Journal of Applied Sciences

ISSN 1812-5654

science alert

ANSInet
an open access publisher
http://ansinet.com

Journal of Applied Sciences 8 (9): 1651-1659, 2008
ISSN 1812-5654

An Algorithm for the Weighted Earliness-Tardiness Unconstrained Project Scheduling Problem

Behrouz Afshar Nadjafi and Shahram Shadrokh
Department of Industrial Engineering, Sharif University of Technology,
P.O. Box 11365-9414, Tehran, Iran

Abstract: This research considers a project scheduling problem with the object of minimizing weighted earliness-tardiness penalty costs, taking into account a deadline for the project and precedence relations among the activities. An exact recursive method has been proposed for solving the basic form of this problem. We present a new depth-first branch and bound algorithm for extended form of the problem, which time value of money is taken into account by discounting the cash flows. 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.

Key words: Project scheduling, net present value, branch and bound

INTRODUCTION

Project Scheduling (PS) is a central field within operations research and management science. A deterministic project consists of activities, subject to precedence relations, that have a predetermined objective. Motivated by real-world situations, a wide variety of objectives for project scheduling have been studied in the literature. One of the most common objectives is related to financial measures of a project. Financial aspects of the problem appear when, generally, a series of cash flows occur over the course of the project. Time value of money is taken into account by discounting the cash flows.

Since the introduction of cash flows in project scheduling by Russell (1986) the maximization of the Net Present Value (NPV) has gained increasing attention throughout the literature. Russell (1970) considers unconstrained project scheduling problem with positive and negative cash flows and presents a non-linear programming model. Elmaghraby and Herroelen (1990) offer an optimal algorithm that builds tree structures in as AOA network. Etgar *et al.* (1996) consider a problem with an AOA network, where cash flows are associated with events. Shtub and Etgar (1997) also present a branch and bound approach to it. Etgar and Shtub (1999) again consider special version of this problem, assuming that cash flows are linear functions of the event's realization times. Vanhoucke *et al.* (2001b) study the unconstrained max-npv problem with a fixed deadline. Adding resource constraints to the max-npv problem, results in an NP hard optimization problem. Some recent surveys on Resource-Constrained Project Scheduling Problem (RCPSP) with discounted cash flows (RCPSPDC) are given by Vanhoucke *et al.* (2001a), Icmeli and Erenguc (1996) and Ulusoy and Ozdamar (1995). Doersch and Patterson (1977) developed an exact zero-one integer programming model for the RCPSPDC. Yang *et al.* (1992) propose a branch and bound algorithm for this problem. Baroum and Patterson (1999) consider an AON network with non-negative cash flows associated with the activities and proposed a branch and bound procedure for it. Zhu and Padman (1996) propose a tabu search approach. There are plenty of papers concerning heuristic approaches to the RCPSPDC. The first one was developed by Russell (1970). Some recent surveys on RCPSPDC are given by Padman and Smith-Daniels (1993), Padman *et al.* (1997), Smith-Daniels *et al.* (1996) and Icmeli and Erenguc (1994). Yang *et al.* (1995) propose nine stochastic scheduling rules. Baroum and Patterson (1996) describe several priority rule heuristics and compare them on the basis of computational experiments. Pinder and Maruchech (1996) develop 10 new scheduling heuristics and compare them with several well-known rules.

A nonregular performance measure, which is gaining attention in just-in-time environments, is the minimization of the weighted earliness-tardiness penalty costs of the project activities. In this problem setting, activities have an individual activity due date with associated unit earliness and unit tardiness penalty costs. This problem involves the scheduling of project activities in order to

Corresponding Author: Behrouz Afshar Nadjafi, Department of Industrial Engineering, Sharif University of Technology, P.O. Box 11365-9414, Tehran, Iran Tel: +989125817105

minimize the weighted earliness-tardiness costs in the absence of resource constraints. According to the classification scheme of Herroelen *et al.* (1999), problem can be classified as cpm|early/tardy.

