Transaction E: Industrial Engineering
Vol. 16, No. 1, pp. 55–64

Research Note

A Branch and Bound Algorithm for the Weighted Earliness-Tardiness Project Scheduling Problem with Generalized Precedence Relations

B. Afshar Nadjafi[1,*] and S. Shadrokh[1]

Abstract. *In this paper, an exact solution procedure is presented for the Weighted Earliness-Tardiness Project Scheduling Problem (WETPSP) with Generalized Precedence Relations (WETPSP-GPR), taking into account the time value of money (WETPSPDC-GPR) and a fixed deadline for the project. The WETPSP-GPR extends the WETPSP to arbitrary minimal and maximal time-lags between the starting and completion times of activities. We present a new depth-first Branch and Bound (B&B) algorithm for an extended form of the problem, in which the time value of money is taken into account by discounting cash flows, and minimum, as well as maximum, time-lags between different activities may be given. The algorithm is extended with two bounding rules in order to reduce the size of the branch and bound tree. Finally, some test problems are solved and computational results are reported.*

Keywords: *Project scheduling; Branch and bound; Net present value; Generalized recedence relations.*

INTRODUCTION

Since the introduction of cash flows in project scheduling by Russell [1], the maximization of the Net Present Value (NPV) has gained increasing attention throughout the literature. This has led to a large number of algorithms, including those presented by Baroum and Patterson [2,3], Elmaghraby and Herroelen [4], Etgar and Shtub [5,6], Herroelen and Gallens [7], Icmeli and Erenguc [8,9], Ozdamar et al. [10], Padman et al. [11,12], Pinder and Maruchech [13], Russell [14], Shtub and Etgar [15], Smith-Daniels et al. [16], Ulusoy and Ozdamar [17] and Vanhoucke et al. [18,19].

This paper addresses the Weighted Earliness-Tardiness Project Scheduling Problem (WETPSP) for the extended form in which the time value of money is taken into account by continuous discounting of the cash flow, and the minimum, as well as maximum, time-lags between different activities may be given. The literature on solution methods for the WETPSP is scant. Vanhoucke et al. [20,21] have developed an exact recursive search algorithm for the basic form. Vanhoucke et al. [22] have exploited the logic of the recursive procedure for solving the WETPSP in their branch and bound procedure for maximizing the net present value of a project, in which progress payments occur. Kazaz and Sepil [23] solve the problem using Benders decomposition, while Sepil and Ortac [24] developed heuristics for the problem under renewable resource constraints. The paper is organized as follows. First, the concept of GPRs is presented and the terminology used is clarified. Then, the temporal analysis of activity networks with GPRs is introduced and WETPSP is described. Following that, a branch and bound procedure and a numerical example and computational results are represented, respectively. Finally, the paper is concluded.

1. *Department of Industrial Engineering, Sharif University of Technology, Tehran, P.O. Box 11155-9414, Iran.*
*. *Corresponding author. E-mail: afsharnb@mehr.sharif.edu*

Received 19 June 2006; received in revised form 6 September 2007; accepted 23 April 2008

GENERALIZED PRECEDENCE RELATIONS

In practice, it is often necessary to specify other than the finish-start precedence relations with zero time-lag used in PERT and CPM. In accordance with Elmaghraby and Kabmurowski [25], we denote them 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 represent a minimal or maximal

time-lag between a pair of activities. Let one use S_i to denote the start time and f_i to denote the finish time of activity i, $(1 \leq i \leq n)$. Then, the minimal and maximal time-lags between the two activities, i and j, have the following form:

$$S_i + SS_{ij}^{\min} \leq S_j \leq S_i + SS_{ij}^{\max},$$

$$S_i + SF_{ij}^{\min} \leq f_j \leq S_i + SF_{ij}^{\max},$$

$$f_i + FS_{ij}^{\min} \leq S_j \leq f_i + FS_{ij}^{\max},$$

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

The GPRs specifying a maximal time-lag can be represented by a minimal time-lag in the opposite direction. The GPRs can be represented in standardized form by transforming them to, for instance, minimal start-start precedence relations.

TEMPORAL ANALYSIS

Consequently, assume a project represented in AON format by a directed graph $G = \{N, A\}$, where the set of nodes, N, represents activities and the set of arcs, A, represents Generalized Precedence Relations (GPRs). The non-preemptable activities are numbered from the dummy start activity, 1, to the dummy end activity, n, and are topologically ordered. Let d_i denote the fixed duration of activity i, $(1 \leq i \leq n)$.

A path $[i_s, i_k, i_l, \cdots, i_t]$ is called a cycle if $s = t$. 'Path' refers to a directed path, and 'cycle' refers to a directed cycle. The length of a path (cycle) is defined as the sum of the lags associated with the arcs belonging to that path (cycle). 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 non-negative length from node 1 to every other node 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 or d_i, respectively. A schedule, $S = (s_1, s_2, \cdots, s_n)$, is called time-feasible, if the activity starting times satisfy all GPRs, i.e. if they satisfy the following conditions:

$$S_i \geq 0, \qquad \forall i \in N, \tag{1}$$

$$S_i + l_{ij} \leq S_j, \qquad \forall (i,j) \in A, \tag{2}$$

where Equation 1 ensures that no activity starts before the current time (time zero), and Equation 2 denots the GPRs in standardized form. The minimum starting times, $(s_1, s_2, \cdots, s_n)$, satisfying both Equations 1 and 2, form the early start time, EST=(EST_1, $\text{EST}_2, \cdots, \text{EST}_n$), associated with the temporal constraints. The calculation of an EST 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, if G has no cycle of positive length [26]. Such cycles would unable one to compute activity starting times that satisfy Equations 1 and 2. Therefore, if we calculate matrix $D = [d_{ij}]$, where d_{ij} denote the longest path from node i to node j, a positive path length from any 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 D can be done by using standard graph algorithms for the longest paths in the networks, for instance by the Floyd-Warshall (see [27]). Let $d_{ij}^{(k)}$ represent the length of a longest path from node i to node j, subject to the condition that this path uses only the nodes $1, 2, \cdots, k-1$ as internal nodes. Clearly, $d_{ij}^{(n+1)}$ represents the actual longest path distance from node i to node j. If one starts with matrix $D^{(1)} = [d_{ij}^{(1)}]$, with:

