**Vlerick Working Papers 2002/19**

# OPTIMAL DUE DATE ASSIGNMENT IN PROJECT SCHEDULING

MARIO VANHOUCKE

e-mail: mario.vanhoucke@vlerick.be

D/2002/6482/20

# **PROJECT SCHEDULING**

MARIO VANHOUCKE

e-mail: mario.vanhoucke@rug.ac.be

Department of Management Information, Operations Management and Technology Policy

Faculty of Economics and Business Administration, Ghent University

Hoveniersberg 24, B-9000 Gent (Belgium)

e-mail: mario.vanhoucke@vlerick.be

Operations and Technology Management Center

Vlerick Leuven Gent Management School

Bellevue 6, B-9050 Gent (Belgium)

## ABSTRACT

In this paper we introduce the concept of due date assignment in the project scheduling literature. Despite the fact that due date assignment problems belongs to the core of the machine scheduling literature, no attempts have been made to tackle this problem in a project scheduling environment. However, of obvious practical importance, an optimal assignment of due dates is of primary interest to the project manager.

In a recent research paper on project scheduling with due dates, the problem has been restricted to considering projects with pre-assigned due dates. In reality, due dates are the results of negotiations, rather than simply dictated by the client of the project. In this paper we consider this negotiation process and take a contractor's point of view who faces the problem of assigning due dates to a particular project, based on the negotiation arguments of the client. We show that the problem under study can be solved by means of the combination of different ideas from the operations research community.

**Keywords:** *Project Management; Due date assignment; Weighted earliness-tardiness costs, Due dates;*

## 1. INTRODUCTION

Over the past decades, a lot of research attention has been devoted to both machine and project scheduling problems. Despite the fact that the problem types have a lot in common, little effort has been done to incorporate elements from one field to the other. In this paper, we elaborate on the growing research interest in the just-in-time (*JIT*) philosophy, in which costs are incurred in schedules when activities finish earlier or later than planned. Although the research efforts in the machine scheduling literature are numerous, little transfer of ideas has been made to the project scheduling community.

An overview and state-of-the-art survey of due date assignment and machine scheduling can be found in Gordon et al. (1998, 2002). In this paper, the authors focus on common due date assignment. While a large body of the research focuses on these common due dates, distinct due dates are also the subject to intensive research. A detailed description of the excellent papers and numerous books of machine scheduling problems with due dates is however outside the scope of this paper. References can be found in the paper by Gordon et al. (2002).

Scheduling problems with due dates have been introduced only recently in the project scheduling literature. Vanhoucke et al. (1999) have introduced the problem of scheduling project activities as close as possible to pre-specified due dates, within the given precedence constraints. Once the finishing time $f_i$ of an activity *i* deviates from the pre-assigned due date $d_i$, an earliness ($f_i < d_i$) or tardiness ($f_i > d_i$) cost has to be paid. The optimal schedule reports the finishing times of all activities, such that the total weighted earliness-tardiness cost is minimized. The problem is referred to as the ***w**eighted **e**arliness-**t**ardiness **p**roject **s**cheduling **p**roblem* (*WETPSP*). The obvious extension to renewable resource constraints has been investigated by the same authors (see Vanhoucke et al. (2000)). This ***r**esource-**c**onstrained **p**roject **s**cheduling **p**roblem with **w**eighted **e**arliness-**t**ardiness costs* (*RCPSPWET*) has been solved by a branch-and-bound procedure in which the unconstrained *WETPSP* is used for the lower bound calculations. In both papers, the problem type has been restricted to considering projects with pre-assigned due dates.

The problem under study stems from the observation that due dates are the result of negotiations, rather than dictated by the client or customer of the project. Indeed, instead of assigning due date arbitrarily to the project activities, the contractor has to tune the different possibilities, wishes and proposals of the different parties. The contractor is responsible for scheduling the project activities as close as possible to their due dates and risks to pay a penalty cost for possible deviations. Therefore, an optimal assignment of due dates is of primary interest to the contractor of the project. Moreover, the contractor's bid must include target dates, which are promised due dates towards the client of the project. The combination of assigning due dates and scheduling the project has led to the algorithm presented in this article.

In this paper we present a double branch-and-bound algorithm to minimize the weighted earliness-tardiness penalty costs in project networks subject to zero-lag finish-start precedence constraints, multiple due date possibilities and renewable resource constraints (subsequently denoted as the *RCPSPMDD,* i.e. the ***r**esource-**c**onstrained **p**roject **s**cheduling **p**roblem with **m**ultiple **d**ue **d**ates*). The *RCPSPMDD* extends the NP-hard resource-constrained project scheduling problem with weighted earliness-tardiness costs (problem *m*,1|*cpm*|*early/tardy* or the *RCPSPWET*) to multiple due date possibilities. These multiple due date possibilities are the result of negotiations and contain the different possibilities that the contractor can choose from. Consequently, the problem under study consists of a combination of due date assignment and project scheduling. To the best of our knowledge, no exact procedure has yet been suggested for its solution.

The organization of the paper is as follows. In section 2 we present the features of the problem in detail. Section 3 describes the logic of the double branch-and-bound algorithm. We refer to a double branch-and-bound procedure since the lower bound calculation used in the main branch-and-bound algorithm also relies on a branch-and-bound logic. In section 4 we report detailed computational results on a randomly generated problem set. In section 5 we give our overall conclusions.

