The different types of GPRs 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):

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

If there is more than one time lag l_{ij} between two activities i and j, only the maximal time lag is retained. The interval $[s_i + l_{ij}\ ,\ s_i - l_{ji}]$ is called the *time window* of s_j relative to s_i (Bartusch et al., 1988). Applying these transformation rules to an activity network with GPRs results in a so-called *constraint digraph*, which is short for *digraph of temporal constraints* (Bartusch et al., 1988).

Then, the unconstrained max-npv project scheduling problem with generalized precedence relations can be conceptually formulated as follows:

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

Subject to

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

$$s_1 = 0 \qquad [3]$$

$$s_n - D \le 0 \qquad [4]$$

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

where D denotes the project deadline, which is enforced by a maximal start-start time lag of D between the dummy start activity 1 and the dummy end activity n.

We will show how the procedure of Herroelen et al. (1996b) can be adapted to cope with GPRs, allowing it to be incorporated in a procedure for the *resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations* (RCPSPDC-GPR) for the calculation of upper bounds on the net present value of a project.

3. Temporal analysis of project networks with GPRs

Because activity networks with GPRs contain cycles, additional concepts are needed (Bartusch et al., 1988). A path $<i_s, i_k, i_l, ..., i_t>$ is called a *cycle* if $s = t$. With 'path' we mean a *directed* path, and with 'cycle' we mean a *directed* cycle. The *length* of a path (cycle) is defined as the sum of all the lags associated with the arcs belonging to that path (cycle). Activity durations do not have to be included in the calculation of a path length, since all time lags l_{ij} in a constraint digraph are of the SS-type. 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 *predecessor* of j and j a *successor* of i.

The goal of project scheduling problems is to obtain a schedule S, which is a vector of starting times $\{s_1, s_2, ..., s_n\}$ for all activities. Schedules can be subject to temporal constraints and resource constraints. In this paper, we focus on the temporal constraints. A schedule is called *time-feasible*, if all the starting times satisfy all GPRs. In other words, a time-feasible schedule with starting times $\{s_1, s_2, ..., s_n\}$ satisfies the conditions that:

$$\begin{cases} s_i \geq 0 & \forall\, i \in V \quad [6] \\ s_i + l_{ij} \leq s_j & \forall\, (i,j) \in E \quad [7] \end{cases}$$

where Eqs. 6 ensure that no activity starts before the current time (time zero), and Eqs. 7 denote the precedence constraints in standardized form. The minimum starting times $\{s_1, s_2, ..., s_n\}$ satisfying both Eqs. 6 and 7 form the *early start schedule* $ESS = \{es_1, es_2, ..., es_n\}$ associated with the temporal constraints.

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). Cycles of positive length would unable us to calculate starting times for the activities which satisfy conditions [6] and [7]. Therefore, if we calculate the distance matrix $D = [d_{i,j}]$, where $d_{i,j}$ 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). If we start with the matrix $D^{(1)} = \left[d_{i,j}^{(1)}\right]$ $(i,j = 1, 2, ..., n)$ with

$$d_{i,j}^{(1)} = \begin{cases} 0 & \text{if } i = j \\ l_{ij} & \forall\,(i,j) \in E \\ -\infty & \text{otherwise} \end{cases}$$

we can compute the matrix $D = D^{(n+1)}$ according to the updating formula $d_{i,j}^{(v)} = \max\left\{d_{i,j}^{(v-1)}, d_{i,l}^{(v-1)} + d_{l,j}^{(v-1)}\right\}$ $(i, j, l = 1, 2, ..., n)$. If $d_{i,i} = 0$ for all $i = 1, 2, ..., n$ (the numbers in the diagonal of D), there exists a time-feasible schedule. The *ESS* is given by the numbers in the upper row of D: $ESS = \{d_{1,1}, d_{1,2}, ..., d_{1,n}\}$.

The computation of D takes $O(|V|^3)$ time (Bartusch et al., 1988). The *ESS* can be calculated more efficiently by using the Modified Label Correcting Algorithm (Ahuja et al., 1989), which is of time complexity $O(|V|\,|E|)$ and which also allows for the identification of positive cycles.

4. The optimal solution procedure

4.1. Description

The procedure of Herroelen et al. (1996b) can be extended to cope with GPRs in the following way: We start in STEP 1 by computing the constraint digraph using the transformation rules discussed in section 2 (time complexity $O\left[n^2\right]$). Then, the distance matrix is computed using the Floyd-Warshall algorithm (time complexity $O\left[n^3\right]$). If the project is not time-feasible, i.e. if there is an activity i for which $d_{i,i} > 0$, 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: For every activity i, a predecessor j is determined for which $d_{1,j} + d_{j,i} = d_{1,i}$, upon which activities j and i are linked. For every activity i, there always exists a predecessor activity j satisfying $d_{1,j} + d_{j,i} = d_{1,i}$, 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 $d_{1,j}+d_{j,i}=d_{1,i}$ holds (using a reverse search scheme).

The *current tree* is calculated in STEP 3 of the algorithm 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, the procedure will enter a recursive search, in which partial trees PT (with a negative net present value) 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 $d_{1,l}-d_{1,k}-d_{k,l}$. 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. Then, we update the completion times of the activities in the partial tree as follows: $\forall j \in PT$: $d_{1,j}=d_{1,j}+\min_{\substack{k\in PT \\ l\notin PT}}\{d_{1,l}-d_{1,k}-d_{k,l}\}$. 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.

Notice that, contrary to the procedure of Herroelen et al. (1996b), it is not possible that the current tree disconnects into two parts, one part being shifted forward till it hits the deadline.

When the deadline is enforced using a generalized precedence relation (a maximal start-start time lag between the dummy start and the dummy end activity), the current tree will never disconnect because this deadline-GPR will always keep both parts together. Therefore, the extra step that is needed in the procedure of Herroelen et al. (1996b) to cope with such disconnected current trees is not needed in our procedure.

4.2. The algorithm

STEP 1. DISTANCE MATRIX CALCULATION

Compute the constraint digraph *cd*.

Compute the distance matrix.

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: For each activity $i \in V \setminus \{1\}$, search for an activity $j \in V\ (j < i)$ 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 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 calculated 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.