This research addresses the Weighted Earliness-Tardiness Project Scheduling Problem (WETPSP), for the extended form, which time value of money is taken into account by continuous discounting the cash flows and minimum as well as maximum time-lags between different activities may be given. The problem is faced by many firms hiring subcontractors, maintenance crews as well as research team. Costs of earliness include extra storage requirements and idle times and implicitly incur opportunity costs. Tardiness leads to costumer complaints, loss of reputation and profits, monetary penalties or goodwill damages. The literature on solution methods for the WETPSP is scant. The Weighted Earliness-Tardiness Project Scheduling Problem (WETPSP) reduces to the problem of finding a minimal cut in a transformed digraph so that the problem can be solved in polynomial time.

Vanhoucke *et al.* (2000a) and Vanhoucke (2001) have developed an exact recursive search algorithm for the basic form. The algorithm exploits the basic idea that the earliness-tardiness costs of a project can be minimized by first scheduling activities at their due date or at a later time instant if forced so by binding precedence constraints, followed by a recursive search which computes the optimal displacement for those activities for which a shift towards time zero proves to be beneficial. Vanhoucke *et al.* (2000b) 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. This is often the case when activities are subcontracted and the subcontractors are paid upon activity completions. Kazaz and Sepil (1996) solve the problem using Benders decomposition, while Sepil and Ortac (1997) developed heuristics for the problem under renewable resource constraints.

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. The project is represented by an AON network where the set of nodes, N, represents activities and the set of arcs, A, represents finish-start precedence constraints with a time-lag of zero. The activities are numbered from the dummy start activity 1 to the dummy end activity n and are topologically ordered, i.e., each successor of an activity has a larger activity number than the activity itself. The fixed duration of an activity is denoted by $d_i (1 \le i \le n)$, while h_i denotes its deterministic due date. The completion time of activity i is denoted by the nonnegative integer variable $f_i (1 \le i \le n)$.

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:

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

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 time value of money is not taken into account.

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 (Vanhoucke *et al.*, 2000a):

$$\min \sum_{i=2}^{n} (e_i E_i + t_i T_i) \qquad (1)$$

$$f_i \le f_j - d_i \qquad \forall (i, j) \in A \qquad (2)$$

$$E_i \ge h_i - f_i \qquad \forall i \in N \qquad (3)$$

$$T_i \ge f_i - h_i \qquad \forall i \in N \qquad (4)$$

$$f_1 = 0 \qquad (5)$$

$$f_i \ge 0, E_i \ge 0, T_i \ge 0 \qquad \forall i \in N \qquad (6)$$

The objective in Eq. 1 is to minimize the weighted earliness-tardiness cost of the project. The constraint set given in Eq. 2 imposes the finish-start precedence relations among activities. Equation 3 and 4 compute the earliness and tardiness of each activity. Equation 5 forces the dummy start activity to end at time zero. Equation 6 ensure that the activity finish times, earliness and tardiness of activities assume nonnegative integer values.

Extended form of WETPSP: When dealing with the NPV criterion, time value of money is taken into account by

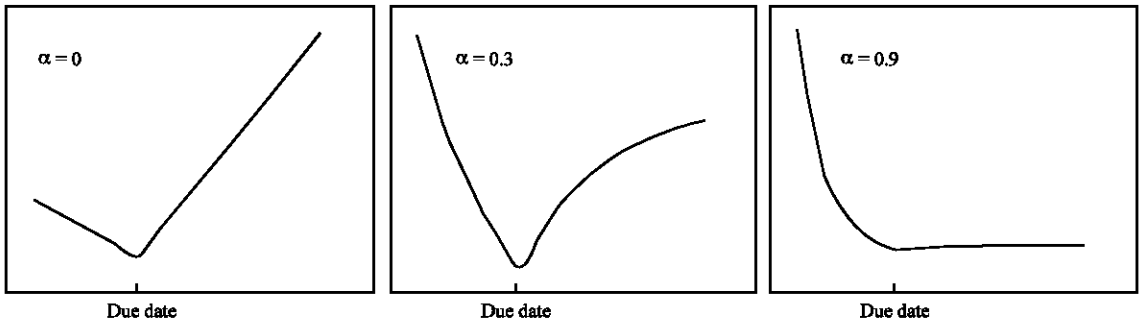

Fig. 1: Early-tardy cost curve showing the effect of discount rate on NPV

discounting the cash flows. The value of an amount of money is a function of the time of receipt or disbursement of cash. A dollar received today is more valuable than a dollar to be received in some time in the future, since the today's dollar can be invested immediately.