## 2. DESCRIPTION OF THE PROBLEM

### 2.1. Problem characteristics

In this paper we do not assume that activity due dates are pre-assigned but instead negotiated rather than simply dictated by the cleints. Indeed, the contractor who is in charge of the project has to assign due dates and propose a schedule to the client of the project. Therefore, we advocate that due dates, including earliness and tardiness penalty costs for possible deviations, are agreed upon by the client, the contractor and possibly some subcontractors. Indeed, when the contractor makes use of subcontracting of certain activities, he receives certain proposals with respect to possible execution periods. During the negotiation process, the contractor transfers these proposals as possible due dates to the client. The client, on its turn, is not indifferent between the different due date scenarios. Instead, we assume that the client associates a certain fixed cost to due dates later in time. Moreover, we also assume that penalty costs are associated with due date deviations, which increase as the due date increases in time. Indeed, a due date overrun is considered more severe when this due date is scheduled further in time. An overrun for a very tight due date, however, can be tolerated by the client in a much easier way. As a summary, we conclude that both earliness and tardiness costs are set by the contractor or are a result of the contractor/subcontractor negotiations. Therefore, we assume that these penalty costs are relatively constant over time (almost equal between the different due date possibilities). On top of that, extra tardiness penalty costs, however, are a result of the client/contractor negotiations and are positively correlated with the promised due date, i.e. the later the due date, the larger the penalty cost of an overrun. The overall effect is that earliness costs are relatively constant over time while the tardiness costs is an increasing function of the due dates. Typical reasons for earliness costs include extra storage requirements and idle times and implicitly incur opportunity costs. Tardiness leads to cleint complaints, loss of reputations and profits, monetary penalties or goodwill damages.

Note that Ulusoy and Cebelli (2000) have developed a double-loop genetic algorithm in order to find an equitable solution for the multi-mode, resource-constrained payment scheduling problem. They define an equitable solution as a solution where both the contractor and the client deviate from their respective ideal solutions by an equal percentage. Although their problem is fundamentally different than the problem under study, they have given the first signal to investigate the negotiation process between different parties involved in the project.

Throughout this paper, we assume that a project is represented by an activity-on-the-node (AoN) network where the set of activity 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. Each activity has a duration $d_i$ ($1 \leq i \leq n$) and a number of due date possibilities $nr(i)$. The $l^{\text{th}}$ due date of activity $i$ is denoted by $h_{il}$ ($1 \leq i \leq n$ and $1 \leq l \leq nr(i)$). Note that $nr(0) = nr(n) = 1$ and $h_{11} = 0$, since nodes 0 and $n$ are dummy activities. The due date of the dummy end activity $h_{n1}$ equals the negotiated project deadline. The completion time of activity $i$ is denoted by the nonnegative integer variable $f_i$ ($1 \leq i \leq n$).

The earliness of activity $i$ can be computed as $E_i = \max(0, h_{il} - f_i)$ and its tardiness as $T_i = \max(0, f_i - h_{il})$. As one can seen, the calculation of these penalty costs depend on the assignment of the due date $h_{il}$ to this activity. To that purpose, we need to introduce a binary decision variable in our conceptual model which determines the assignment or selection of a specific due date for an activity $i$,

$$y_{il} = \begin{cases} 1, \text{if due date } h_{il} \text{ has been assigned to activity } i \\ 0, \text{otherwise} \end{cases}.$$

We use $b_{il}$ to denote a fixed cost associated with each due date $l$ of activity $i$. As explained previously, we assume that $b_{i1} < b_{i2} < \ldots < b_{i,nr(i)}$, i.e. the larger the due date, the larger the fixed cost. When $e_{il}$ and $t_{il}$ respectively denote the per unit earliness and tardiness penalty cost of activity $i$ for due date $h_{il}$, its total earliness-tardiness cost is $\sum_{l=1}^{nr(i)} (e_{il} E_i + t_{il} T_i) y_{il}$ . There are $K$ renewable resources with $a_k$ ($1 \leq k \leq K$) as the availability of resource type $k$ and with $r_{ik}$ ($1 \leq i \leq n$, $1 \leq k \leq K$) as the resource requirements of activity $i$ with respect to resource type $k$. The *RCPSPMDD* can be conceptually formulated as follows:

$$\text{Minimize} \sum_{\text{i}=1}^{n} \sum_{l=1}^{nr(i)} (b_{il} + e_{il} E_i + t_{il} T_i) y_{il}$$ [1]

Subject to

$$f_i \leq f_j - d_j \qquad \forall (i, j) \in A$$ [2]

$$E_i \geq \sum_{l=1}^{nr(i)} h_{il} y_{il} - f_i \qquad i = 1, \ldots, n$$ [3]

$$T_i \geq f_i - \sum_{l=1}^{nr(i)} h_{il} y_{il} \qquad i = 1, \ldots, n$$ [4]

$$\sum_{i \in S(t)} r_{ik} \leq a_k \qquad k = 1, \ldots, K \text{ and } t = 1, \ldots, T$$ [5]

$$\sum_{l=1}^{nr(i)} y_{il} = 1 \qquad i = 1, \ldots, n$$ [6]

$$f_1 = 0$$ [7]

$$f_i \in \text{int}^+, E_i \in \text{int}^+, T_i \in \text{int}^+ \qquad i = 1, \ldots, n$$ [8]

$$y_{il} \in \text{bin} \qquad i = 1, \ldots, n \text{ and } l = 1, \ldots, nr(i)$$ [9]

where *S(t)* denotes the set of activities in progress in period ]$t$-1,$t$]. The objective in Eq. 1 minimizes the total cost of the project (i.e. the fixed cost for each due date and the corresponding weighted earliness-tardiness cost). The constraint set given in Eq. 2 maintains the finish-start precedence relations among the activities. The constraint sets in Eq. 3 and Eq. 4 compute the earliness and tardiness of each activity and Eq. 5 represents the renewable resource constraints. Eq. 6 represents the due date assignment and forces to have a single due date for each activity. Eq. 7 forces the dummy start activity to end at time zero and Eq. 8 ensures that the activity finishing times as well as the earliness and tardiness assume nonnegative integer values. Eq. 9 ensures that the due date assignment variable is a binary (0/1) variable.

## 2.2. Due date scenarios

In this section, we take a closer look to the features and characteristics of the due date possibilities and associated penalty costs. Throughout this paper we assume that every individual activity has several due date possibilities with associated earliness and tardiness costs. Of course, in reality, only very important activities (so-called milestones) that are crucial to the execution of the project will have a due date. However, our assumption does not harm to the real-life value of the problem. Instead, the less activities with a due date, the easier to solve the problem.