5. Example

Consider the project with cash flows and GPRs given in Fig. 1 (adapted from Elmaghraby and Kamburowski, 1992). The nodes represent the project activities. The number above each node denotes the activity duration, the number below denotes the associated cash flow. The labels associated with the arcs denote the GPRs. The critical paths are indicated in bold. Notice that the maximal start-start time lag between dummy activities 1 and 10 represents the project deadline, which is assumed to be 25. The discount rate α equals 0.02. We will compute the optimal solution by going through the steps of the algorithm.

STEP 1. DISTANCE MATRIX CALCULATION

Compute the constraint digraph *cd* (see Fig. 2).

Compute the distance matrix (see Fig. 3).

The project is time-feasible ($\forall i \in V: d_{i,i} = 0$). As can be seen from Fig. 3, the critical path length, or, equivalently, the length of the earliest start schedule equals 16 time units:

$ESS = \{d_{1,1}, d_{1,2}, \ldots, d_{1,n}\} = \{0,0,2,2,8,3,9,10,11,16\}$. The net present value of the ESS equals 2.10.

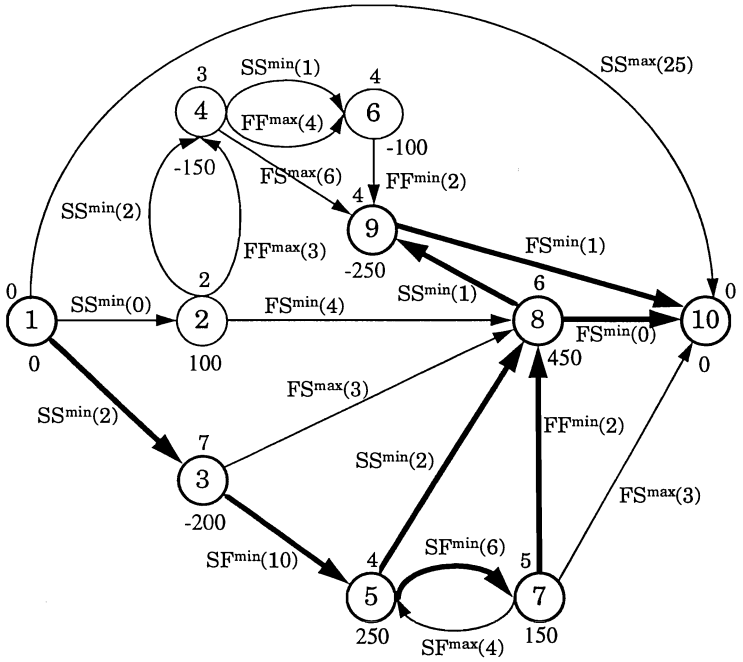

Fig. 1. A project network with GPRs and cash flows

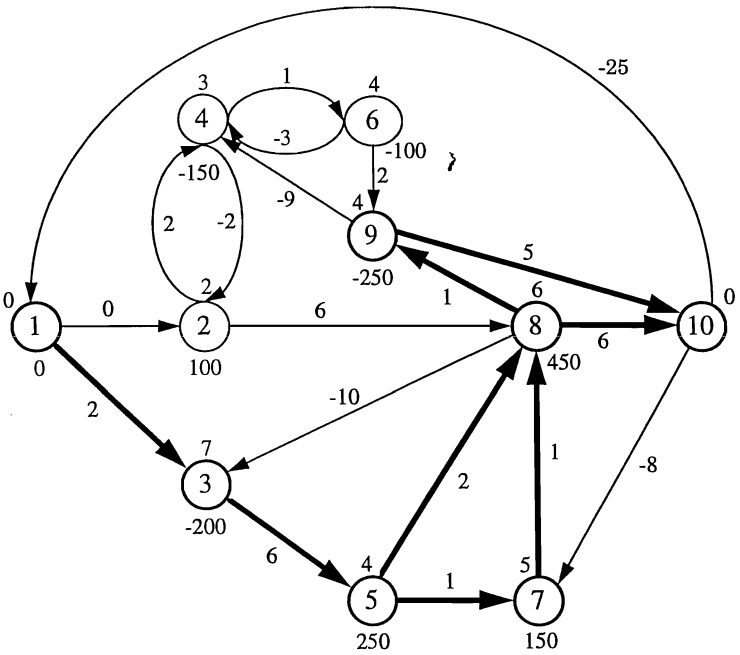

Fig. 2. The example of Fig. 1 in standardized form

$$D = \begin{bmatrix} 0 & 0 & 2 & 2 & 8 & 3 & 9 & 10 & 11 & 16 \\ -\infty & 0 & -4 & 2 & 2 & 3 & 4 & 6 & 7 & 12 \\ -\infty & -2 & 0 & 0 & 6 & 1 & 7 & 8 & 9 & 14 \\ -\infty & -2 & -6 & 0 & 0 & 1 & 2 & 4 & 5 & 10 \\ -\infty & -8 & -8 & -6 & 0 & -5 & 1 & 2 & 3 & 8 \\ -\infty & -5 & -9 & -3 & -3 & 0 & -1 & 1 & 2 & 7 \\ -\infty & -9 & -9 & -7 & -3 & -6 & 0 & 1 & 2 & 7 \\ -\infty & -10 & -10 & -8 & -4 & -7 & -2 & 0 & 1 & 6 \\ -\infty & -11 & -12 & -9 & -6 & -8 & -3 & -2 & 0 & 5 \\ -\infty & -17 & -17 & -15 & -11 & -14 & -8 & -7 & -6 & 0 \end{bmatrix}$$

Fig. 3. The distance matrix of the example in Fig. 1

STEP 2. EARLY TREE CALCULATION

Compute the early tree (see Fig. 4). As stated before, the early tree corresponds to a feasible schedule with a net present value of 2.10. Make this tree the current tree.

