Scheduling an R&D Project with Quality-Dependent Time Slots

Mario Vanhoucke[1,2]

[1] Faculty of Economics and Business Administration,
Ghent University, Gent, Belgium
[2] Operations & Technology Management Centre,
Vlerick Leuven Gent Management School, Gent, Belgium
mario.vanhoucke@ugent.be

Abstract. In this paper we introduce the concept of quality-dependent time slots in the project scheduling literature. Quality-dependent time slots refer to pre-defined time windows where certain activities can be executed under ideal circumstances (optimal level of quality). Outside these time windows, there is a loss of quality due to detrimental effects. The purpose is to select a quality-dependent time slot for each activity, resulting in a minimal loss of quality. The contribution of this paper is threefold. First, we show that an R&D project from the bio-technology sector can be transformed to a resource-constrained project scheduling problem (RCPSP). Secondly, we propose an exact search procedure for scheduling this project with the aforementioned quality restrictions. Finally, we test the performance of our procedure on a randomly generated problem set.

1 Introduction

In the last decades, the research in resource-constrained project scheduling has been investigated from different angles and under different assumptions. The main focus on project time minimization has shifted towards other objectives (e.g. net present value maximization), extensions such as multi-mode scheduling and/or preemption, and many other facets (for the most recent overview, see [1]).

However, the literature on project scheduling algorithms where quality considerations are taken into account is virtually void. [9] maximize the quality in the resource-constrained project scheduling problem by taking the rework time and rework cost into account. They argue that their work is a logical extension of the classical resource-constrained project scheduling efforts. Therefore, they refer to a previous study by the same authors that indicated that over 90% of the project managers take the maximization of the quality of projects and their outcomes as their primal objective. Given that emphasis on quality management and its implementation in project management, and the need to develop new tools and techniques for scheduling decisions, we elaborate on that issue based on a real-life project in the bio-technology sector.

The motivation of this research paper lies in the effort we put in the scheduling of a project with genetically manipulated plants. In this project, several activities need to be scheduled in the presence of limited resources and severe quality restrictions. More

M. Gavrilova et al. (Eds.): ICCSA 2006, LNCS 3982, pp. 621 – 630, 2006.

precisely, some activities need to be executed preferably within certain pre-defined periods, referred to as *quality-dependent time slots*. Although the execution is also possible outside these pre-defined intervals, it is less desirably since it leads to a decrease in quality. The concept of pre-defined time windows for activity execution is not new in the project scheduling literature. [2] criticize the traditional models in which it is assumed that an activity can start at any time after the finishing of all its predecessors. To that purpose, they consider two improvements over the traditional activity networks by including two types of time constraints. *Time-window constraints* assume that an activity can only start within a specified time interval. *Time-schedule constraints* assume that an activity can only begin at one of an ordered schedule of beginning times. [15] elaborate on these time constraints and argue that time can be treated as a repeating cycle where each cycle consists of two categories: (*i*) some pairs of rest and work windows and (*ii*) a leading number specifying the maximal number of time each pair should iterate. By incorporating these so-called *time-switch constraints*, activities are forced to start in a specific time interval and to be down in some specified rest interval. *Quality-dependent time slots* refer to pre-defined time windows where certain activities can be executed under ideal circumstances (optimal level of quality). Outside these time windows, there is a loss of quality due to detrimental effects. Unlike the time-switch constraints, the quality-dependent time slots allow the execution of the activity outside the pre-defined window leading to an extra cost or decrease in quality. Consequently, the quality-dependent time slots are similar to the time-switch constraints. The latter are hard constraints (execution is only possible *within* the interval) and the former are soft constraints (execution is preferable *within* the interval but is also possible *outside* the interval) that can be violated at a certain penalty cost.

Our project settings assume that each activity has several pre-defined quality-dependent time slots, from which one has to be selected. The selection of a time slot must be done before the start of the project (in the planning phase). Given a fixed set of time slots per activity, the target is then to select a time slot and to schedule the project such that the loss in quality will be minimized.