Due to the specific characteristics of the due date scenarios and corresponding penalty costs, the contractor is confronted with a trade-off between due date assignment and earliness/tardiness cost. On the one hand, assigning an early due date of a certain activity implies that the contractor has not much freedom to schedule. Indeed, due to the limited availability of the resources, many activities will deviate from the promised due date, resulting in earliness or tardiness costs. A late due date, on the other hand, results in a higher probability of finishing on time (due date). Deviations (especially overruns), although occurring less frequently, are penalized in a much stronger way, possibly resulting in a high cost of tardiness. Another drawback is a longer total project duration, which can also be the subject of a severe penalty cost. As a summary, the trade-off basically implies the choice between many due date deviations with each a small penalty cost or a few due date deviations with a severe penalty per day of due date deviation.

Consider the following example for activity $i$ with three possible due dates as proposed by the contractor. The due dates with the corresponding costs agreed with the client are $h_{i1} = 2$, $h_{i2} = 6$ and $h_{i3} = 8$ with $b_{i1} = 0$, $b_{i2} = 8$ and $b_{i3} = 12$. The associated penalty costs for earliness and tardiness are given by $e_{i1} = e_{i2} = e_{i3} = 6$ and $t_{i1} = 10$, $t_{i2} = 13$ and $t_{i3} = 20$. In figure 1, we illustrate the total penalty cost profile (denoted by $E/T$) for each possible due date.

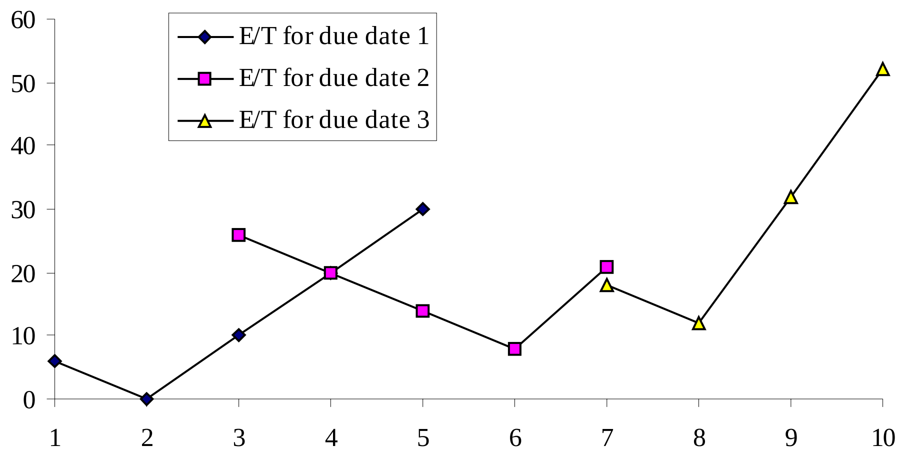


**FIGURE 1. Earliness-tardiness (e/t) profile for different due dates**

Note that certain assumptions can be relaxed without changing the global philosophy of the algorithm. Firstly, we can make the penalty costs nonlinear. Instead of assigning a linear penalty cost for due date deviations we can propose a quadratic penalty function, i.e. the larger the deviation, the higher the increase in penalty cost. We only use linear penalty functions for the sake of simplicity in explaining our algorithm. Second, we can easily extend the problem by ready times or fixed due dates for which no deviation is allowed. To that purpose, we simply set the penalty cost to a very high value. Ready times can also be incorporated by means of extra arcs between the dummy start node and the activity but we prefer to use a high earliness cost. In doing so, the problem is less restrictive than incorporating a 'hard' precedence constraint.

## 3. THE ALGORITHM

In this section we describe a double branch-and-bound algorithm for the *RCPSPMDD*. The first branch-and-bound algorithm ($BB_1$) serves to calculate the lower bounds $lb_2$ for the *RCPSPMDD* and is related with the procedure of Vanhoucke et al. (2002) (this problem without resources is called the ***p****roject* ***s****cheduling* ***p****roblem with* ***m****ultiple* ***d****ue* ***d****ates*, or the *PSPMDD*). The second, main branch-and-bound procedure ($BB_2$) takes the resolving of resources conflicts into account and is similar to the branch-and-bound algorithm for solving the *RCPSPWET* (see Vanhoucke et al. (2000)). Consequently, each node of the tree of $BB_2$ consists of a lower bound $lb_2$ as given by solving the branch-and-bound approach $BB_1$ (see table 1).

**TABLE 1. Overview of the double branch-and-bound scheme for the rcpspmdd**

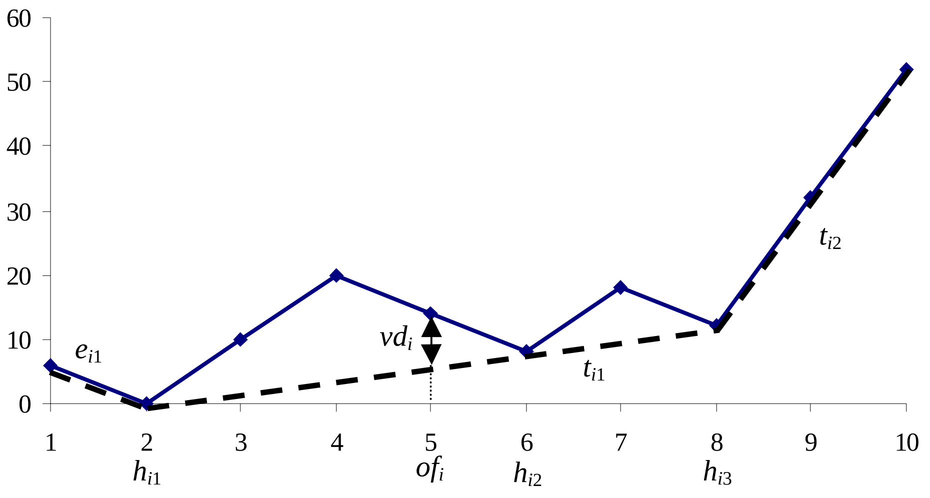
| **Problem type** | **B&B algorithm** | **Lower bound at each node** | **Solution** |
|---|---|---|---|
| Unconstrained *PSPMDD* | $BB_1$ | $lb_1$ | $lb_2$ |
| *RCPSPMDD* | $BB_2$ | $lb_2$ | Optimal resource profile with minimal $E/T$ cost |

## 3.1. Calculation of $lb_1$ by means of a recursive search algorithm

During the calculation of the lower bound for the *RCPSPMDD* we solve the problem described in section 2.1 without the resource constraints [5]. To that purpose, we have developed a branch-and-bound algorithm $BB_1$ for which we use an adapted recursive search procedure of Vanhoucke et al. (1999) for solving the *WETPSP* to calculate the lower bounds $lb_1$ at each node of the tree of $BB_1$. This is the topic of this subsection.