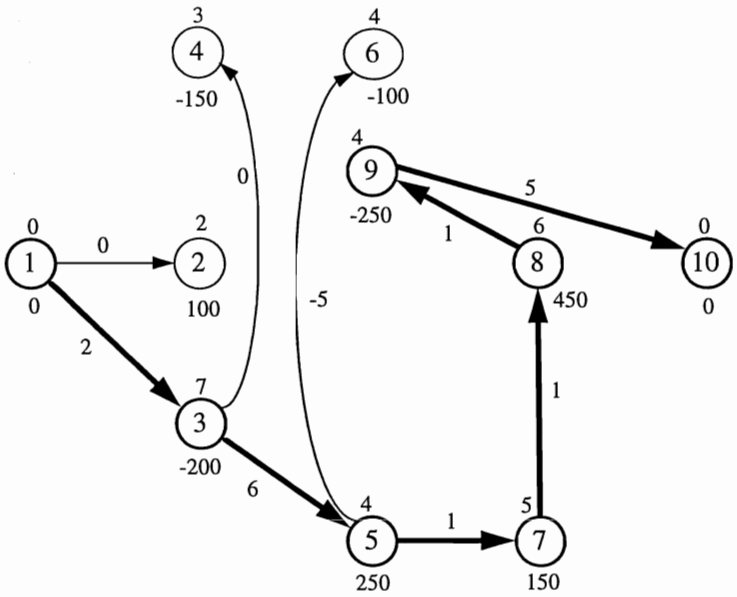

Fig. 4. The early tree with an *npv* of 2.10

STEP 3. CURRENT TREE CALCULATION

Compute a new current tree. Activity 9 has a negative cash flow but cannot be delayed because it is already linked to successor activity 10. Activity 6 with a negative cash flow and no successor in the current tree can be delayed. The delay is calculated as:

$\min_{j \in V \setminus \{6\}} \{d_{1,j} - d_{1,i} - d_{i,j}\} = d_{1,2} - d_{1,6} - d_{6,2} = 0 - 3 - (-5) = 2$. Therefore, delay activity 6 till time period 5, i.e. set $d_{1,6} = 5$, remove the link between activities 5 and 6 and link activity 6 to activity 2. Activity 4 with a negative cash flow and no successor in the current tree can be delayed. The delay is calculated as: $\min_{j \in V \setminus \{4\}} \{d_{1,j} - d_{1,i} - d_{i,j}\} = d_{1,2} - d_{1,4} - d_{4,2} =$ $0 - 2 - (-2) = 0$. The delay is equal to zero, which implies that a delay of activity 4 is impossible due to the precedence relation with activity 2. However, we do remove the link between activities 3 and 4 and link activity 4 to activity 2. Activity 3 cannot be delayed since it is already linked to successor activity 5 in the current tree. Notice that, contrary to the case when only zero-lag finish start precedence constraints are present, it is possible that loose nodes are created during this step. For instance, if the cash flow associated with activity 2 would have been negative, activity 2 would have been delayed till time period 4 ($d_{1,2} = 4$), thereby linking activity 2 to activity 8 and removing the link between activities 1 and 2, activities 4 and 2 and activities 6 and 2. Consequently, activities 4 and 6 would not be connected anymore to any other activity. This, however, would be resolved when Step 3 is repeated, in which the loose nodes will be relinked with their appropriate successor.

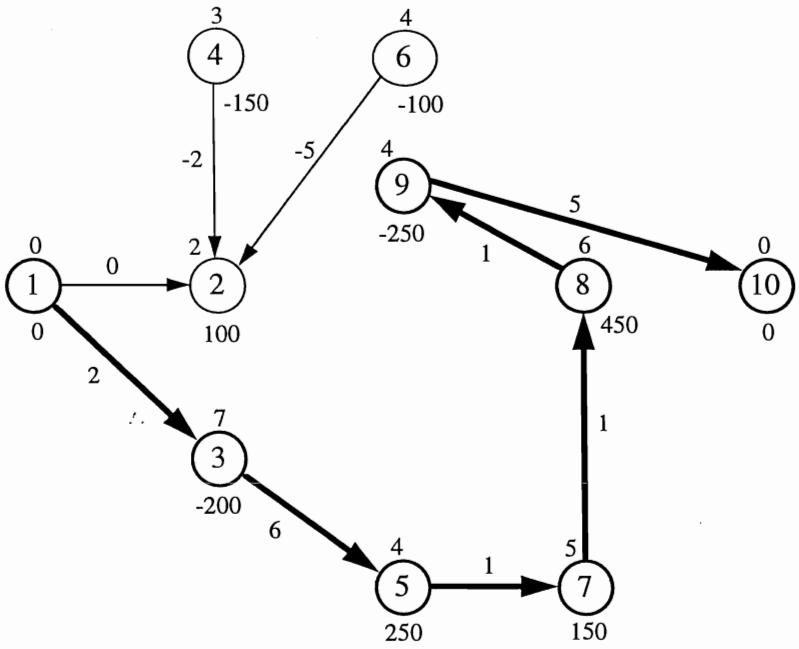

Fig. 5. The current tree with an *npv* of 161.36

STEP 3. CURRENT TREE CALCULATION

No activities can be delayed any further. Make the so obtained tree the current tree (Fig. 5)

with starting times $\{0,0,2,2,8,5,9,10,11,16\}$ and net present value 161.36. Notice that, when the cash flow associated with activity 2 would have been negative, the loose nodes 4 and 6 would have been connected to their appropriate successor in the current tree, namely activity 2.

STEP 4.

$A = \varnothing$

RECURSION (1)

$PT = \{1\},\ DC = 0\ , A = \{1\}.$

RECURSION (2)

$PT = \{2\},\ DC = 100\,e^{-0.02(0+2)} = 96.08\ , A = \{1,2\}.$

RECURSION (4)

$PT = \{4\},\ DC = -150\,e^{-0.02(2+3)} = -135.73\ , A = \{1,2,4\}.$

$PT = \{2,4\},\ DC = 96.08 - 135.73 = -39.65\ .$

RECURSION (6)

$PT = \{6\},\ DC = -100e^{-0.02(5+4)} = -83.53\ , A = \{1,2,4,6\}.$

$PT = \{2,4,6\},\ DC = -39.65 - 83.53 = -123.17\ .$

$DC = -123.17 < 0$. Delete arc (1,2). Add arc (2,8) with displacement $d_{1,8} - d_{1,2} - d_{2,8}$ $= 10 - 0 - 6 = 4$ to the current tree: $d_{1,2} = 4\ ; d_{1,4} = 6\ ; d_{1,6} = 9$. The *npv* now equals 170.83. The resulting current tree is displayed in Fig. 6.