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

we can compute $D = D^{(n+1)}$, according to the updating formula, $d_{ij}^{(v)} = \max\{d_{ij}^{(v-1)}, d_{iv}^{(v-1)} + d_{vj}^{(v-1)}\}$ $(i, j, l = 1, 2, \cdots, n)$. If $d_{ii} = 0$ for all $i = 1, 2, \cdots, n$, there exists a time-feasible schedule. The EST is given by the numbers in the upper row of D : EST $= (d_{11}, d_{12}, \cdots, d_{1n})$. Computing D takes $O(n^3)$ time.

A latest allowable start schedule (LST) can be computed, based on the network with all arcs reversed and with the condition that $\text{EST}_n = \text{LST}_n$. Note that, for calculating the LST, the maximal time-lags between activities have to be taken into account.

PROBLEM DESCRIPTION

The deterministic Weighted Earliness-Tardiness Project Scheduling Problem (WETPSP) involves the scheduling of project activities, in order to minimize the weighted earliness-tardiness costs of the project in the absence of resource constraints.

Basic form of the WETPSP

In the basic form of the problem, we suppose that all precedence relations are finish-start with a time-lag of zero, and the time value of money is not taken into account. If we let h_i denote the deterministic due date of activity i, $(1 \leq i \leq n)$, the earliness of activity i can be computed as:

$$E_i = \max(0, h_i - f_i).$$

The tardiness of activity i can be computed as:

$$T_i = \max\,(0, f_i - h_i).$$

If we let e_i and t_i denote the per unit earliness and tardiness cost of activity i, respectively, the total earliness-tardiness cost of activity i equals:

$$e_i E_i + t_i T_i.$$

It is assumed that $h_1 = 0$, $h_n = \infty$, $e_1 = t_1 = \infty$ and $e_n = t_n = 0$. Problem $cpm|$early/tardy can then be formulated as follows [21]:

$$\min \sum_{i=2}^{n} (e_i E_i + t_i T_i), \tag{3}$$

s.t :

$$f_i \leq f_j - d_j, \qquad \forall (i,j) \in A, \tag{4}$$

$$E_i \geq h_i - f_i, \qquad \forall i \in N, \tag{5}$$

$$T_i \geq f_i - h_i, \qquad \forall i \in N, \tag{6}$$

$$f_1 = 0, \tag{7}$$

$$f_i \geq 0,\ E_i \geq 0,\ T_i \geq 0, \qquad \forall i \in N. \tag{8}$$

The objective in Equation 3 is to minimize the weighted earliness-tardiness cost of the project. The constraint set given in Equation 4 imposes the finish-start precedence relations among activities. Equations 5 and 6 compute the earliness and tardiness of each activity. Equation 7 forces the dummy start activity to end at time zero. Equation 8 ensures that the activity finish times and the earliness and tardiness of activities assume nonnegative integer values.

Extended form of WETPSP (WETPSP-GPR)

When dealing with the NPV criterion, the time value of money is taken into account by discounting the cash flows. The value of an amount of money is a function of the time of receipt or disbursement of cash. In order to calculate the value of NPV, a discount rate, α, has to be chosen, which represents the return, following investment in the project, rather than, e.g., in securities. Then, the continuous discounted factor, $e^{-\alpha T}$, denoted the present value of a dollar to be paid at the end of period T using a discount rate, α. Figure 1 shows that the NPV of the early/tardy penalty costs associated with an activity changes, with respect to the activity completion times.

The objective of WETPSPDC is to find a schedule such that the NPV of the project is minimized. The way of calculating the value of NPV depends on the payment model considered. It is supposed that the cost, due to the earliness or tardiness of each activity, will impose on the progress model.

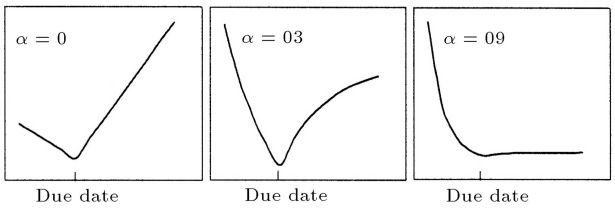

Figure 1. Early-tardy cost curve showing the effect of discount rate on NPV.

The objective of the WETPSPDC-GPR is to schedule a number of activities, in order to minimize the Net Present Value (NPV) of the project, subject to generalized precedence relations and a fixed deadline.