In figure 2 we illustrate the earliness-tardiness profile for each possible due date $h_{il}$. The contractor has to assign due dates to the project and schedule all the activities such that the total penalty cost is minimized. Therefore, the relevant cost function is given by the minimal penalty cost functions over all possible due dates. This is illustrated in a bold straight line in figure 2, which consists of the minimal penalty costs for each time period taking all the relevant due dates into account. For example, there is no need to consider a tardiness penalty cost for due date $h_{i1}$ at time period 5 since then, it is more advantageous to assign due date $h_{i2}$ with the corresponding earliness cost.

**FIGURE 2. The earliness-tardiness cost profile and the corresponding convex envelope**



In order to solve the problem without resource constraints (i.e. a lower bound $lb_2$), we rely on a branch-and-bound procedure $BB_1$. At each node, we calculate a lower bound $lb_1$ for the total earliness-tardiness cost. To that purpose, we construct the *convex envelope* of the total *E/T* cost profile over the whole time window [$ef_i$, $lf_i$] for each activity *i*. The convex envelope of a function *F* taken over *C* = [$ef_i$, $lf_i$] is defined as the highest convex function which fits below *F*. This is illustrated in a dotted line in figure 2. As a result, the problem has been transformed into a *WETPSP* with a due date $h_{i1}$ and corresponding penalty costs for earliness ($e_{i1}$) and tardiness ($t_{i1}$ and $t_{i2}$).

The procedure to solve the *WETPSP* has been described in Vanhoucke et al. (1999) and basically runs as follows. The algorithm starts with an initial schedule for which all activities are scheduled at their due date or later. In doing so, the total weighted earliness-tardiness cost of the project can only be improved by shifting sets of activities towards the beginning of the project (towards time zero). A recursive search in a so-called due date tree detects these sets of activities for which such a shift is beneficial, i.e. that results in a lower earliness-tardiness cost. The algorithm continues that search until no further shift is beneficial and reports the optimal finishing times of the activities.

Optimal finishing times $of_i$ for the *WETPSP* problem with due date $h_{i1}$ and penalty costs $e_{i1}$, $t_{i1}$ and $t_{i2}$ (i.e. the dotted line in figure 2) are not necessarily optimal to the original problem (straight lines in figure 2). To that purpose, we need to measure the quality of our solution by calculating the vertical distance $vd_i$ for each activity *i* between the straight and dotted line. If deviations occur, we need to branch on the activity with the largest vertical distance, which is the subject of the following subsection.

## 3.2. Calculation of $lb_2$ by means of $BB_1$

The basic idea of this solution approach relies on the approach used by Falk and Soland (1969) and Horst (1990). Their problem is to find the vector $x = (x_1, \ldots, x_n)$ which minimizes

$$\varphi(x) = \sum_{i=1}^{n} \varphi_i(x_i)$$

subject to

$$x \in G$$

$$l \le x \le L$$

for which it is assumed that $G$ is closed and that each $\varphi_i$ is lower semi-continuous, possibly nonconvex, on the interval $[l_i, L_i]$. In their paper, they have presented an algorithm for separable nonconvex programming problems. To that purpose, they solve a sequence of problems in a branch-and-bound approach in which the objective function is convex. These problems correspond to successive partitions of the feasible set. This approach has been successfully applied for a facility location problem by Soland (1974) and for a project scheduling problem by Vanhoucke et al. (2002). In this last paper, the authors present a branch-and-bound algorithm for an unconstrained project scheduling problem with activity-based cash flows which depend on the time of occurrence.

As mentioned in the previous subsection, the algorithm reports the optimal finishing times $of_i$ of the activities with their corresponding earliness or tardiness cost for the *WETPSP*. In order to measure the quality of our solution, a vertical distance $vd_i$ is computed based on the convex envelope. If all activities have a distance $vd_i = 0$ then no branching is needed and the reported solution $lb_1$ is optimal (i.e. $lb_2 = lb_1$). Otherwise, the algorithm selects the activity with the largest $vd_i$ and starts to branch. Branching is done based on the solution approach of Falk and Soland (1969) and corresponds to successive partitions of the feasible set. In the problem under study, we partition the earliness-tardiness profile into two disjoint subsets.

The algorithm calculates to new convex envelopes for these subsets and solves two new problems by means of the adapted *WETPSP* procedure of the previous subsection. Branching continues from the node with the lowest lower bound. If all activities at a particular node of the branch-and-bound tree have $vd_i = 0$, then we update the lower bound $lb_2$ of the project (initially set to $\infty$) and explore the second node at that level of the branch-and-bound tree. Backtracking occurs when the calculated lower bound is larger than or equal to the current lower bound $lb_2$. The algorithm stops when we backtrack to the initial level of the branch-and-bound tree and reports the optimal $lb_2$. This lower bound can be used as a lower bound at each node of the branch-and-bound algorithm $BB_2$. This is the subject of the following subsection.

## 3.3. Calculation of an exact solution for the *RCPSPMDD* by means of $BB_2$

The branching strategy of $BB_2$ boils down to resolving resource conflicts. Indeed, the branch-and-bound procedure $BB_1$ searches for an optimal solution for the unconstrained *PSPMDD* and consequently, ignores the limited availability of the renewable resources. If resource conflicts occur, we need to branch. A resource conflict occurs when there is at least one period $]t\text{-}1, t]$ for which $\exists k \leq K : \sum_{i \in S(t)} r_{ik} > a_k$ . To that purpose, we rely on an adapted version of the branching scheme developed by Demeulemeester and Herroelen (1992, 1997a,b) for the resource-constrained project scheduling problem (*RCPSP*). Since the earliness-tardiness minimization objective of the *RCPSPMDD* belongs to the class of nonregular objective functions, we modify this basic branching scheme to take the nonregular properties into account.