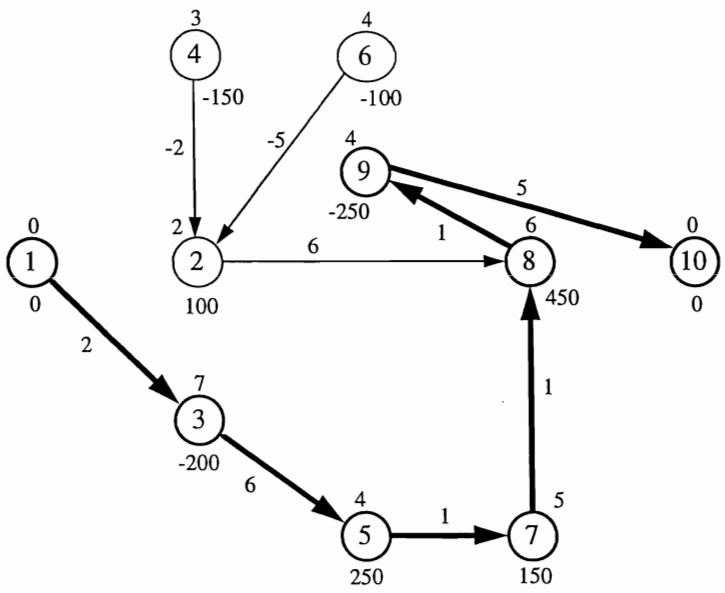

Fig. 6. The intermediate current tree with an *npv* of 170.83

RECURSION (1)

$PT = \{1\}$, $DC = 0$, $A = \{1\}$.

RECURSION (3)

$PT = \{3\}$, $DC = -200\,e^{-0.02(2+7)} = -167.05$, $A = \{1,3\}$.

RECURSION (5)

$PT = \{5\}$, $DC = 250\,e^{-0.02(8+4)} = 196.66$, $A = \{1,3,5\}$.

RECURSION (7)

$PT = \{7\}$, $DC = 150\,e^{-0.02(9+5)} = 113.37$, $A = \{1,3,5,7\}$.

RECURSION (8)

$PT = \{8\}$, $DC = 450\,e^{-0.02(10+6)} = 326.77$, $A = \{1,3,5,7,8\}$.

RECURSION (9)

$PT = \{9\}$, $DC = -250\,e^{-0.02(11+4)} = -185.20$, $A = \{1,3,5,7,8,9\}$.

RECURSION (10)

$PT = \{10\}$, $DC = 0$, $A = \{1,3,5,7,8,9,10\}$.

$PT = \{9,10\}$, $DC = -185.20 + 0 = -185.20$.

$DC = -185.20 < 0$. Delete arc (8,9). Add arc (9,7) with displacement $d_{1,7} - d_{1,9} - d_{9,7} = 9 - 11 - (-3) = 1$ to the current tree: $d_{1,9} = 12$; $d_{1,10} = 17$. The *npv* of this schedule is 174.50. The resulting current tree is given in Fig. 7.

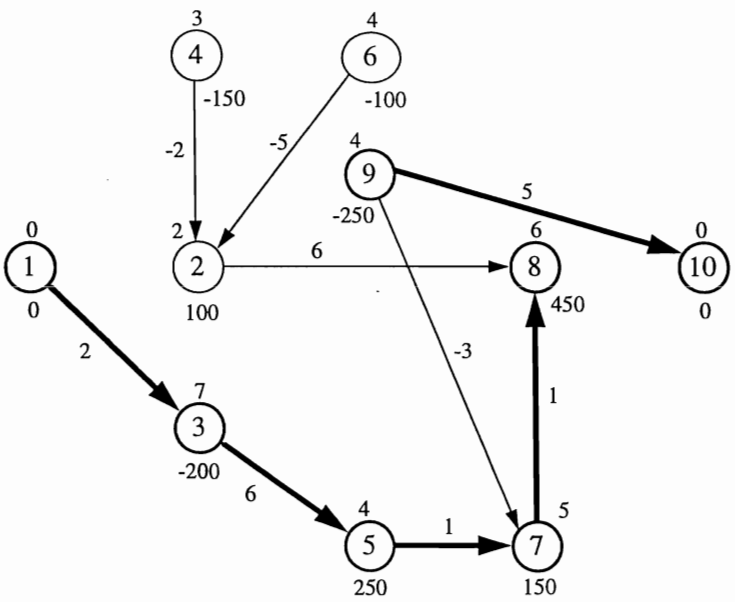

Fig. 7. The intermediate current tree with an *npv* of 174.50

RECURSION (1)

$PT = \{1\}$, $DC = 0$, $A = \{1\}$.

RECURSION (3)

$PT = \{3\}$, $DC = -200\,e^{-0.02(2+7)} = -167.05$, $A = \{1,3\}$.

RECURSION (5)

$PT = \{5\}$, $DC = 250\,e^{-0.02(8+4)} = 196.66$, $A = \{1,3,5\}$.

RECURSION (7)

$PT = \{7\}$, $DC = 150\,e^{-0.02(9+5)} = 113.37$, $A = \{1,3,5,7\}$.

RECURSION (8)

$PT = \{8\}$, $DC = 450\,e^{-0.02(10+6)} = 326.77$, $A = \{1,3,5,7,8\}$.

RECURSION (2)

$PT = \{2\}$, $DC = 100\,e^{-0.02(4+2)} = 88.69$, $A = \{1,2,3,5,7,8\}$.

RECURSION (4)

$PT = \{4\}$, $DC = -150\,e^{-0.02(6+3)} = -125.29$, $A = \{1,2,3,4,5,7,8\}$.

$PT = \{2,4\}$, $DC = 88.69 - 125.29 = -36.60$.

RECURSION (6)

$PT = \{6\}$, $DC = -100\,e^{-0.02(9+4)} = -77.11$, $A = \{1,2,3,4,5,6,7,8\}$.

$PT = \{2,4,6\}$, $DC = -36.60 - 77.11 = -113.70$.

$PT = \{2,4,6,8\}$, $DC = 326.77 - 113.70 = 213.06$

$PT = \{2,4,6,7,8\}$, $DC = 113.37 + 213.06 = 326.43$

RECURSION (9)

$PT = \{9\}$, $DC = -250\,e^{-0.02(12+4)} = -181.54$, $A = \{1,2,3,4,5,6,7,8,9\}$.

RECURSION (10)

$PT = \{10\}$, $DC = 0$, $A = \{1,2,3,4,5,6,7,8,9,10\}$.