In order to calculate the value of NPV, a discount rate α has to be chosen, which represents the return following from investing 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. Graphical representation of the NPV of the cash flows of the activity i over time t, shows that the discount rate α is the main parameter that affects the overall shape of the curves and curve is a nonlinear function of the activity finish time and the discount rate. It is clear from the curves, which as α is increasing, NPV of tardy cost of activity tends to zero in infinity. Thus, for high discount rate α activities tend to finish in infinity. Although, a daily discount rate α greater than 0.02 corresponds to an annual discount rate of 3070% which is very unrealistic value, but for preserve of generality, we consider a deadline δ_n for the project.

The objective of the extended form of WETPSP is to find a schedule such that the net present value of the project is minimized. The way of calculating the value of NPV depends on the payment model considered. We suppose that the cost due to earliness or tardiness of each activity, will impose in progress model (e.g., at the end of each month, earliness or tardiness of each activity may impose some cost to the client).

We have the following notations for weighted earliness-tardiness project scheduling with discounted cash flows (The extended form of WETPSP):

- n = Number of activities
- N = Set of nodes of acyclic digraph representing the project
- A = Set of arcs of acyclic digraph representing the project
- d_i = Duration of activity i
- h_i = Due date of activity i
- f_i = Finish time of activity i (integer decision variable)
- EFT_i = Earliest finish time of activity i
- LFT_i = Latest finish 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
- $a \prec b$ = Denotes that activity b may start as soon as activity a is completed.

Using the above notation, the resource-unconstrained project scheduling problem under the minimum discounted early-tardy penalty cost objective (classified as problem early/tardy following the Herroelen *et al.* (1999)) can be mathematically formulated as follows:

$$\min Z = \sum_{i=2}^{n}\left(e_i \sum_{k=f_i}^{h_i-1} e^{-\alpha k} + t_i \sum_{k=h_i+1}^{f_i} e^{-\alpha k}\right) \quad (7)$$

Subject to:

$$f_i \le f_j - d_j \qquad \forall (i,j) \in A \quad (8)$$

$$f_n \le \delta_n \quad (9)$$

$$f_1 = 0 \quad (10)$$

$$f_i \ge 0 \qquad \forall i \in N \quad (11)$$

The objective in Eq. 7 minimizes the NPV of the project. The constraint set in Eq. 8 maintains the finish-start precedence relations among the activities. In order to restrict the project duration, we add a negotiated project deadline δ_n, given in Eq. 9 and 10 forces the dummy start activity to end at time zero. Equation 11 ensure that the activity finish 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 our proposed algorithm, initial scheduling procedure sets the finish time of each activity i at max (h_i, EFT_i) and calculates its cost. Let us P denotes the level of the branch and bound tree. Then in the initial schedule, we set P = 0. Although this schedule has minimum cost, but it may be infeasible. Feasibility of a schedule can be evaluated by State Matrix (SM). Each element of this matrix can be calculated as follows:

$$SM(i,j) = \begin{cases} f_j - d_j - f_i ; & (i,j) \in A \\ 0 ; & \text{Otherwise} \end{cases}$$

In this matrix, for example, if activity i be predecessor of activity j and finish time of activity i is equal to start time of activity j then we will have SM(i, j) = 0. If finish time of activity i is greater than (less than) the start time of activity j then we will have SM(i, j) < 0 (SM(i, j) > 0). If all elements of SM are nonnegative, then current schedule is time-feasible that means all precedence relations are preserved.