2 Description of the Problem

The project under study can be represented by an activity-on-the-node 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 quality-dependent time windows $nr(i)$. Each window l of activity i ($1 \leq i \leq n$ and $1 \leq l \leq nr(i)$) is characterized by a time-interval $\left[\overline{q}_{il}^{-}, \overline{q}_{il}^{+}\right]$ of equal quality, while deviations outside that interval result in a loss of quality. Note that the time slot $\left[\overline{q}_{il}^{-}, \overline{q}_{il}^{+}\right]$ is used to refer to a window with optimal quality and can be either an interval or a single point-in-time. The quality deviation of each activity i can be computed as

$Q_i^{\text{loss}} = \max\{q_{il}^- - s_i ; s_i - q_{il}^+ ; 0\}$ and depends on the selection of the time window l, with s_i the starting time of activity i. To that purpose, we need to introduce a binary decision variable in our conceptual model which determines the selection of a specific time interval for each activity i, $y_{il} = \begin{cases} 1, \text{if time interval } l \text{ has been selected for activity } i \\ 0, \text{otherwise} \end{cases}$.

We use q_{il}^{opt} to denote the minimal activity cost associated with a fixed and optimal level of quality for each time window l of activity i. We use q_{il}^{extra} to denote the loss in quality per time unit deviation from the time interval and consequently, the total cost of quality equals $\sum_{i=1}^{n}\sum_{l=1}^{nr(i)}(q_{il}^{\text{opt}} + q_{il}^{\text{extra}} Q_i^{\text{loss}})\, y_{il}$. Note that $nr(0) = nr(n) = 1$, since nodes 0 and n are dummy activities with $\bar{q}_{01}^{-} = \bar{q}_{01}^{+}$ and $\bar{q}_{n1}^{-} = \bar{q}_{n1}^{+}$. Moreover, we set $q_{01}^{\text{extra}} = \infty$ to force the dummy start activity to start at time instance zero. The project needs to be finished before a negotiated project deadline δ_n, i.e. $\bar{q}_{n1}^{-} = \bar{q}_{n1}^{+} = \delta_n$. Consequently, setting $q_{n1}^{\text{extra}} = \infty$ denotes that the project deadline can not be exceeded (a hard constraint), while $q_{n1}^{\text{extra}} < \infty$ means that the project deadline can be exceeded at a certain penalty cost (soft constraint).

There are K renewable resources with a_k ($1 \le k \le K$) as the availability of resource type k and with r_{ik} ($1 \le i \le n$, $1 \le k \le K$) as the resource requirements of activity i with respect to resource type k. The project with renewable resources and quality-dependent time windows can be conceptually formulated as follows:

$$\text{Minimize} \sum_{i=1}^{n}\sum_{l=1}^{nr(i)}(q_{il}^{\text{opt}} + q_{il}^{\text{extra}} Q_i^{\text{loss}})\, y_{il} \tag{1}$$

Subject to

$$s_i + d_i \le s_j \qquad \forall (i, j) \in A \tag{2}$$

$$\sum_{i \in S(t)} r_{ik} \le a_k \qquad k = 1, \ldots, K \text{ and } t = 1, \ldots, T \tag{3}$$

$$Q_i^{\text{loss}} \ge \sum_{l=1}^{nr(i)} q_{il}^- y_{il} - s_i \qquad i = 1, \ldots, n \tag{4}$$

$$Q_i^{\text{loss}} \ge s_i - \sum_{l=1}^{nr(i)} q_{il}^+ y_{il} \qquad i = 1, \ldots, n \tag{5}$$

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

$$s_1 = 0 \tag{7}$$

$$s_i \in \text{int}^+, Q_i^{\text{loss}} \in \text{int}^+ \qquad i = 1, \ldots, n \tag{8}$$

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

where $S(t)$ denotes the set of activities in progress in period $]t\text{-}1,t]$. The objective in Eq. 1 minimizes the total quality cost of the project (i.e. the fixed cost within the selected time window plus the extra cost of quality loss due to deviations from that interval). The constraint set given in Eq. 2 maintains the finish-start precedence relations among the activities. Eq. 3 represents the renewable resource constraints and the constraint sets in Eq. 4 and Eq. 5 compute the deviation from the selected time window of each activity. Eq. 6 represents the time window selection and forces to select a single time window for each activity. Eq. 7 forces the dummy start activity to start at time zero and Eq. 8 ensures that the activity starting times as well as the time window deviations assume nonnegative integer values. Eq. 9 ensures that the time window selection variable is a binary (0/1) variable. Remark that the quality loss function measuring the quality decrease due to a deviation from the ideal time window l can be off any form (such as stepwise functions, convex functions, etc…). However, we assume in Eqs. [1]-[9], without loss of generality, a linear quality deviation function.