$PT = \{9,10\}$, $DC = -181.54 + 0 = -181.54$

$PT = \{2,4,6,7,8,9,10\}$, $DC = 326.43 - 181.54 = 144.89$

$PT = \{2,4,5,6,7,8,9,10\}$, $DC = 196.66 + 144.89 = 341.55$

$PT = \{2,3,4,5,6,7,8,9,10\}$, $DC = -167.05 + 341.55 = 174.50$

$PT = \{1,2,3,4,5,6,7,8,9,10\}$, $DC = 174.50 + 0 = 174.50$

Report the optimal schedule with activity starting times $\{0,4,2,6,8,9,9,10,12,17\}$ and net present value 174.50. The optimal current tree is the one given in Fig. 7.

STOP.

6. Computational experience

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 75kb of memory, whereas only 83kb are reserved for data storage, which makes it very well suited to run on any platform, even those with a small amount of available memory. In order to validate the search procedure, we solved one of the problem sets used by Herroelen et al. (1996b) to validate their solution procedure for the unconstrained max-npv case with zero-lag finish-start precedence relations. The problem set consists of 98 of the 110 RCPSP instances assembled by Patterson (1984) with up to 27 (non-dummy) activities, for which the resource requirements have been deleted and randomly generated cash flows have been added. The experiment showed identical results, and indicated that our procedure was almost 12 times slower (5.11 milliseconds on the average vs. 0.43 milliseconds on the average) for that problem set than the procedure of Herroelen et al. (1996b), the latter, however, being unable to solve GPR-instances. The fact that our procedure is slower on problem instances without GPRs is quite logical and due to the fact that it is developed especially for problem instances with GPRs, in which cycles may occur, whereas the procedure of Herroelen et al. (1996b) can ignore cyclic structures due to the absence of cycles in project networks with zero-lag finish-start precedence relations.

6.1. Benchmark problem set

Schwindt (1995) developed a random 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). ProGen/max can generate RCPSP instances, multiple-mode RCPSP (MRCPSP) instances, RCPSP-GPR instances as well as MRCPSP-GPR (a combination of multiple modes and GPRs) instances. In addition, instances of the resource levelling problem with generalized precedence relations (RLP-GPR) and the resource availability cost problem with generalized precedence relations (RACP-GPR) can be generated. 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. Obviously, the resource-based measures given in the second column are irrelevant for the unconstrained max-npv project scheduling problem.

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)

Three RCPSP-GPR problem sets have already been generated using ProGen/max. The first set (Schwindt, 1996) consists of 1080 instances, of which 540 are generated using the DIRECT method and 540 using the CONTRACT method. The second set (Franck and Neumann, 1996) consists of 1440 problem instances generated using the DIRECT method. The third set (De Reyck and Herroelen, 1996b) consists of 7200 problem instances generated using the DIRECT method, which allows for a more extensive testing of the impact of several problem characteristics. We will use the third benchmark set to test the effectiveness and efficiency of our solution procedure.

The control parameters used to generate the 7200 instances are 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 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 $[-500,+500]$.

[1] Schwindt (1996) uses an estimator for the restrictiveness (Thesen, 1977) as a network complexity measure. However, De Reyck (1995) 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.

Table II. The parameter settings of the benchmark problem set

Control parameter	Value
# activities	10; 20; 30; 50; 100
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

6.2. Basic results

The results are given in Tables III through V. The reported values are the average CPU time in *milli*seconds, its range and its standard deviation.

Table III. The impact of the number of activities

Activities	# problems	Average CPU time	CPU time range	Standard Deviation
10	1440	0,94	[0 – 3]	0,20
20	1440	5,69	[3 – 10]	0,84
30	1440	17,20	[11 – 35]	2,51
50	1440	137,38	[110 – 270]	19,47
100	1440	836,12	[550 – 1,843]	181,60

A first observation we can make from Table III is that the required CPU times are very small. The average computation times are smaller than 1 second, even for the 100-activity projects. However, we should keep in mind that the unconstrained max-npv project scheduling problem (with GPRs) is probably not a goal by itself. Its solution may be used to compute an upper bound on the project *npv* for a resource-constrained project scheduling problem with

discounted cash flows (and GPRs), to be solved by an optimal procedure such as branch-and-bound. In that case, the unconstrained problem should be solved in every (undominated) node of the branch-and-bound tree, which may run in the thousands. Therefore, we should be able to execute the algorithm thousands of times within acceptable computation times, which is, as Table III indicates, clearly the case. Notice also the very low standard deviations and small ranges, reflecting a very robust behaviour of the procedure over the different problem instances.

Table III reveals that the number of activities has a strong impact on the required computation time. Moreover, Table IV shows a positive correlation between *OS* and the required CPU time: when *OS* increases, the problem becomes harder. The more dense the project network becomes, the more recursion steps are needed and, consequently, the more computation time is spent. These findings are completely in line with the results for the case with zero-lag finish-start precedence relations only reported by Herroelen et al. (1996b), who also found that an increased network complexity (density), either in the form of a higher complexity index *CI* (De Reyck and Herroelen, 1996a) or a higher order strength *OS*, leads to an increase in computational requirements. The effect of the percentage of maximal time lags as a cyclic network-based measure can be observed from Table V. The addition of more maximal time lags adversely affects the efficiency of our procedure, thus reflecting an increased problem complexity. However, the effect of the percentage of maximal time lags is less pronounced than the effect of *OS*.

Table IV. The impact of the order strength

OS	# problems	Average CPU time	CPU time range	Standard Deviation
0.25	2400	177,56	[0 – 1,542]	288,00
0.50	2400	199,02	[0 – 1,592]	328,95
0.75	2400	221,82	[0 – 1,843]	373,67

Table V. The impact of the percentage of maximal time lags

Max. time lags	# problems	Average CPU time	CPU time range	Standard Deviation
0 %	1800	178,33	[0 – 1,232]	288,60
10 %	1800	198,46	[0 – 1,562]	329,16
20 %	1800	209,18	[0 – 1,843]	352,37
30 %	1800	211,91	[0 – 1,642]	354,84

6.3. The effect of the distance matrix computation

The major part of the required computation time is needed to calculate the distance matrix (STEP 1). Table VI displays the portion of the total CPU time spent on (initializing and) calculating the distance matrix.