Demeulemeester and Herroelen (1992, 1997a,b) argue that it is sufficient to consider only a set of *minimal delaying alternatives DS* to resolve a resource conflict, i.e. *DS* contains minimal sets of activities which, when delayed, release enough resources to resolve the resource conflict and which do not contain any other delaying alternative as a subset. Each of these minimal delaying alternatives is delayed by each of the remaining activities in progress in period $]t^*\text{-}1,t]$ (the period of the *first* encountered resource conflict). Therefore, each minimal delaying alternative can give rise to several *minimal delaying modes* (De Reyck, 1998). For each delaying mode we impose additional precedence relations to resolve the resource conflict and compute a new lower bound $lb_2$ using the branch-and-bound scheme $BB_1$. If, for example, a resource conflict is caused in period $]t\text{-}1,t]$ by the set of activities $S(t)$= {1,2,3} and the delaying set contains two minimal delaying alternatives, i.e. $DS$ = {{1,2},{3}}, then the three different delaying modes are ($1 \prec 3$), ($2 \prec 3$) and ($3 \prec 1,2$) corresponding to four additional precedence relations.

Each delaying mode corresponds to a *node* in the branch-and-bound tree which will be further explored during the branching process. We select among these nodes the delaying mode with the smallest $lb_2$. If the lower bound of a node corresponds to a solution which is resource feasible, we update the upper bound *ub* of the project (initially $ub = \infty$) and we backtrack to the previous level. Moreover, if the lower bound $lb_2$ of a node is greater than or equal to the current upper bound, we fathom this node and we also backtrack to the previous level. If the lower bound is smaller than the current upper bound and a resource conflict occurs, we generate a new set of delaying alternatives at the next level of the tree. If there are no delaying modes left, we backtrack to the previous level, we search for the following delaying mode at this level and proceed in the same way. The algorithm stops when we backtrack to the initial level of the branch-and-bound tree.

In order to prune certain nodes of the branch-and-bound tree, we have implemented the so-called *subset dominance rule*. This dominance rule has originally been developed by De Reyck and Herroelen (1998) and has been applied in the branch-and-bound procedures of Vanhoucke et al. (2000, 2001). This dominance rule can be applied when the set of added precedence constraints (to resolve resource conflicts) of a previously examined node in the tree is a subset of the set of precedence constraints of the current node.

## 4. COMPUTATIONAL EXPERIENCE

In order to validate the efficiency of our branch-and-bound procedure, we have coded the *RCPSPMDD* in Visual C++ Version 6.0 under Windows NT 4.0 on a Compaq personal computer (Pentium 500 MHz processor). We have used two problem sets to validate the efficiency of our algorithm. The first set has been generated by the well-known network generator *ProGen/Max* (Schwindt, 1995) while the second set has been generated by a network generator *RanGen*, recently proposed by Demeulemeester et al. (2002).

### 4.1. The *ProGen/Max* instances

We generated 12,960 activity-on-the-node test instances with *ProGen/Max* (Schwindt, 1995). Table 2 represents the parameter settings used to generate the test instances for the *RCPSPMDD*. The parameters used in the full factorial experiment are indicated in bold. These instances in activity-on-the-node format use three settings for the number of activities, the order strength *OS* and the resource strength *RS* and four setting for the resource factor *RF*. The order strength, *OS* (Mastor, 1970), is defined as the number of precedence relations (including the transitive ones) divided by the theoretical maximum of number of precedence relations ($n(n$-1)/2, where $n$ denotes the number of activities). The resource factor *RF* (Pascoe (1966)) reflects the average portion of resource types requested per activity and consequently measures the density of the resource matrix $r_{ik}$. The resource strength *RS* was first introduced by Cooper (1976) and later used by Alvarez-Valdes and Tamarit (1989). We use the new definition, $RS_k = \frac{a_k - r_k^{\min}}{r_k^{\max} - r_k^{\min}}$, introduced by Kolisch et al. (1995) where $a_k$ denotes the total availability of renewable resource type $k$, $r_k^{\min}$ equals $\max_{i=1,\ldots,n} r_{ik}$ and $r_k^{\max}$ denotes the peak demand of resource type $k$ in the precedence preserving earliest start schedule. We provided the problems with due dates and penalty costs as explained below. Using 10 instances for each problem class, we obtain a problem set with 12,960 test instances.

**TABLE 2. Parameter settings used to generate the progen/max test instances for the rcpspmdd**

| RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEMS GENERATED BY *PROGEN/MAX* 12,960 *instances* | |
|---|---|
| **Number of activities** | 10, 20 or 30 |
| Activity durations | randomly selected from the interval [1,10] |
| Number of initial and terminal activities | randomly selected from the interval [2,4] |
| Maximal number of successors and predecessors | 4 |
| **Order strength *OS*** | 0.25, 0.50 or 0.75 |
| Number of resource types | 4 |
| Number of resources used per activity | randomly selected from the interval [1,4] |
| Activity resource demand | randomly selected from the interval [1,10] |
| **Resource factor *RF*** | 0.25, 0.50, 0.75 or 1.00 |
| **Resource strength *RS*** | 0.00, 0.25 or 0.50 |
| **Number of due dates *NDD*** | 5, 10, 15 or 20 |
| **[x, y] value for due date generation** | [2, 5], [2,10] or [2,15] |
| Unit penalty cost | randomly selected from the interval [1,10] |

We calculated the earliest finishing $ef_i$ of each activity $i$ based on forward pass calculations. In doing so, we were able to define the earliest possible project deadline $ef_n$. We then randomly generated numbers between 1 and the earliest possible project deadline $ef_n$. These number has been sorted and assigned to the activities in increasing order, activity 1 having the smallest number and activity $n$ the largest. These numbers are used to denote the first possible due date $h_{i1}$ of each activity $i$. All other possible due date $h_{ij}$ ($j$ = 2,… , $nr(i)$) are generated by adding a randomly generated number to the previous due date, i.e. $h_{ij} = h_{ij-1} + random[x, y]$. The values of $x$ and $y$ are used to denote the range from which the random number is generated and determines the spread of all possible due dates for an activity. We have used three different settings for the [$x$, $y$] range, i.e. [2, 5], [2, 10] or [2, 15]. The dummy end activity $n$ has only one due date which acts as a project deadline and which equals the largest possible due date over all activities. Depending on the tardiness cost of this due date, the project may have an overrun.