Although our first acquaintance with this problem type was during the scheduling of a genetically manipulated plants project, we believe that there are numerous other examples where pre-defined time-windows need to be selected before the execution of the project. The following four examples illustrate the possible generalization of multiple quality-dependent time windows to other project environments:

Perishable Items. The project of this paper, which motivates us to elaborate on this issue, is a typical example where items (i.e. plants) are perishable. Many project activities consist of tests on growing plants where the quality is time-dependent since there is an optimal time interval of consumption. Earlier consumption is possible, at a cost of a loss in quality, since the plants are still in their ripening process. Later consumption results in loss of quality due to detrimental effects.

State-of-Nature Dependencies. In many projects, the performance of some activities might depend on the state-of-nature. In this case, a pre-defined set of possible starting times depending on the state-of-nature are linked with possible execution times of the activity, and the deviation from these time windows is less desirable (resulting in higher costs or quality loss) or even completely intolerable.

Multiple Activity Milestones. The project scheduling literature with due dates (milestones) has been restricted to considering projects with pre-assigned due dates (see e.g. [11] and [12]). In reality, milestones are the results of negotiations, rather than simply dictated by 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 and the contractor (and possibly some subcontractors). This results in a set of possible due dates for each activity, rather than a single pre-defined due date. The objective is then to select a due date for each activity such that the total earliness/tardiness penalty costs will be minimized.

Time-Dependent Resource Cost. In many projects, the cost of (renewable) resources heavily depends on the time of usage. The aforementioned time-switch constraints are a typical and extreme example of time-dependent resource costs, since it restricts the execution of activities to pre-defined time intervals (work periods) without any possibility to deviate. However, if we allow the activities to deviate from their original work periods (e.g. by adding more (expensive) resources to an activity in the pre-defined rest period), the work periods can be considered as the quality-dependent time slots while the rest periods are periods outside these slots in which the activity can be executed at an additional cost.

3 The Algorithm

The problem type under study requires the selection of a quality dependent time-window from a possible set of windows such that the total quality loss is minimized. A closer look to the problem formulation of (1)-(9) reveals the following observations:

- When $K = 0$, i.e. there are no resources with limited capacity, the problem given in (1)-(9) reduces to an unconstrained project scheduling problem with non-regular measures of performance. Due to the special structure of the quality loss functions and the multiple time-windows, the problem reduces to a separable nonconvex programming problem (see "solution algorithm for problem (1)-(2), (4)-(9)" described below).
- When $K > 0$, i.e. there is at least one renewable resource with limited capacity, resource conflicts can arise during the scheduling of the project. Therefore, this problem type can be solved to optimality by any branch-and-bound enumeration scheme for project scheduling problems with non-regular measures of performance (see "solution algorithm for problem (1)-(9)" described below).

In this section we describe a double branch-and-bound algorithm for the problem type under study. The first branch-and-bound procedure ignores the renewable resource constraints and searches for an exact solution of the unconstrained project scheduling problem. The second branch-and-bound procedure aims at resolving resource conflicts and needs the previously mentioned solution as an input in every node of the tree.

Solution Algorithm for Problem (1)-(2), (4)-(9): The basic idea of this solution approach relies on the approach used by [6] and [7]. Their problem is to find the vector $x = (x_1, \ldots, x_n)$ which minimizes

$$\varphi(x) = \sum_{i=1}^{n} \varphi_i(x_i) \text{ subject to } x \in G \text{ and } l \le x \le L \tag{10}$$

for which it is assumed that G is closed and that each φ_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 different optimization problems.

[10] have shown that this problem type is a special case of irregular starting-time costs project scheduling which can be formulated as a maximum-flow problem and hence, can be solved using any maximum-flow algorithm.