Table VI. Portion of the CPU time needed for computing the distance matrix

Activities	Number of problems	% of CPU time needed to compute the distance matrix
10	1440	65 %
20	1440	77 %
30	1440	77 %
50	1440	77 %
100	1440	62 %

Due to the fact that the distance matrix computation accounts for a major portion of the required CPU time, the effect of the network-based measures on the computational complexity of the max-npv project scheduling problem (Table IV and V) may be obscured or deflated, since these measures will not have any significant effect on the time needed to compute the distance matrix. Tables VII through IX therefore report the computational requirements excluding the time needed to calculate the distance matrix. Note that the computation times reported in Tables VII through IX will determine the extra amount spent in the nodes of a branch-and-bound algorithm for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations (RCPSPDC-GPR), if this algorithm is used to compute an upper bound on the net present value of the project network in each node, represented by its distance matrix.

Table VII. The impact of the number of activities

Activities	# problems	Average CPU time	CPU time range	Standard Deviation
10	1440	0,33	[0 – 2]	0,19
20	1440	1,33	[0 – 7]	0,79
30	1440	3,91	[0 – 22]	2,50
50	1440	31,84	[6 – 154]	19,39
100	1440	313,56	[20 – 1,292]	178,17

When we compare Tables III and VII, it is clear that the calculation of the distance matrix is largely responsible for the total computation time needed to solve these problem instances. When the time spent on calculating the distance matrix is removed from the reported CPU times, the effect of the network-based problem characteristics on the required computational effort of the recursive procedure becomes even more clear. From Table VIII, we again observe a positive correlation between *OS* and the required CPU time. Table IX now clearly indicates the effect of the percentage of maximal time lags as a cyclic network-based measure, although it is still not as pronounced as the effect of *OS*. For instance, the hardest problem in the set, with a CPU time of 1.29 seconds, has an *OS* of 0.75 but only has 20% maximal time lags.

Table VIII. The impact of the order strength

OS	# problems	Average CPU time	CPU time range	Standard Deviation
0.25	2400	49,30	[0 – 1,042]	99,06
0.50	2400	69,75	[0 – 1,071]	139,76
0.75	2400	91,52	[0 – 1,292]	184,06

Table IX. The impact of the percentage of maximal time lags

Max. time lags	# problems	Average CPU time	CPU time range	Standard Deviation
0 %	1800	49,35	[0 – 721]	95,80
10 %	1800	69,50	[0 – 1,042]	141,29
20 %	1800	79,87	[0 – 1,292]	166,65
30 %	1800	82,05	[0 – 1,082]	167,09

6.4. 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 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 unconstrained max-npv project scheduling problem, 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%. The effect on the average CPU time required to solve the 7200 problem instances is indicated in Fig. 8 (for the projects with up to 30 activities) and Fig. 9 (for the projects with 50 and 100 activities).

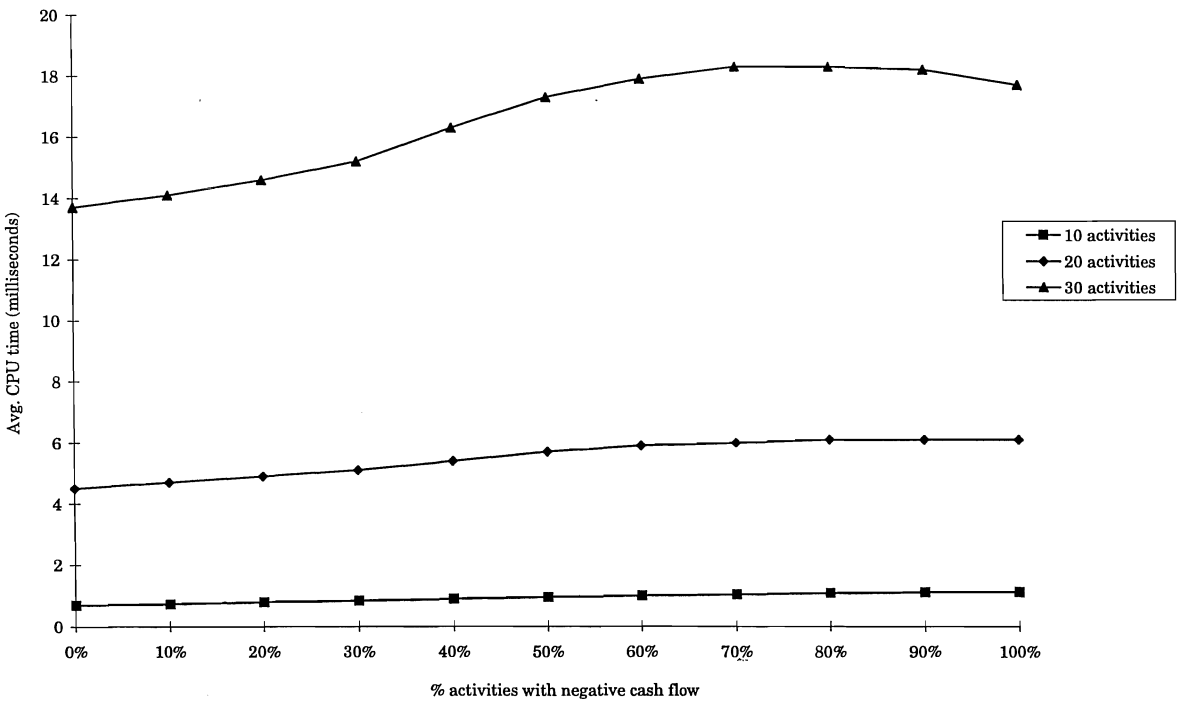

Fig. 8. The effect of the % of activities with a negative cash flow on the average CPU time