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 feasibility of 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 branch and bound tree, if there is more than one such node, ties are broken in favor of children who have 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 time conflict (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 (SM(i, j) < 0) and activity i is predecessor of activity j. Suppose that the duration of time conflict between these activities is l time unit. We have at most (l + 1) alternatives to resolve this time conflict. In the first node, activity j is shifted to right l time unit. In the second node, activity j is shifted to right (l – 1) time unit and activity i is shifted to left 1 time unit. In the kth node, activity j is shifted to right (l – k + 1) time unit and activity i is shifted to left (k – 1) time unit. Finally, in (l + 1)th node, activity i is shifted to left l time unit. In shifting process we must aware that, for each activity i, activity's finish time don't be later than LFT_i, or earlier than EFT_i.

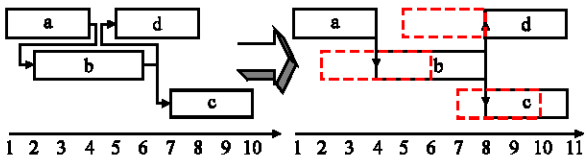

Fig. 2: Shifting process

In order to avoid of creating new conflicts, in shifting process, it is necessary to device preventive approaches. Therefore it is assumed that, in right (left) shifting of each activity i, if we faced with new conflicts, the associated activities must shifted to right (left) too. For example this can be shown graphically as given in Fig. 2, which we suppose $a \prec b$, $b \prec c$, $b \prec d$ and activity b is selected for right shift.

Consequently, the following lemma applies:

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

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

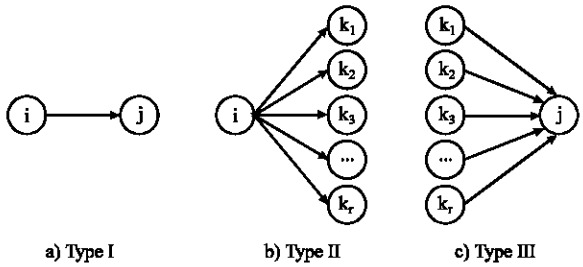

Fig. 3: Types of conflict set

is based on concept called 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 predecessor of activity j and these activities have time conflict (SM(i, j) < 0), b) activity i is predecessor of some activities k_1, k_2, k_3,…, k_r and their generalized precedence relations are ignored, or c) activity j is successor of some activities k_1, k_2, k_3,…, k_r and their generalized precedence relations are ignored. These three types of conflict sets are given in Fig. 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 maximum such conflict sets, so that they didn't have common activity. In the other hand, each activity i, may be member of one conflict set. In this stage, other generalized precedence constraints that didn't consider in none of conflict sets are relaxed. In the second stage, LB is calculated based of the following propositions:

Proposition 1: Associated with each type I conflict set, that a precedence relation between activities i and j is ignored, let C(i, j) be the minimum cost incurred for resolving time-conflict between activities i and j. If we suppose that the relative time conflict is l(i, j) time unit, then C(i, j) can be computed as follows:

$$C(i, j) = \min_{v=0}^{l(i,j)} \left\{ e_i \sum_{k=f_i-v}^{f_i-1} e^{-\alpha k} + t_i \sum_{k=f_i+1}^{f_i+l(i,j)-v} e^{-\alpha k} \right\}$$

Let $\lambda(\tau)$ be the minimum cost incurred for resolving τth conflict set in current schedule. Thus, about type I conflict set, it may be stated as:

$$\lambda(\tau) = C(i, j)$$

Proposition 2: Associated with each type II conflict set, which activity i is predecessor of some activities k_1, k_2, k_3,…, k_r and their precedence relations are ignored, we can compute as follows:

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

Proposition 3: Associated with each type III conflict set, which activity j is successor of some activities k_1, k_2, k_3,…, k_r and their precedence relations are ignored, we may derive:

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

Proposition 4: Associated with each schedule, a Lower Bound (LB) may be written as follows:

$$\text{Lower bound} = \sum_{\tau} \lambda(\tau) + Z$$

Dominance rule: The second pruning rule, the dominance rule, is based on 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* = ∞, optimal schedule = Φ;