The generation of all possible due dates for activities is important since not all values are relevant for the search process. Some due dates that are outside the range [$ef_i$, $lf_i$] of the activity can

be removed. Only those due dates outside the range closest to either the $ef_i$ or the $lf_i$ are relevant for the search process. All other due dates outside this range have little value since they can never have any influence on the decision process of assigning due dates. Of course, due dates inside the time frame [$ef_i$, $lf_i$] are also relevant during the search process. Therefore, we have generated the due dates in such a way that most – if not all – of them are relevant.
Table 3.

Insert Table 3 About Here

In table 3 we report the computational results for the *RCPSPMDD*. To that purpose, we display the average CPU-time in seconds, the number of problems solved to optimality within 100 seconds CPU-time, the average number of created nodes in the branch-and-bound tree $BB_2$, the average number of branched nodes for $BB_2$ and the average number of created nodes in the branch-and-bound tree $BB_1$ (lower bound calculation of $BB_2$).

The row labelled '*all instances'* gives the average results over all 12,960 problem instances and illustrates the efficiency of our double branch-and-bound procedure. In the remaining rows we show more detailed results for the different parameters of our full factorial experiment.

The row labelled '*number of activities*' illustrates that the number of activities has a significant effect on the average CPU-time and on the number of problems solved to optimality. All problems with 10 activities can be solved to optimality within 100 second of CPU-time (with an average CPU-time of 0.02 seconds). For problems containing 20 activities, 89% of the number of problems can be solved to optimality within the allowed CPU-time (average CPU-time of 16.969 seconds). 62% of the problem instances with 30 activities can be solved to optimality within the allocated CPU-time (average CPU-time of 45.413 seconds). The number of nodes confirms that the number of activities has a significant effect on the problem complexity. The higher the number of activities, the more nodes needed to solve the problem (both resource-constrained ($BB_2$) and unconstrained ($BB_1$)). Note that each setting contains 4,320 instances. All other rows can be interpreted in a similar way. We give our basic conclusions without a detailed explanation of the figures of table 3.

The row labelled '*OS*' indicates that the order strength has a negative correlation with the problem hardness, that is, the higher *OS*, the easier the problem. This is in line with the literature and has been described in Vanhoucke et al. (2000, 2001).

Also the rows labelled '*RF*' and '*RS*' are completely in line with literature. The higher the resource factor, the more difficult the instances. An opposite effect can be observed for the resource strength. The number of problems solved to optimality increases when *RS* increases. These results were also observed by De Reyck and Herroelen (1996) and Vanhoucke et al. (2000, 2001).

The row labelled ‘*NDD*’ illustrates the effect of the number of due date possibilities on the problem complexity. It is quite straightforward that the number of due dates is positively correlated with the problem complexity. The more due date possibilities, the more choices the procedure has to evaluate in order to find the optimal solution.

The last row, labelled ‘[*x*, *y*]’, reveals that the higher the discrepancy between the due dates, the easier the problem to solve. A similar effect has been found for the *RCPSPWET* (Vanhoucke et al. (2000)). In this paper, the authors claim that activity due dates that are close to each other will result in a high complexity to solve the problem to optimality. The main reason is that the number of nodes in the search tree will increase dramatically since the probability of a resource conflict will increase. The same effect holds for [*x*, *y*], since the closer the due dates to each other, the larger the probability that activities are scheduled at the same time, resulting in one or more resource conflicts.

## 4.2. The *RanGen* instances

Table 4 represents the parameter settings used to generate the test instances for the *RCPSPMDD* by *RanGen* (Demeulemeester et al. (2002)). Apart from the resource related measures, these instances use the same parameter settings. The resource factor *RF* and the resource strength *RS* have been replaced by the resource use *RU* and the resource constrainedness *RC*, respectively. The resource use *RU* varies between zero and the number of resource types available and measures for each activity the number of resource types, i.e. $RU_i = \sum_{k=1}^{K} \begin{cases} 1, \text{if } r_{ik} > 0 \\ 0, \text{otherwise} \end{cases}$ $(1 \leq i \leq n)$. The resource-constrainedness *RC* is defined as $RC_k = \frac{\bar{r}_k}{a_k}$ where $a_k$ is defined as above and $\bar{r}_k$ denotes the average quantity of resource type *k* demanded when required by an activity, i.e. $\bar{r}_k = \sum_{i=1}^{n} r_{ik} \Big/ \sum_{i=1}^{n} \{1, \text{if } r_{ik} > 0; 0 \text{ otherwise}\}$. This measure has has been introduced by Patterson (1976). For arguments and a critical discussion for using either the resource strength *RS* or the resource-constrainedness *RC*, we refer the reader to Demeulemeester et al. (2002).

The generation of due dates and the corresponding penalty costs are similar as in section 4.1. Since we also use 10 instances for each problem class, we obtain a problem set with 17,280 test instances.

**TABLE 4. Parameter settings used to generate the rangen test instances for the rcpspmdd**

| **RESOURCE-CONSTRAINED PROJECT SCHEDULING PROBLEMS GENERATED BY *RANGEN*** 17,280 *instances* | |
|---|---|
| **Number of activities** | 10, 20 or 30 |
| Activity durations | randomly selected from the interval [1,10] |
| **Order strength *OS*** | 0.25, 0.50 or 0.75 |
| Number of resource types | 4 |
| Activity resource demand | randomly selected from the interval [1,10] |
| **Resource use *RU*** | 1, 2, 3 or 4 |
| **Resource constrainedness *RC*** | 0.25, 0.50, 0.75 or 1.00 |
| **Number of due dates *NDD*** | 5, 10, 15 or 20 |
| **[x, y] value for due date generation** | [2, 5], [2,10] or [2,15] |

Table 5.