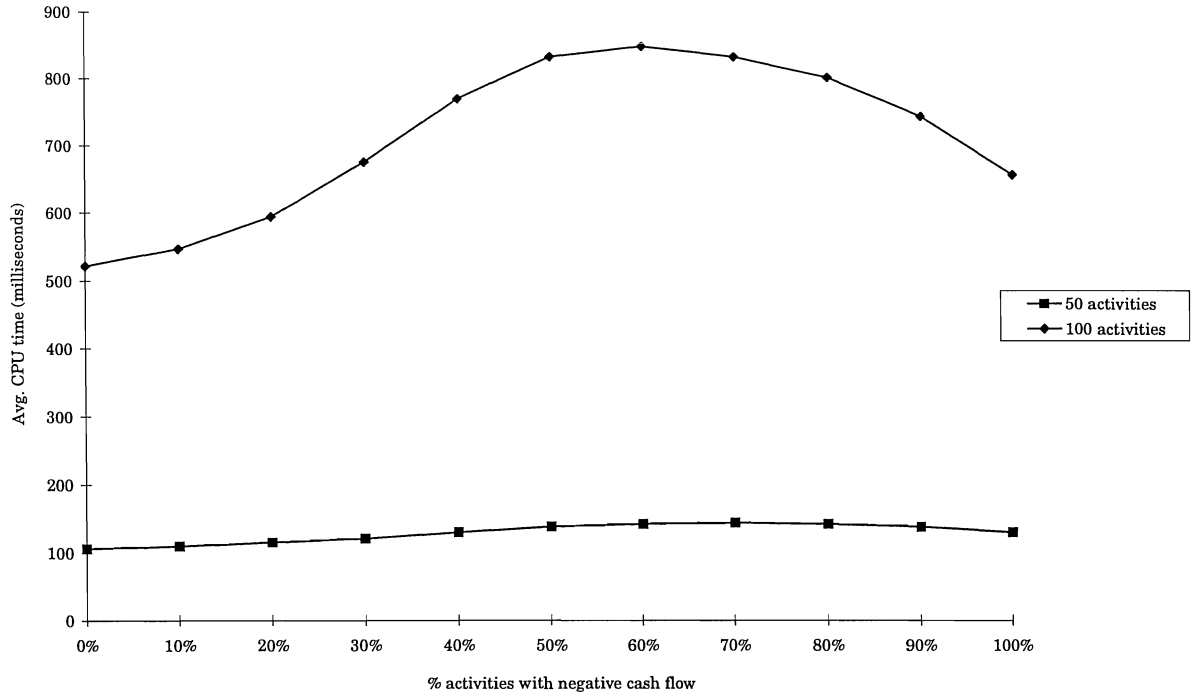

Fig. 9. The effect of the % of activities with a negative cash flow on the average CPU time

As can be seen from Fig. 8 and 9, the percentage of activities with a negative cash flow has an impact on the computational complexity of the unconstrained max-npv project scheduling problem, albeit not very significant. Clearly, if no activities with a negative cash flow are present, the problem becomes very easy since the earliest start schedule will always represent the optimal solution, i.e. no forward shifts of the activities and no recursion steps are necessary. Only the distance matrix needs to be computed. If all activities carry negative cash flows, the problem is also relatively easy, because all activities can be shifted forward till one of them hits the deadline. If, however, activities with positive and negative cash flows are mixed, the problem becomes harder. This is why, in the experiments reported above, we have set the number of activities with a negative cash flow to 50%, representing more or less the hardest problem instances.

As mentioned before, the effect of the problem characteristics on the computational complexity of the max-npv project scheduling problem may be obscured by the fact that the calculation of the distance matrix is included in the reported computation times. Fig. 10 and 11 show graphs similar to Fig. 8 and 9, but with the calculation of the distance matrix excluded from the reported average computation times. The complexity of the distance matrix computation is independent of the number of activities with a negative cash flow, since no cash flow considerations are taken into account. As can be seen from Fig. 10 and 11, the effect of the percentage of activities with a negative cash flow on the average CPU time excluding the time needed to compute the distance matrix is much more pronounced.

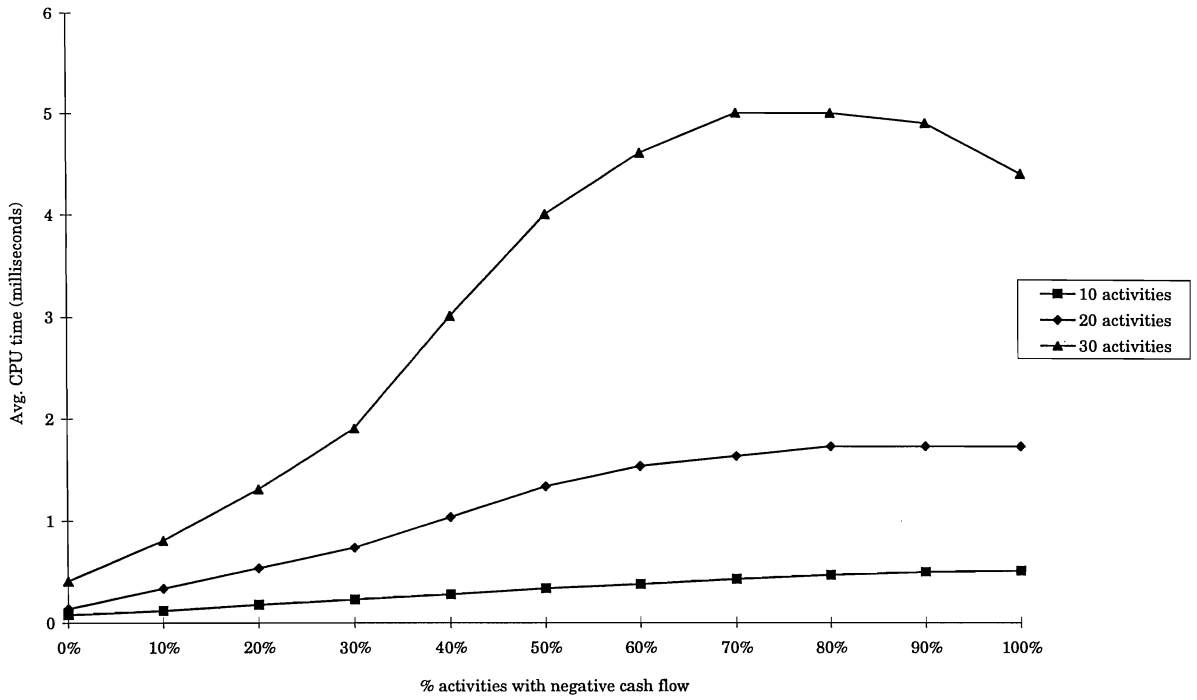

Fig. 10. The effect of the % of activities with a negative cash flow on the average CPU time