Step 1: Generate the initial schedule, set it as current schedule;
Step 2: Create CA and SM and calculate Z and LB, for 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 current schedule (generate all schedules directly reachable from current schedule, by all possible shifting activities i and j. For resulting schedules, set P = P+1 and create their CA and SM and calculate their Z and LB);

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

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

Step 8: Incumbent rule;

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

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

Step 11: Set current schedule as optimal schedule and set Z* = Z;

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

Step 13: if Z* = ∞, print 'no possible solution', else print optimal schedule;

Step 14: End;

A NUMERICAL EXAMPLE

In this section, we determine an optimal schedule for a numerical example by means of the branch and bound procedure presented. Consider an example of the WETPSP with 14 activities borrowed from Vanhoucke *et al.* (2000b). Figure 4 shows the precedence relations among the activities and Table 1 shows the duration, due date and the unit penalty cost of the activities. For ease of representation, we assume the unit earliness costs to equal the unit tardiness costs. The project deadline δ_n amounts to 21 and the monthly discount rate α equals 0.01 (1%).

At the initial level P = 0 of the tree, we determine the initial scheduling with setting finish times of each activity i at max (hi, EFTi). An extended Gantt chart associated with the root of the enumeration tree, node 1, is displayed in Fig. 5. Such a Gantt chart shows this schedule is not feasible and there are three conflict set. State Matrix (SM) for initial scheduling is shown in Fig. 6. This matrix has some negative elements which is another representation of infeasibility. Then, the set of conflicted activities are CA = {3, 5, 6, 7, 8, 10, 11 and 12}. In the root of the enumeration tree, the algorithm computes the objective function (Z) and Lower Bound (LB), 0.87 and 15.31, respectively. Table 2 shows the detailed computation of lower bound in node 1.

First negative element in SM corresponds to activities 3 and 5, which conflict 1 time unit. Consequently, the algorithm continues with step 5. The level of the branch and bound tree is increased, i.e., P = 1

Table 1: Project's data for numerical example

	Activity													
	1	2	3	4	5	6	7	8	9	10	11	12	13	14
d_i	0	6	5	3	1	6	2	1	4	3	2	3	5	0
h_i	0	7	6	5	0	13	9	8	11	12	9	13	20	21
$e_i = t_i$	∞	4	3	6	0	7	7	7	8	7	4	1	6	∞