Insert Table 5 About Here

In table 5 we report the computational results in a similar way as table 3. Although this table reveals similar results as for table 3, it appears that the *RanGen* instances are more difficult to solve. One possible explanation could be that these instances do not have additional parameters, such as the number of initial and terminal activities of a project or the maximal number of successors and predecessors. The overall computational results remain, as previously mentioned, the same. Note also that the average number of created nodes in $BB_1$ at each node of $BB_2$ is lower than for the *ProGen/Max* instances.

Since the effect of the resource *RU* and the resource constrainedness *RC* was not included in table 3 we now analyze these results. As expected, the resource use *RU* is positively correlated with the problem complexity. Indeed, the more resources in the project, the higher the probability for a resource conflict to occur. The effect of the resource constrainedness is in line with literature and can be explained as follows. The basic resource-constrained project scheduling problem (*RCPSP*) shows an easy-hard-easy transition (Herroelen and De Reyck (1999)) for the resource constrainedness. This means that instances with low and high values for the *RC* are easy to solve. In the former case, most activities can be scheduled in parallel while in the latter case, almost all activities can simply be scheduled in series. The difficulty is in between. This does not hold for the *RCPSPMDD* instances where we, instead, observe a easy-hard transition. Indeed, problem instances with a large value for the resource constrainedness will have relatively little freedom in scheduling the activities. While the *RCPSP* simply schedules the activities in series and the project makespan equals the sum of the activity durations, the *RCPSPMDD* is more difficult to solve. Even if there is little freedom in scheduling the activities (and consequently, the activities have to be scheduled in series), it still remains very important in which sequence we schedule these activities in order to minimize the weighted earliness tardiness costs. This has been explained in Vanhoucke (2001) with computational results for the *RCPSPWET* and the following example.  Compare the scheduling of three activities 1, 2 and 3 in series: the *RCPSP* only considers one sequence $1 \prec 2 \prec 3$ since all other sequences lead to the same minimal makespan. The *RCPSPWET* has to consider six different sequences ($1 \prec 2 \prec 3$, $1 \prec 3 \prec 2$, $2 \prec 1 \prec 3$, $2 \prec 3 \prec 1$, $3 \prec 1 \prec 2$, $3 \prec 2 \prec 1$), each possibly leading to a different value of the weighted earliness tardiness cost.

## 5. CONCLUSIONS

In this paper we presented a branch-and-bound procedure for the resource-constrained project scheduling problem with multiple due date possibilities (*RCPSPMDD*). The motivation for the new problem type lies in the observation that due dates are a result of negotiations and are not simply dictated by the client of the project. Therefore, the contractor has to choose among a set of due dates, based on negotiations with the client of the project and possible subcontractors. In doing so, he/she is in charge in scheduling the project in order to minimize the deviations from the chosen due dates (penalty costs). The depth-first branch-and-bound strategy to solve resource conflicts ($BB_2$) makes use of another branch-and-bound procedure ($BB_1$) to calculate the lower bounds. The branching scheme $BB_2$ was extended with the subset dominance rule to prune the search tree considerably.

The procedure has been coded in Visual C++, version 6.0 and has been tested on a randomly generated data set containing 12,960 instances. The results on a Compaq PC (Pentium 500 MHz) are encouraging. We have concluded that the problem complexity, as measured by the CPU-time, is positively correlated with the number of project activities, the resource factor and the number of due dates. The order strength and the spread of the due dates possibilities over time are negatively correlated with the problem complexity.

It is in our future intentions to broaden the research efforts of due date assignment in project scheduling. Also the development of heuristic solution methods to solve larger, real-life problems, lies within our future research possibilities.

## REFERENCES

Alvarez-Valdes, A. and Tamarit, J.M., 1989, “Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis”, in: Slowinski, R. and Weglarz, J. (Eds.), *Advances in Project Scheduling*, Elsevier, Amsterdam, 113-134.

Cooper, D.F., 1976, “Heuristics for scheduling resource-constrained scheduling projects: An experimental investigation”, *Management Science*, 22, 1186-1194.

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

Demeulemeester E. and Herroelen, W., 1997a, “New benchmark results for the resource-constrained project scheduling problem”, *Management Science*, 43, 1485-1492.

Demeulemeester E. and Herroelen, W., 1997b, “A branch-and-bound procedure for the generalized resource-constrained project scheduling problem”, *Operations Research*, 45, 201-212.

Demeulemeester, E., Vanhoucke, M. and Herroelen, W., 2002, “A random network generator for activity-on-the-node networks”, *Journal of Scheduling*, to appear.

De Reyck, B., 1998, Scheduling projects with generalized precedence relations – exact and heuristic procedures, Ph.D. Dissertation, Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.

De Reyck, B. and Herroelen, W., 1996, “A branch-and-bound procedure for the resource-constrained project scheduling problem with generalized precedence relations”, *European Journal of Operational Research*, 111, 125-174.

De Reyck, B. and Herroelen, W., 1998, “An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations”, *Computers and Operations Research*, 25, 1-17.

Falk, J.E. and Soland, R.M., 1969, “An algorithm for separable nonconvex programming problems”, *Management Science*, 15, 550-569.

Gordon, V.S., Proth, J-M. and Chu, C., 1998, “A state-of-the-art survey of due date assignment and scheduling research: SLK, TWK, and other due date assignment models”, *Production Planning and Control*, to appear.

Gordon, V.S., Proth, J-M. and Chu, C., 2002, “A survey of the state-of-the-art of common due date assignment and scheduling research”, *European Journal of Operational Research*, 139, 1-25.
Herroelen, W. and De Reyck, B., 1999, “Phase transitions in project scheduling”, *Journal of Operational Research Society*, 50, 148-156.

Horst, R., 1990, “Deterministic methods in constrained global optimization: Some recent advances and new fields of application”, *Naval Research Logistics*, 37, 433-471.

Kolisch, R., Sprecher, A. and Drexl, A., 1995, “Characterization and generation of a general class of resource-constrained project scheduling problems”, *Management Science*, 41, 1693-1703.

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

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

Patterson, J.H., 1976, “Project scheduling: the effects of problem structure on heuristic scheduling”, *Naval Research Logistics*, 23, 95-123.

Schwindt, C., 1995, “A new problem generator for different resource-constrained project scheduling problems with minimal and maximal time lags”, WIOR-Report-449, Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe.