The following notations are for weighted earliness-tardiness project scheduling with discounted cash flow and generalized precedence relations (WETPSPDC):

n	number of activities,
d_i	duration of activity i,
h_i	due date of activity i,
s_i	start time of activity i (integer decision variable),
EST_i	earliest start time of activity i,
LST_i	latest start time of activity i,
e_i	per unit earliness cost of activity i,
t_i	per unit tardiness cost of activity i,
α	discount rate,
δ_n	deadline of the project,
SM	state matrix representing the precedence relations conflict between activities,
CA	set of conflicted activities in schedule,
Z	objective function,
Z^*	optimal objective function,
$S_a + l_{ab} \leq S_b$	denotes that activity b may start when its predecessor, a, has already started for a l_{ab} time units. (Standard form of GPRs),
$a \prec b$	denotes that activity a is the predecessor of activity b.

Using the above notation, the resource-unconstrained project scheduling problem with GPRs under the minimum discounted early-tardy penalty cost objective can be mathematically formulated as follows:

$$\min Z = \sum_{i=2}^{n} \left(e_i \sum_{k=S_i+d_i}^{h_i-1} e^{-\alpha k} + t_i \sum_{k=h_i+1}^{S_i+d_i} e^{-\alpha k} \right), \tag{9}$$

s.t:

$$S_i + l_{ij} \leq S_j, \qquad \forall (i,j) \in A, \tag{10}$$

$$S_n \leq \delta_n, \tag{11}$$

$$S_1 = 0, \tag{12}$$

$$S_i \in N, \qquad \forall i \in N. \tag{13}$$

The objective in Equation 9 minimizes the NPV of the project. Constraints 10 represent the GPRs. In order to restrict the project duration, we add a negotiated project deadline, δ_n, for dummy end activity n, given in Equation 11. Equation 12 forces the dummy start activity to start (finish) at time zero. Constraints 13 ensure that all activity start times assume nonnegative integer values.

BRANCH AND BOUND ALGORITHM

In this section, we give a description of the branch and bound procedure.

Initial Schedule

The process of constructing the tree starts with generating an initial and probably infeasible scheduling. In the proposed algorithm, the initial scheduling procedure sets the start time of each activity, i, at:

$$\begin{cases} \text{EST}_i; & \text{if } h_i - d_i \leq \text{EST}_i \\ h_i - d_i; & \text{if EST}_i < h_i - d_i \leq \text{LST}_i \\ \text{LST}_i; & \text{if LST}_i < h_i - d_i \end{cases}$$

and calculates its cost. Let P denote the level of the branch and bound tree. Then, in the initial schedule, we set $P = 0$. Although this schedule has minimum cost, it may be infeasible. The feasibility of a schedule can be evaluated by the State Matrix (SM). Each element of this matrix can be calculated as follows:

$$\text{SM}(i,j) = \begin{cases} s_j - s_i - l_{ij}; & (i,j) \in A \\ 0; & \text{Otherwise} \end{cases}$$

where l_{ij} denotes the minimal time-lag between start times of activities i and j. In this matrix, if activity i is the predecessor of activity j, and the time interval between the start times of activities i and j is equal to l_{ij}, then, one will have $\text{SM}(i,j) = 0$. If the time interval between the start times of activities i and $j(s_j - s_i)$ is less (greater) than l_{ij}, then one will have $\text{SM}(i,j) < 0 (\text{SM}(i,j) > 0)$. If all elements of SM are nonnegative, then, the current schedule is time-feasible.

Branching Strategy

In this subsection, we describe how to create new nodes of the enumeration tree and how to select a node for further branching. Branching is based on the evaluation of SM, which presents the feasibility of the current schedule, in order to obtain a feasible schedule. Nodes which represent time-infeasible project networks and which are not fathomed by any of the node pruning rules described below, lead to a new branching. In each level of the B&B tree, if there is more than one such node, the ties are broken in favor of children who have been created with more right shift. The branching process takes place from the selected child, who has now become the current schedule, and its children are generated. Among these children, the best one is selected again. This branching process continues until all nodes are pruned, based on pruning rules. The only difference between any current schedule and its parent is that it includes one new decision about two activities that have a time conflict, $(\text{SM}(i,j) < 0)$. Time conflicts are resolved using the concepts of left shift and right shift. Assume, for example, that, in a certain node, two activities, i and j, have time conflict $(\text{SM}(i,j) < 0)$ and activity i is the predecessor of activity j. Suppose that the duration of the time conflict between these activities is an l time unit. We consider, at most, $(l+1)$ alternatives to resolve this time conflict. We can, theoretically, enumerate an infinite number of situations to solve the conflict, but it is clear that any solution with more than l_{ij} lag between these two activities would be dominated by the minimum cost solution inside the $l+1$ alternative. Longer lags between i and j are justifiable only when we need to resolve other conflicts, since longer lags mean greater cost. Resolving other conflicts would be considered in other nodes of the B&B tree. Therefore, it is enough to enumerate only $l+1$ alternatives. In the first node, activity j is shifted to the right l time unit. In the second node, activity j is shifted to a right $(l-1)$ time unit and activity i is shifted to the left 1 time unit. In the kth node, activity j is shifted to the right $(l-k+1)$ time unit and activity i is shifted to the left $(k-1)$ time unit. Finally, in the $(l+1)$th node, activity i is shifted to the left l time unit. In the shifting process, one must be aware that, for each activity i, the activity start time should not be later than LST_i or earlier than EST_i.