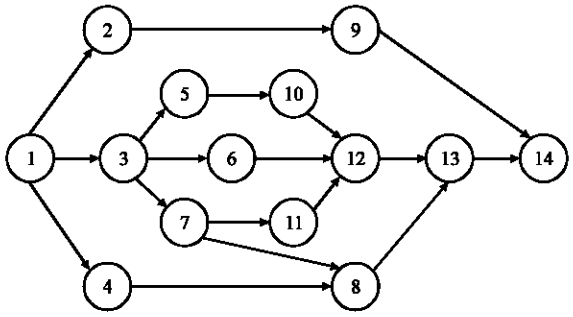

Fig. 4: Precedence relations

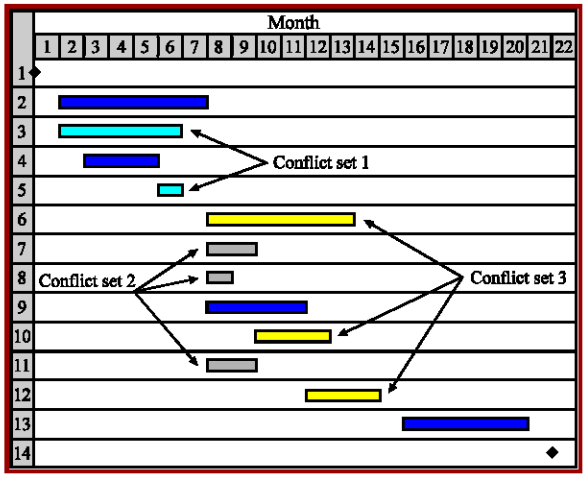

Fig. 5: Extended Gantt chart

$$\begin{bmatrix} 1 & 1 & 2 & & & & & & & & & & & \\ & & & 0 & & & & & & & & & & \\ & & -1 & 1 & 1 & & & & & & & & & \\ & & & & 2 & & & & & & & & & \\ & & & & & 3 & & & & & & & & \\ & & & & & & & & -2 & & & & & \\ & & & & -2 & & & -2 & & & & & & \\ & & & & & & & & & 7 & & & & \\ & & & & & & & & & & 10 & & & \\ & & & & & & & & -1 & & & & & \\ & & & & & & & & 2 & & & & & \\ & & & & & & & & & 1 & & & & \\ & & & & & & & & & & 1 & & & \end{bmatrix}$$

Fig. 6: SM for initial scheduling

Table 2: Detailed computation of lower bound in node 1

Conflict set No (τ)	Type	Members	C(i, j)	$\hat{\lambda}(\tau)$	Lower bound (LB)
1	I	3 and 5	0.00	0.00	12.73+1.71+0.87 = 15.31
2	II	7 and 8	12.73	12.73	
		7 and 11	7.20		
3	III	6 and 12	1.71	1.71	
		10 and 12	0.86		

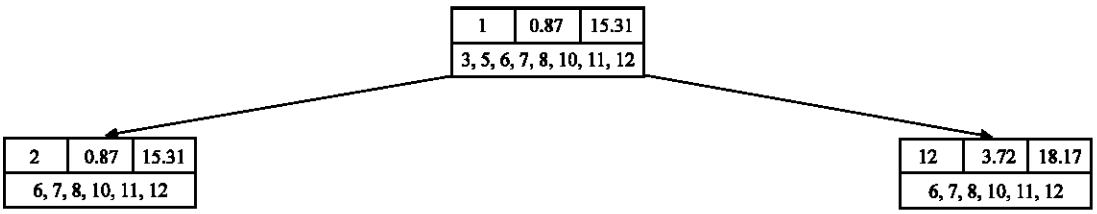

Fig. 7: Branching process in initial schedule

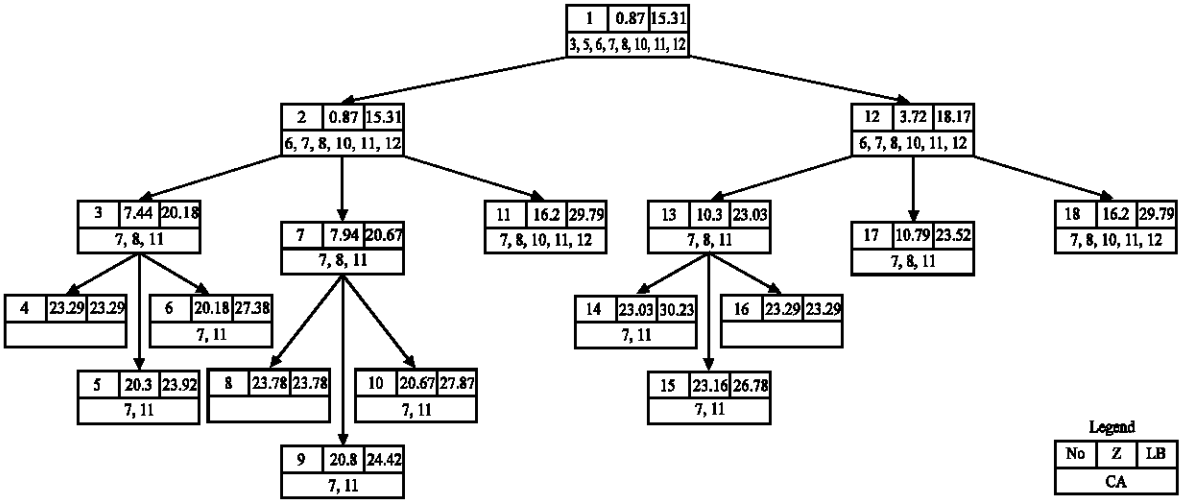

Fig. 8: Branch and bound tree

and two descendant nodes will be generated. Node 2, which is due to right shift of activity 5 and node 3, which is due to left shift of activity 3.