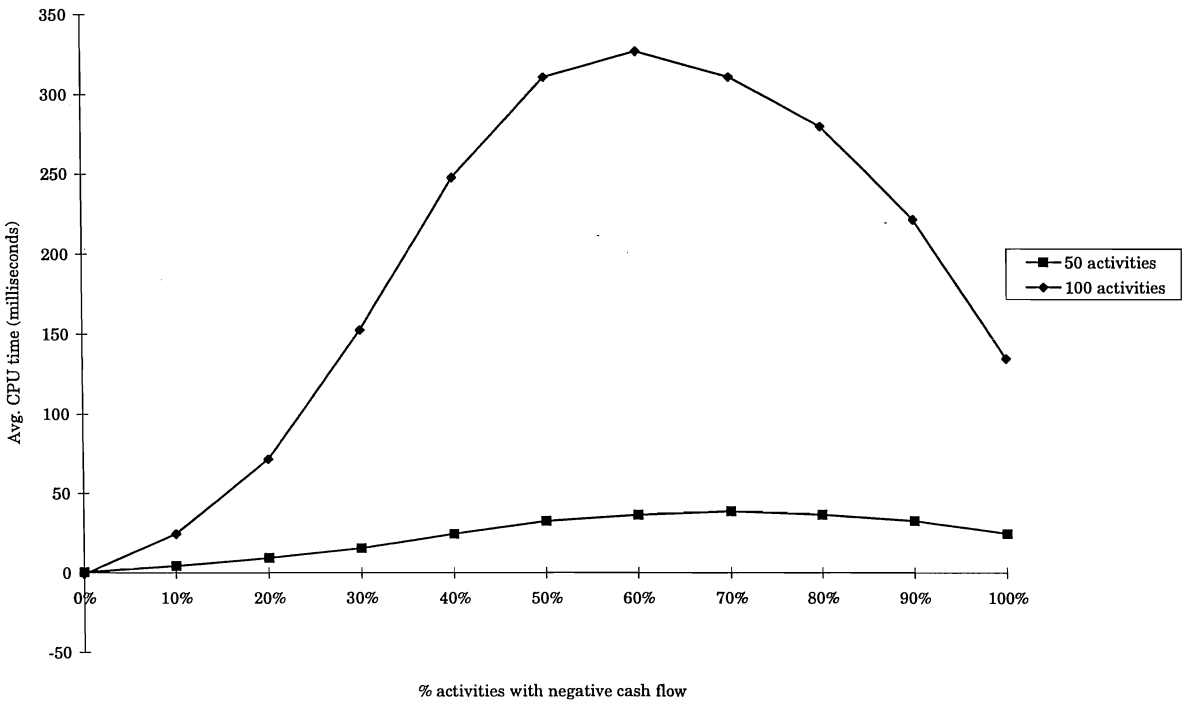

Fig. 11. The effect of the % of activities with a negative cash flow on the average CPU time

7. Conclusions

The unconstrained max-npv project scheduling problem involves the scheduling of the activities of a project in order to maximize its net present value. In this paper, we presented a model and an optimal solution procedure for the unconstrained max-npv project scheduling problem with generalized precedence relations (GPRs), which allows for the introduction of arbitrary minimal and maximal time lags between the start and completion of activities. We described how one of the most efficient optimal procedures for the unconstrained max-npv project scheduling problem with zero-lag finish-start precedence constraints only, namely the procedure of Herroelen et al. (1996b), can be adapted to cope with generalized precedence relations.

Computational results are reported which show the effectiveness and efficiency of the proposed procedure, in that it is able to solve randomly generated problem instances up to 100 activities with very modest computation time and memory requirements. Even 100-activity problem instances can be solved in, on the average, less than 1 second of CPU time on a Pentium-60 personal computer. The promising results indicate that the proposed procedure is very well suited to be implemented for the calculation of upper bounds on the project *npv* in a more general solution procedure for the resource-constrained case, i.e. the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations (RCPSPDC-GPR).

References

Ahuja, R.K., Magnanti, T.L. and Orlin, J.B., 1989, "Network flows", in: Nemhauser, G.L., Rinnooy Kan, A.H.G. and Todd, M.J. (Eds.), *Handbooks in Operations Research and Management Science*, Elsevier, Amsterdam, 258-263.

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.

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.

Baroum, S.M., 1992, "An Exact Solution Procedure for Maximizing the Net Present Value of Resource-Constrained Projects", unpublished Ph.D. dissertation, Indiana University.

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

De Reyck, B., 1995, "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, "Computational experience with a branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations", Research Report 9628, Department of Applied Economics, Katholieke Universiteit Leuven.

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. 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.

Grinold, R.C., 1972, "The payment scheduling problem", *Naval Research Logistics Quarterly*, 19(1), 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 9603, Department of Applied Economics, Katholieke Universiteit Leuven.

Icmeli, O. and Erengüç, S.S., 1994, "A tabu search procedure for resource-constrained project scheduling with discounted cash flows", *Computers and Operations Research*, 21, 841-853.

Icmeli, O. and Erengüç, S.S., 1995, "A branch-and-bound procedure for the resource-constrained project scheduling problem with discounted cash flows", Working Paper, Cleveland State University.

Jensen, P.A. and Barnes, J.W., 1987, *Network Flow Programming*, Robert E. Krieger Publishing Company, Florida.

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

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.

Mastor, A. A., 1970, "An experimental and comparative evaluation of production line balancing techniques", *Management Science*, 16 (11), 728-746.

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., 1984, "A Comparison of Exact Procedures for Solving the Multiple-Constrained Resource Project Scheduling Problem", *Management Science*, 20, 767-784.

Patterson, J.H., Slowinski, R., Talbot, F.B., and Weglarz, J., 1989, "An Algorithm for a General Class of Precedence and Resource Constrained Scheduling Problems", Part I, Chapter 1 in Slowinski, R. & Weglarz, J. (eds.), *Advances in Project Scheduling*, Elsevier Science Publishers, Amsterdam, 3-28.

Patterson, J.H., Talbot, F.B. , Slowinski, R. and Weglarz, J., 1990, "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.

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., 1996, private communication.

Sepil, C. and Kazaz, B., 1994, "Project scheduling with discounted cash flows and progress payments", Working Paper 94-7, Middle East Technical University, Ankara.

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.

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.

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

Figure Captions

Table Captions