In order to avoid creating new conflicts in the shifting process, it is necessary to devise preventive approaches. Therefore, it is assumed that, in the right (left) shifting of each activity i, if one is faced with new conflicts, the associated activities must be shifted to the right (left) too. For example, this can be shown, graphically, as given in Figure 2, in which we suppose that $a \prec b$, $b \prec c$, $b \prec d$, $l_{ab} = d_a$, $l_{bc} = l_{bd} = d_b$ and

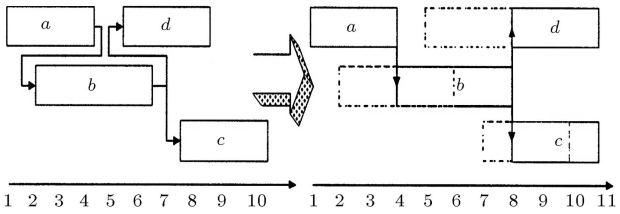

Figure 2. Shifting process.

activity b is selected for a right shift. Consequently, the following lemma applies.

Lemma

The above shifting strategy will lead to the complete enumeration of the search tree.

Proof

See the Appendix.

Pruning Rules

If it can be established that further branching from a node cannot lead to an optimal solution, then, the node can be pruned away. In this subsection, we present two rules for pruning the enumeration tree.

Incumbent Rule

The first pruning rule, the incumbent rule, is based on the incumbent solution, which is the best solution, up to a certain point, in the solution process. In this case, it prunes away any schedule that could potentially lead to an equal or worse solution, compared to an incumbent solution. In order to apply this pruning rule, a procedure has been used to create a Lower Bound (LB) for the problem. The corresponding bounding procedure is based on a concept known as a conflict set. To describe the conflict set, which calculates the lower bound associated with the schedule scheme, we introduce some notations.

A subset, X, of activities compose a conflict set if a) activity i is the predecessor of activity j and these activities have time conflict ($\mathrm{SM}(i,j) < 0$), b) activity i is the predecessor of some activities, $k_1, k_2, k_3, \cdots, k_r$, and their GPRs are ignored, or c) activity j is the successor of some activities, $k_1, k_2, k_3, \cdots, k_r$, and their GPRs are ignored. These three types of conflict set are given in Figure 3, graphically.

In order to calculate a lower bound for each time-infeasible schedule, we follow a two-stage procedure. In the first stage, for a time-infeasible schedule, we identify the maximum length conflict sets, so that they do not have a common activity. On the other hand, each activity, i, may be a member of one conflict set. At this stage, other generalized precedence constraints, which have not been considered in any of the conflict sets, are relaxed. At the second stage, LB is calculated, based on the following propositions.

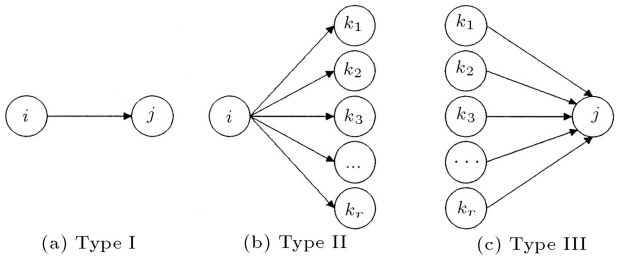

Figure 3. Types of conflict set.

Proposition 1

Associated with each Type I conflict set, where a generalized precedence relation between activities i and j is ignored, let $C(i,j)$ be the minimum cost incurred for resolving the time-conflict between activities i and j. If we suppose that the relative time conflict is a $l(i,j)$ time unit, then, $C(i,j)$ can be computed as follows:

$$C(i,j) = \min_{\nu=0}^{l(i,j)} \left\{ e_i \sum_{k=s_i+d_i-\nu}^{s_i+d_i-1} e^{-\alpha k} + t_j \sum_{k=s_j+d_j+1}^{s_j+d_j+[l(i,j)-\nu]} e^{-\alpha k} \right\}.$$

Let $\bar{\lambda}(\tau)$ be the minimum cost incurred for resolving the τth conflict set in the current schedule. Thus, regarding a Type I conflict set, it may be described as:

$$\bar{\lambda}(\tau) = C(i,j).$$

Proposition 2

Associated with each Type II conflict set, where activity i is the predecessor of some activities, $k_1, k_2, k_3, \cdots, k_r$, and their generalized precedence relations are ignored, one can compute $\bar{\lambda}(\tau)$ as follows:

$$\bar{\lambda}(\tau) = \max_{j=1}^{r} \{C(i,k_j)\}.$$

Proposition 3

Associated with each Type III conflict set, in which activity j is the successor of some activities, $k_1, k_2, k_3, \cdots, k_r$, and their generalized precedence relations are ignored, one may derive:

$$\bar{\lambda}(\tau) = \max_{i=1}^{r} \{C(k_i,j)\}.$$

Proposition 4

Associated with each schedule, a Lower Bound (LB) may be written as follows:

$$\text{Lower Bound} = \sum_{\tau} \bar{\lambda}(\tau) + Z. \tag{14}$$

Dominance Rule

The second pruning rule, the dominance rule, is based on the fact that some nodes may repeat. Each node in the search tree represents the initial scheduling

extended with a set of activity shifts to resolve time conflicts. Therefore, it is possible that a certain node represents a project network which has been examined earlier at another node in the search tree. One way of checking whether two nodes represent the same project network is to check the State Matrix (SM). Identical sets of activity shifts lead to identical project networks. This rule applies when a node is compared to a previously examined node in another path of the search tree. This can be enforced by saving the information required during backtracking.