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) is updated for each node and a lower bound on the NPV is computed. Resulting tree is given in Fig. 7.

We have coded the branch and bound procedure in MATLAB version 6.5 under windows XP. The complete branch and bound tree for the example is given in Fig. 8.

In node 4 a first feasible schedule is found with a NPV of 23.29, corresponding with feasible finish times (0, 7, 5, 5, 7, 13, 7, 8, 11, 12, 9, 16, 21 and 21). Search in all tree shows that this schedule is optimal, too. This optimal schedule is repeated in node (16). A feasible schedule is found in node 8, with a NPV of 23.78. Other nodes are fathomed because of the two pruning rules that were described previously.

COMPUTATIONAL RESULTS

In order to validate the proposed branch and bound method for the WETPSPDC, a problem set consisting of 120 problem instances was generated. This problem set consisting of equally 40 instances with 10, 30 and 50 activities. The problem set was extended with unit earliness-tardiness penalty costs for each activity which are randomly generated between 1 and 10. The due dates were generated in the same way as described by Vanhoucke *et al*. (2000b). First, a maximum due date was obtained for each project by multiplying the critical path length by 1.5. Subsequently, we generate random

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

No. of activities	No. of problems	Average CPU-time
10	40	0.188
30	40	0.912
50	40	1.897

numbers between 1 and maximum due date. The numbers are sorted and assigned to the activities in increasing order. Activity durations are randomly selected between 1 and 10. Maximum number of predecessors and successors supposed 3.

We have coded the branch and bound procedure in MATLAB version 6.5. The problem set has solved under windows XP on a personal computer with Pentium IV, 1.7 GHz processor. Table 3 represents the average CPU-time in second for a different number of activities. As can be seen from Table 3, the result shows that computational requirements are small.

CONCLUSION

This research reports on an exact branch and bound procedure for extended form of problem cpm|early/tardy, i.e., the unconstrained project scheduling problem with continuous discounted negative cash flows. Negative cash flows occur when an activity is completed at prior or later than its due date. The objective is to schedule the activities in order to minimize the NPV subject to the precedence constraints and a fixed deadline on project. Branching is done by right and left shifting process on two activities which have time conflict, so that predecessor activity has smallest number. Two rules are used for node fathoming, incumbent rule and dominance rule. First pruning procedure computes lower bound by making disjunctive subset of activities so called conflict set. Second pruning procedure fathomed previously examined node in another path of the search tree. Finally, the new branch and bound procedure used for solving a numerical example.

REFERENCES

Baroum, S.M. and J.H. Patterson, 1996. The development of cash flow weighted procedures for maximizing the net present value of a project. J. Operat. Manage., 14 (3): 209-227.

Baroum, S.M. and J.H. Patterson, 1999. An Exact Solution Procedure for Maximizing the Net Present Value of Cash Flows in a Network, Chapter 5. In: Handbook on Recent Advances in Project Scheduling, Weglarz, J. (Ed.). Kluwer Academic Publishers, Dordrecht, pp: 107-134.

Doersch, R.H. and J.H. Patterson, 1977. Scheduling a project to maximize its present value: A zero-one programming approach. Manage. Sci., 23 (8): 882-889.

Elmaghraby, S.E. and W. Herroelen, 1990. The scheduling of activities to maximize the net present value of projects. Eur. J. Operat. Res., 49 (1): 35-49.

Etgar, R., A. Shtub and L.J. LeBlanc, 1996. Scheduling projects to maximize net present value-The case of time independent, contingent cash flows. Eur. J. Operat. Res., 96 (1): 90-96.

Etgar, R. and A. Shtub, 1999. Scheduling projects activities to maximize net present value-The case of linear time dependent, contingent cash flows. Int. J. Prod. Res., 37 (2): 329-339.