Soland, R.M., 1974, “Optimal facility location with concave costs”, *Operations Research*, 22, 373-382.

Ulusoy, G. and Cebelli, S., 2000, “An equitable approach to the payment scheduling problem in project management”, *European Journal of Operational Research*, 127, 262-278.

Vanhoucke, M., 2001, “Exact algorithms for various project scheduling problems: Nonregular measures of performance and time/cost trade-offs”, PhD Dissertation.

Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 1999, “An exact procedure for the unconstrained weighted earliness-tardiness project scheduling problem”, Onderzoeksrapport 9907, Departement Toegepaste Economische Wetenschappen, K.U.Leuven, 13 pp.

Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 2000, “An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem”, *Annals of Operations Research*, Vol. 102, 179-196.

Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 2001, “On maximizing the net present value of a project under renewable resource constraints”, *Management Science*, Vol. 47, 1113-1121.
Vanhoucke, M., Demeulemeester, E. and Herroelen, W., 2002, “Progress payments in project scheduling problems”, *European Journal of Operational Research,* to appear

**TABLE 3.**

**Computational results for the RCPSPMDD using instances generated with ProGen/Max**

| | **Average CPU time (seconds)** | **Number of problems solved to optimality** | **Average number of created nodes (*$BB_2$*)** | **Average number of branched nodes (*$BB_2$*)** | **Average number of created nodes in *$BB_1$* at each node of *$BB_2$*** |
|---|---|---|---|---|---|
| **All instances** | 20.801 | 10,857 | 68.465 | 57.337 | 40.332 |
| **Number of** | | | | | |
| **activities** | 0.020 | 4320 | 46.725 | 28.930 | 9.609 |
| 10 | 16.969 | 3858 | 75.396 | 65.361 | 44.117 |
| 20 | 45.413 | 2679 | 83.275 | 77.719 | 67.269 |
| 30 | | | | | |
| ***OS*** | | | | | |
| 0.25 | 38.552 | 2943 | 86.366 | 76.878 | 39.004 |
| 0.50 | 16.085 | 3806 | 69.136 | 56.370 | 41.551 |
| 0.75 | 7.766 | 4108 | 49.894 | 38.763 | 40.439 |
| ***RF*** | | | | | |
| 0.25 | 4.494 | 3195 | 35.491 | 22.872 | 42.896 |
| 0.50 | 16.521 | 2854 | 69.043 | 55.573 | 42.528 |
| 0.75 | 27.432 | 2509 | 80.890 | 70.570 | 39.504 |
| 1.00 | 34.755 | 2299 | 88.436 | 80.333 | 36.398 |
| ***RS*** | | | | | |
| 0.00 | 26.338 | 3406 | 74.030 | 65.974 | 35.792 |
| 0.25 | 21.946 | 3577 | 69.534 | 59.039 | 41.013 |
| 0.50 | 14.118 | 3874 | 61.831 | 46.997 | 44.189 |
| ***NDD*** | | | | | |
| 5 | 13.395 | 2915 | 66.677 | 56.921 | 20.256 |
| 10 | 21.509 | 2698 | 69.025 | 57.372 | 40.655 |
| 15 | 24.062 | 2629 | 69.135 | 57.668 | 49.041 |
| 20 | 24.237 | 2615 | 69.024 | 57.387 | 51.373 |
| **[*x*, *y*]** | | | | | |
| [2, 5] | 18.637 | 3678 | 68.897 | 58.315 | 31.779 |
| [2,10] | 21.834 | 3595 | 69.003 | 57.649 | 43.251 |
| [2,15] | 21.931 | 3584 | 67.496 | 56.047 | 45.965 |

**TABLE 5.**

**Computational results for the RCPSPMDD using instances generated with RanGen**

| | **Average CPU time (seconds)** | **Number of problems solved to optimality** | **Average number of created nodes (*BB*$_2$)** | **Average number of branched nodes (*BB*$_2$)** | **Average number of created nodes in *BB*$_1$ at each node of *BB*$_2$** |
|---|---|---|---|---|---|
| **All instances** | 43.517 | 10,212 | 78.924 | 71.517 | 22.881 |
| **Number of activities** | 0.099 | 5760 | 57.378 | 44.173 | 7.825 |
| 10 | 53.149 | 3012 | 86.021 | 80.557 | 26.672 |
| 20 | 77.304 | 1440 | 93.372 | 89.821 | 34.145 |
| 30 | | | | | |
| ***OS*** | | | | | |
| 0.25 | 57.344 | 2517 | 89.312 | 84.040 | 17.856 |
| 0.50 | 46.064 | 3258 | 81.844 | 74.912 | 21.940 |
| 0.75 | 27.144 | 4437 | 65.615 | 55.599 | 28.845 |
| ***RU*** | | | | | |
| 1 | 30.462 | 3123 | 59.043 | 49.905 | 28.101 |
| 2 | 43.544 | 2530 | 80.558 | 72.422 | 24.646 |
| 3 | 48.318 | 2348 | 87.304 | 80.869 | 20.121 |
| 4 | 51.745 | 2211 | 88.791 | 82.872 | 18.654 |
| ***RC*** | | | | | |
| 0.25 | 24.602 | 3328 | 49.583 | 36.353 | 25.489 |
| 0.50 | 48.772 | 2338 | 85.655 | 79.643 | 25.258 |
| 0.75 | 50.547 | 2270 | 89.723 | 84.356 | 20.758 |
| 1.00 | 50.149 | 2276 | 90.735 | 85.716 | 20.017 |
| ***NDD*** | | | | | |
| 5 | 31.851 | 3069 | 76.684 | 70.006 | 10.714 |
| 10 | 45.553 | 2476 | 79.643 | 72.077 | 20.211 |
| 15 | 48.146 | 2339 | 79.423 | 71.860 | 28.533 |
| 20 | 48.519 | 2328 | 79.947 | 72.125 | 32.064 |
| **[*x*, *y*]** | | | | | |
| [2, 5] | 39.394 | 3640 | 78.472 | 71.500 | 15.385 |
| [2,10] | 44.644 | 3333 | 79.704 | 72.422 | 24.841 |
| [2,15] | 46.515 | 3239 | 78.596 | 70.629 | 28.416 |

z