Algorithm

Having discussed all the necessary concepts of the algorithm, we present it with the pseudo-code below:

Initialization: Set $P = 0$, $Z^* = \infty$, optimal schedule= Φ;

Step 1 Generate the initial schedule, set it as the current schedule;

Step 2 Create CA and SM and calculate Z and LB, for the current schedule;

Step 3 If CA is empty and if $Z < Z^*$, go to Step 11;

Step 4 Find the first negative element in SM, called SM(i, j), representing the first time conflict, between activities i and j;

Step 5 Expand the current schedule (generate all schedules directly reachable from the current schedule, by all possible shifting activities, i and j. For resulting schedules, set $P = P + 1$, create their CA and SM and calculate their Z and LB);

Step 6 Find the schedule at level P, which is not fathomed. (Ties are broken in favor of children who have been created with more right shifting.) Set it as the current schedule and go to Step 8;

Step 7 If no result is found in Step 6, set $P = P - 1$. If $P = 0$, go to Step 13, else, go to Step 6;

Step 8 If any result is found in Step 6, try the incumbent rule (determine the conflict type and apply the appropriate lower bound for the node, according to Equation 14);

Step 9 If the result of the incumbent rule is negative, try the dominance rule;

Step 10 If the result of any of the above two rules is positive, fathom the current node and go to Step 6, else, go to Step 3;

Step 11 Set the current schedule as the optimal schedule and set $Z^* = Z$;

Step 12 If $P = 0$, go to Step 13, else, go to Step 6;

Step 13 If $Z^* = \infty$, print 'no possible solution', else, print the optimal schedule;

Step 14 End.

A NUMERICAL EXAMPLE

In this section, we determine an optimal schedule for a numerical example by means of the B&B procedure presented. Consider the example activity network given in Figure 4. The numbers above the nodes (activities) denote the activity duration, d_i. The label associated with the arcs indicates the GPRs. The activity network in a standard form is given in Figure 5. Table 1 shows the due date and the unit penalty cost of the activities. For ease of representation, we assume that the unit earliness costs are equal to the unit tardiness costs. The project deadline, δ_n, amounts to 25 and the monthly discount rate, α, equals 0.01 (1%).

At the initial level, $P = 0$, of the tree, we determine the initial scheduling by setting the start times of each activity, i, at:

$$\begin{cases} \mathrm{EST}_i; & \text{if } h_i - d_i \le \mathrm{EST}_i \\ h_i - d_i; & \text{if } \mathrm{EST}_i < h_i - d_i \le \mathrm{LST}_i \\ \mathrm{LST}_i; & \text{if } \mathrm{LST}_i < h_i - d_i \end{cases}$$

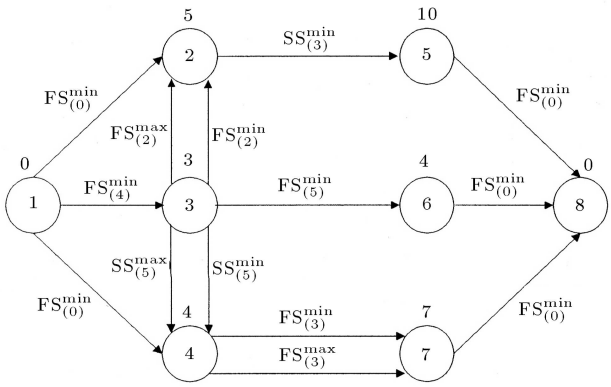

Figure 4. An activity network with GPRs.

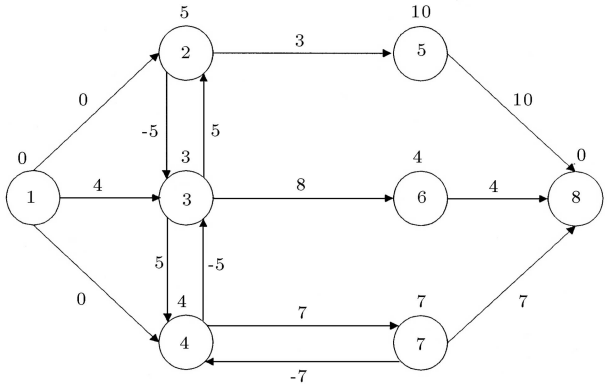

Figure 5. A standardized project network.

Table 1. Project's data for numerical example.

	Activity							
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**
h_i	0	13	9	15	23	17	24	25
$e_i = t_i$	∞	4	7	9	2	4	5	∞

An extended Gantt chart, associated with the root of the search tree, node 1, is displayed in Figure 6. Such a Gantt chart shows that this schedule is not feasible and there are two conflict sets. The State Matrix (SM) for initial scheduling is given in Figure 7. This matrix has some negative elements, which is another representation of infeasibility. Then, the set of conflicted activities are CA= {2, 3, 4, 6 and 7}. At the root of the enumeration tree, the algorithm computes the objective function (Z) and Lower Bound (LB) as 3.477 and 14.222, respectively. Table 2 shows a detailed computation of the lower bound in node 1.

The first negative element in SM corresponds to activities 2 and 3, which conflict with the 2 time unit. Consequently, the algorithm continues with Step 5. The level of the B&B tree is increased, i.e. $P = 1$ and three descendant nodes will be generated. In node 2, activity 2 is shifted to a right two time unit. In node 3, activity 2 is shifted to a right one time unit, and activity 3 is shifted to a left one time unit. Finally, in node 4, activity 3 is shifted to a left two time unit. We create a new node of the enumeration tree for each alternative of this decision and select the best alternative (node 2) for further branching. The State Matrix (SM) and set of Conflicted Activities (CA) are updated for each node and a lower bound on the NPV is computed. The resulting tree is given in Figure 8.

We have coded the B&B procedure in the MATLAB version 6.5 under windows XP. The complete B&B tree for the example is given in Figure 9.