Herroelen, W., B. De Reyck and E. Demeulemeester, 1999. A Classification Scheme for Project Scheduling Problems, Chapter 1. In: Handbook of Recent Advances in Project Scheduling, Weglarz, J. (Ed.). Kluwer Academic Publishers, Dordrecht, pp: 1-26.

Icmeli, O. and S.S. Erenguc, 1994. A tabu search procedure for resource constrained project scheduling with discounted cash flows. Comput. Operat. Res., 21 (8): 841-853.

Icmeli, O. and S.S. Erenguc, 1996. A branch and bound procedure for resource constrained project scheduling problem with discounted cash flows. Manage. Sci., 42 (10): 1395-1408.

Kazaz, B. and C.B. Sepil, 1996. Project scheduling with discounted cash flows and progress payments. J. Operat. Res. Soc., 47 (10): 1262-1272.

Padman, R. and D.E. Smith-Daniels, 1993. Early-tardy cost trade-offs in resource constrained projects with cash flows: An optimization-guided heuristic approach. Eur. J. Operat. Res., 64 (2): 295-311.

Padman, R., D.E. Smith-Daniels and V.L. Smith-Daniels, 1997. Heuristic scheduling of resource-constrained projects with cash flows. Nav. Res. Logist, 44 (4): 365-381.

Pinder, J.P. and A.S. Maruchech, 1996. Using discounted cash flow heuristics to improve project net present value. J. Operat. Manage., 14 (3): 229-240.

Russell, A.H., 1970. Cash flows in networks. Manage. Sci., 16 (5): 357-373.

Russell, R.H., 1986. A comparison of heuristics for scheduling projects with cash flows and resource restrictions. Manage. Sci., 32 (10): 1291-1300.

Sepil, C. and N. Ortac, 1997. Performance of the heuristic procedures for constrained projects with progress payments. J. Operat. Res. Soc., 48 (11): 1123-1130.

Shtub, A. and R. Etgar, 1997. A branch and bound algorithm for scheduling projects to maximize net present value: The case of time dependent, contingent cash flows. Int. J. Prod. Res., 35 (12): 3367-3378.

Smith-Daniels, D.E., R. Padman and V.L. Smith-Daniels, 1996. Heuristic scheduling of capital constrained projects. J. Operat. Manage., 14 (3): 241-254.

Ulusoy, G. and L. Ozdamar, 1995. A heuristic scheduling algorithm for improving the duration and net present value of a project. Int. J. Operat. Prod. Manage., 15 (1): 89-98.

Vanhoucke, M., E. Demeulemeester and W. Herroelen, 2000a. An exact procedure for the resource constrained weighted earliness-tardiness project scheduling problem. Ann. Operat. Res., 102 (1-4): 179-196.

Vanhoucke, M., E. Demeulemeester and W. Herroelen, 2000b. Maximizing the net present value of a project with progress payments. Research Report 0028. Department of Applied Economics, Katholieke Universiteit Leuven.

Vanhoucke, M., 2001. Exact algorithms for various types of project scheduling problems_Nonregular objectives and time/cost trade-offs. Unpublished Ph.D Thesis, Katholieke Universiteit Leuven.

Vanhoucke, M., E. Demeulemeester and W. Herroelen, 2001a. On maximizing the net present value of a project under renewable resource constraints. Manage. Sci., 47 (8): 1113-1121.

Vanhoucke, M., E. Demeulemeester and W. Herroelen, 2001b. Scheduling projects with linearly time-dependent cash flows to maximize the net present value. Int. J. Prod. Res., 39 (14): 3159-3181.

Yang, K.K., F.B. Talbot and J.H. Patterson, 1992. Scheduling a project to maximize its net present value: An integer programming approach. Eur. J. Operat. Res., 64 (2): 188-198.

Yang, K.K., L.C. Tay and C.C. Sum, 1995. A comparison of stochastic scheduling rules for maximizing project net present value. Eur. J. Operat. Res., 85 (2): 327-339.

Zhu, D. and R. Padman, 1996. A tabu search for scheduling resource-constrained projects with cash flows. Working Paper: 96-30, Carnegie-Mellon University.