In node 4, a first feasible schedule is found with a NPV of 19.14, corresponding with feasible start times (0, 11, 6, 11, 14, 14, 18 and 25). A search of all trees shows that this schedule is optimal, too. Other feasible schedules are found in nodes 5, 6, 7 and 8. Also, nodes 7 and 8 can be fathomed, because of the two pruning rules described previously.

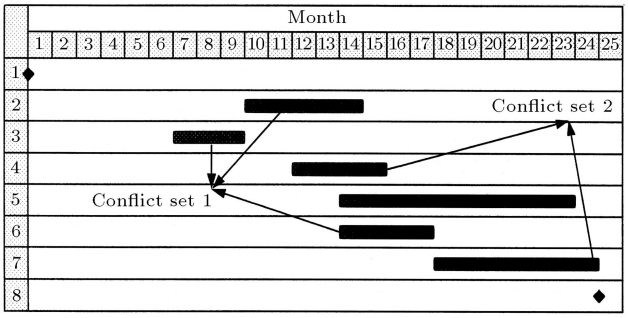

Figure 6. Extended Gantt chart for initial scheduling.

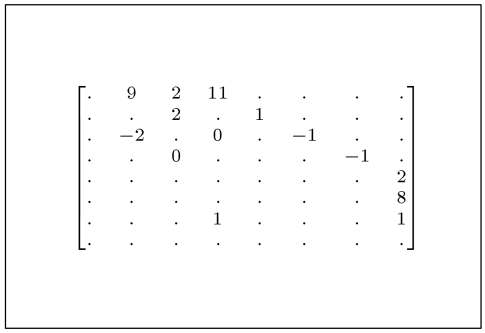

Figure 7. SM for initial scheduling.

COMPUTATIONAL RESULTS

In order to validate the proposed B&B method for the WETPSPDC-GPR, a problem set, consisting of 120 problem instances, was generated. This problem set was consisted equally of 40 instances with 10, 30 and 50 activities. The problem set was extended with unit earliness-tardiness penalty costs for each activity, which were randomly generated between 1 and 10. The due dates were generated in the same way as described by Vanhoucke et al. [21]. First, a maximum due date was obtained for each project by multiplying the

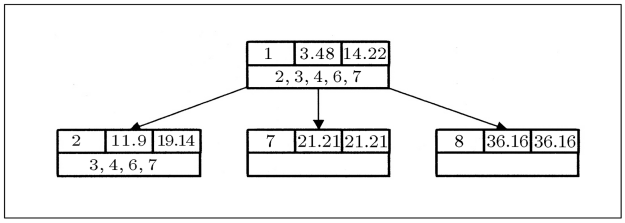

Figure 8. Branching process in initial schedule.

Table 2. Detailed computation of lower bound in node 1.

Conflict Set No (τ)	Type	Members	$C(i,j)$	$\bar{\lambda}(\tau)$	Lower Bound (LB)
1	II	3 and 2	6.851	6.851	6.851+3.894+3.477=14.222
		3 and 6	3.341		
2	I	4 and 7	3.894	3.894	

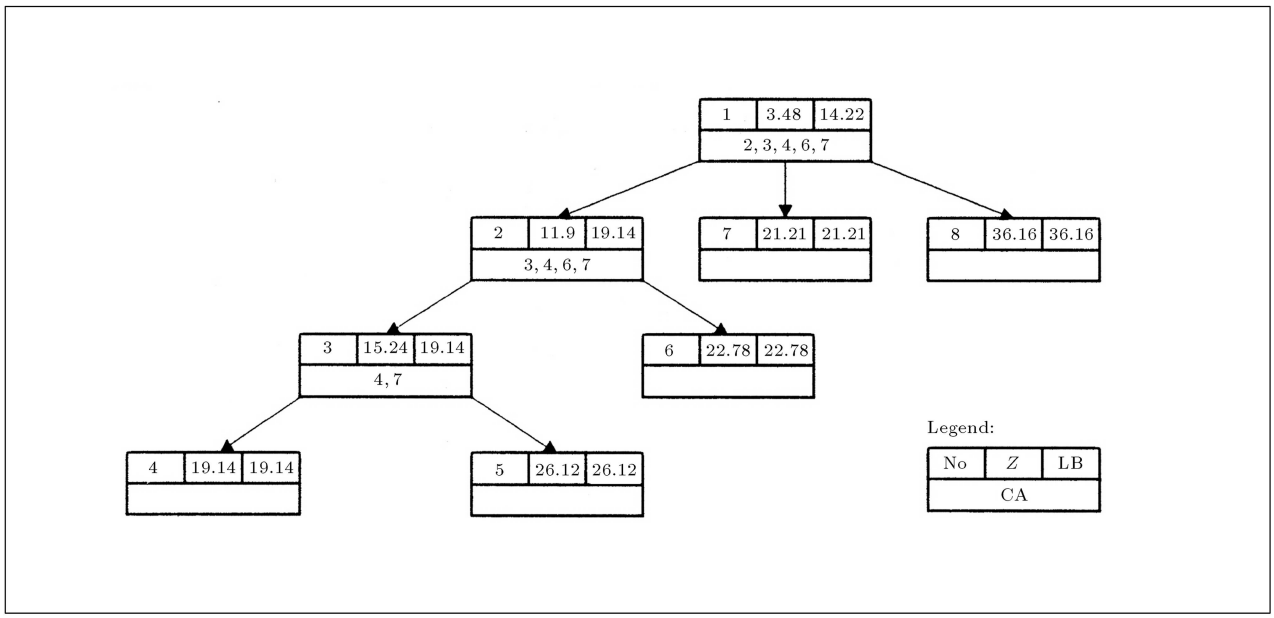

Figure 9. Branch and bound tree.

critical path length by 1.5. Subsequently, we generate random numbers between 1 and the maximum due date. The numbers are sorted and assigned to the activities in increasing order. Activity durations are randomly selected between 1 and 10. The maximum number of predecessors and successors are supposed as 3.

We have coded the B&B procedure in the MATLAB version 6.5. The problem set has been solved under windows XP on a personal computer with a Pentium IV, 1.7 GHz processor. Table 3 represents the average CPU-time, in seconds, for a different number of activities.

SUMMARY AND CONCLUSIONS

This paper reports on an exact B&B procedure for an extended form of the problem, cpm|early/tardy, i.e., an unconstrained project scheduling problem with continuous discounted negative cash flow, subject to Generalized Precedence Relations (GPRs). Negative cash flows occur when an activity is completed prior to, or later than, it's due date. The objective is to schedule the activities, in order to minimize the NPV, subject to the GPRs, with a fixed deadline on the project.

Table 3. The average CPU-time needed to solve the WETPSPDC-GPR.

Number of Activities	Number of Problems	Average CPU-Time
10	40	0.191
30	40	0.928
50	40	1.877

Branching is done by a right and left shifting process on two activities which have a time conflict, so that the predecessor activity has the smallest number. Two rules are used for node fathoming; incumbent rule and dominance rule. First, the pruning procedure computes the lower bound by making a disjunctive subset of activities, so called a conflict set. Second, the pruning procedure, fathomed previously, examined the node in another path of the search tree. Finally, the new B&B procedure is used for solving a numerical example.

REFERENCES

1. Russell, A.H. "Cash flows in networks", *Management Science*, **16**, pp. 357-373 (1970).
2. Baroum, S.M. and Patterson, J.H. "The development of cash flow weighted procedures for maximizing the net present value of a project", *Journal of Operations Management*, **14**, pp. 209-227 (1996).
3. Baroum, S.M. and Patterson, J.H. "An exact solution procedure for maximizing the net present value of cash flows in a network", Chapter 5, J. Weglarz, Ed., *Handbook on Recent Advances in Project Scheduling*, Kluwer Academic Publishers, Dordrecht, pp. 107-134 (1999).
4. Elmaghraby, S.E. and Herroelen, W. "The scheduling of activities to maximize the net present value of projects", *European Journal of Operational Research*, **49**, pp. 35-49 (1990).
5. Etgar, R., Shtub, A. and LeBlanc, L.J. "Scheduling projects to maximize net present value - the case of time independent, contingent cash flows", *European Journal of Operational Research*, **96**, pp. 90-96 (1996).
6. Etgar, R. and Shtub, A. "Scheduling projects activities to maximize net present value - the case of linear time

dependent, contingent cash flows", *International Journal of Production Research*, **37**, pp. 329-339 (1999).

7. Herroelen, W. and Gallens, E. "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**, pp. 274-277 (1993).
8. Icmeli, O. and Erenguc, S.S. "A tabu search procedure for resource constrained project scheduling with discounted cash flows", *Computer and Operations Research*, **21**, pp. 841-853 (1994).
9. Icmeli, O. and Erenguc, S.S. "A branch and bound procedure for resource constrained project scheduling problem with discounted cash flows", *Management Science*, **42**, pp. 1395-1408 (1996).
10. Ozdamar, L., Ulusoy, G. and Bayyigit, M. "A heuristic treatment of tardiness and net present value criteria in resource-constrained project scheduling", *International Journal of Physical Distribution and Logistics Management*, **28**, pp. 805-824 (1994).
11. Padman, R., Smith-Daniels, D.E. and Smith-Daniels, V.L. "Heuristic scheduling of resource-constrained projects with cash flows", *Naval Research Logistics*, **44**, pp. 365-381 (1997).
12. Padman, R. and Smith-Daniels, D.E. "Early-tardy cost trade-offs in resource constrained projects with cash flows: An optimization-guided heuristic approach", *European Journal of Operational Research*, **64**, pp. 295-311 (1993).
13. Pinder, J.P. and Maruchech, A.S. "Using discounted cash flow heuristics to improve project net present value", *Journal of Operations Management*, **14**, pp. 229-240 (1996).
14. Russell, R.H. "A comparison of heuristics for scheduling projects with cash flows and resource restrictions", *Management Science*, **32**, pp. 1291-1300 (1986).
15. Shtub, A. and Etgar, R. "A branch and bound algorithm for scheduling projects to maximize net present value: The case of time dependent, contingent cash flows", *International Journal of Production Research*, **35**, pp. 3367-3378 (1997).
16. Smith-Daniels, D.E., Padman, R. and Smith-Daniels, V.L. "Heuristic scheduling of capital constrained projects", *Journal of Operations Management*, **14**, pp. 241-254 (1996).
17. Ulusoy, G. and Ozdamar, L. "A heuristic scheduling algorithm for improving the duration and net present value of a project", *International Journal of Operations and Production Management*, **15**, pp. 89-98 (1995).
18. Vanhoucke, M., Demeulemeester, E. and Herroelen, W. "On maximizing the net present value of a project under renewable resource constraints", *Management Science*, **47**, pp. 1113-1121 (2001a).
19. Vanhoucke, M., Demeulemeester, E. and Herroelen, W. "Scheduling projects with linearly time-dependent cash flows to maximize the net present value", *International Journal of Production Research*, **39**, pp. 3159-3181 (2001b).
20. Vanhoucke, M. "Exact algorithms for various types of project scheduling problems-nonregular objectives and time/cost trade-offs", Unpublished Ph.D. Dissertation, Katholieke Universiteit Leuven (2001).
21. Vanhoucke, M., Demeulemeester, E. and Herroelen, W. "An exact procedure for the resource constrained weighted earliness-tardiness project scheduling problem", *Annals of Operations Research*, **102**, pp. 179-196 (2000a).
22. Vanhoucke, M., Demeulemeester, E. and Herroelen, W. "Maximizing the net present value of a project with progress payments", *Research Report 0028*, Department of Applied Economics, Katholieke Universiteit Leuven (2000b).
23. Kazaz, B. and Sepil, C.B. "Project scheduling with discounted cash flows and progress payments", *Journal of the Operational Research Society*, **47**, pp. 1262-1272 (1996).
24. Sepil, C. and Ortac, N. "Performance of the heuristic procedures for constrained projects with progress payments", *Journal of the Operational Research Society*, **48**, pp. 1123-1130 (1997).
25. Elmaghraby, S.E. and Kamburowski, J. "The analysis of activity networks under generalized precedence relations", *Management Science*, **38**, pp. 1245-1263 (1992).
26. Bartusch, M., Mohring, R.H. and Rademacher, F.J. "Scheduling project networks with resource constraints and time windows", *Annals of Operations Research*, **16**, pp. 201-240 (1988).
27. Lawler, E.L., *Combinatorial Optimization: Networks and Matroids*, Holt, Rinehart State & Winston, New York (1976).

APPENDIX

Proof of Lemma

Lemma

The shifting strategy, which consists of shifting all in movement path activities, will lead to the complete enumeration of the search tree.

Proof

At any level P of the B&B tree, to resolve the time conflict of activities X and Y with length k, the ith branch is created by shifting activity X, i periods to the left and activity Y, $k-i$ periods to the right, as shown in Figure A1.

Let activities a, b and c exist, such that $a \prec b$, $b \prec c$ and $l_{ab} = d_a$, $l_{bc} = d_b$. Also, suppose that the length of the time conflict between activities a and b is k_1, and is k_2 for activities b and c. Let t_a and t'_a, denote the start time of activity a in the

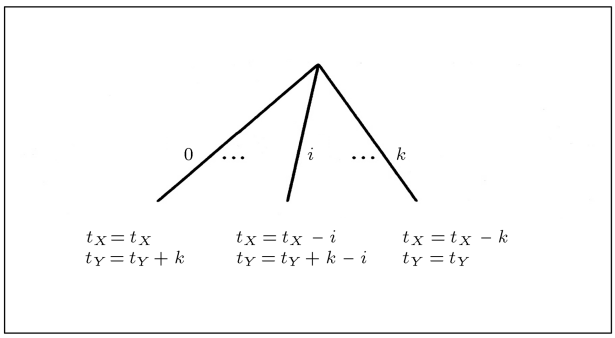

Figure A1. Branching strategy.

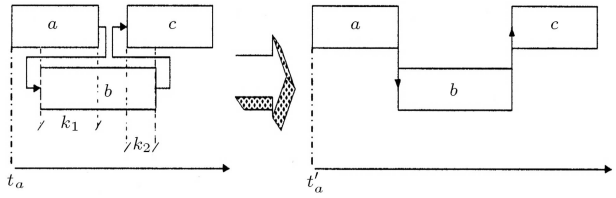

Figure A2. Current and final schedule.

current infeasible schedule and any feasible schedule, respectively, as shown in Figure A2.

It is obvious that, in the best case, at the first branch of level P, we have $t'_a = t_a$, $t'_b = t_b + k_1$ and $t'_c = t_c + k_1 + k_2$. Also, in the worst cast, at the last branch of level P', we have $t'_a = t_a - (k_1 + k_2)$, $t'_b = t_b - k_2$ and $t'_c = t_c$. Thus, we should show that the start time of activity a, t'_a, could get any value from the interval $[t_a - (k_1 + k_2), t_a]$, i.e., it must be shown that our branching strategy would count all points of this interval. Although, at these levels of the B&B tree, we consider only this interval for t'_a, which results in the time feasibility of activities a, b and c, nevertheless, there is an opportunity for t'_a, to get values beyond this interval at the next levels of the B&B tree. It is clear that:

$$[t_a - (k_1 + k_2), t_a] = [t_a - (k_1 + k_2), t_a - k_1] \cup (t_a - k_1, t_a],$$

$$\Phi = [t_a - (k_1 + k_2), t_a - k_1] \cap (t_a - k_1, t_a].$$

Suppose that the time conflict of activities a and b is resolved at level P of the B&B tree and the time conflict of activities b and c is resolved at level P'. Let $t'_a \in (t_a - k_1, t_a]$. The activity, a, starts at t'_a in the $(t_a - t'_a)$th branch at level P and, according to our branching strategy, the time conflict of b and c is also resolved automatically.

Let $t'_a \in [t_a - (k_1 + k_2), t_a - k_1]$. At branch k_1 of level P, the time conflict of activities a and b is resolved by shifting a, k_1 periods to the left, therefore, the time conflict of activities b and c is not resolved. The $(t_a - k_1 - t'_a)$th branch of level P' would result in starting activity a at period t'_a and resolving the time conflict of activities b and c. Consequently, repeating this branching strategy throughout the search tree leads to the enumeration of all possible solutions.