Due to the special structure of the convex envelope, we rely on an adapted procedure of [14] developed for an unconstrained project scheduling problem with activity-based cash flows which depend on the time of occurrence. This branch-and-bound procedure basically runs as follows. At each node, we calculate a lower bound for the total quality cost $q_{il}^{\text{opt}} + q_{il}^{-}Q_i^{-} + q_{il}^{+}Q_i^{+}$ for each activity i. To that purpose, we construct the *convex envelope* of the total quality cost profile over the whole time window $[es_i, lf_i]$ for each activity i (es_i = earliest start and lf_i = latest finish). The convex envelope of a function $F = q_{il}^{\text{opt}} + q_{il}^{-}Q_i^{-} + q_{il}^{+}Q_i^{+}$ ($l = 1, \ldots, nr(i)$) taken over $C = [es_i, lf_i]$ is defined as the highest convex function which fits below F.

If the reported solution is not feasible for the original problem, the algorithm starts to branch. The algorithm calculates two new convex envelopes for these subsets and solves two new problems at each node. Branching continues from the node with the lowest lower bound. If all activities at a particular node of the branch-and-bound tree are feasible, then we update the upper bound of the project (initially set to ∞) 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. The algorithm stops when we backtrack to the initial level of the branch-and-bound tree and reports the optimal lower bound. This lower bound can be calculated at each node of the branch-and-bound algorithm described hereunder.

Solution Algorithm for Problem (1)-(9): In order to take the renewable resource constraints into account (i.e. equation (3)) we rely on a classical branch-and-bound approach that uses the unconstrained solutions as lower bounds at each node of the tree. Since the previous branch-and-bound procedure searches for an optimal solution for the unconstrained project and consequently, ignores the limited availability of the renewable resources, the second branching strategy boils down to resolving resource conflicts. If resource conflicts occur, we need to branch. A resource conflict occurs when there is at least one period $]t - 1, t]$ for which $\exists k \leq K : \sum_{i \in S(t)} r_{ik} > a_k$. To that purpose, we rely on the branch-and-bound approach of [8] for the resource-constrained project scheduling problem with discounted cash flows. This is an adapted version of the branching scheme developed by [3] for the resource-constrained project scheduling problem and is further enhanced by [5], [12] and [13].

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 [5] and has been applied in the branch-and-bound procedures of [12]

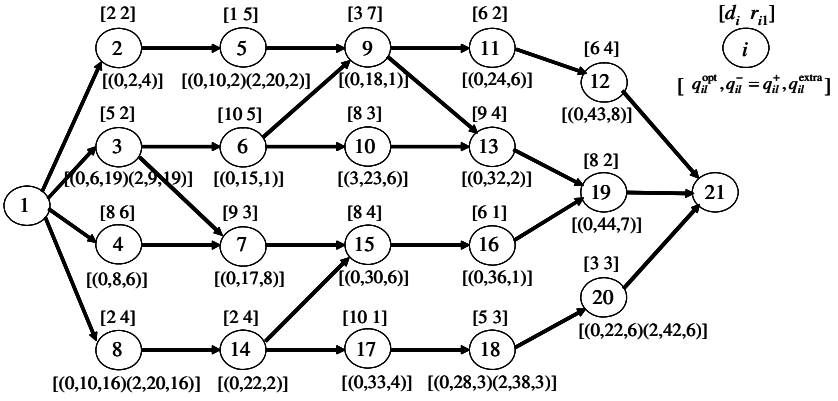

Fig. 1. An example project with quality-dependent time slots

and [13]. 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.

Example: We illustrate the project scheduling problem with limited resources and quality-dependent time slots by means of an example project of figure 1. The two numbers above each node are used to denote the activity duration d_i and its requirement r_{i1} for a single renewable resource with availability $a_1 = 10$. The numbers below the node are used to denote $(q_{il}^{\text{opt}}, q_{il}^- = q_{il}^+, q_{il}^{\text{extra}})$ for each quality-dependent time slot l. For the sake of clarity, we assume that $q_{il}^- = q_{il}^+$, i.e. the quality-dependent time slots are a single point in time. Moreover, we assume q_{il}^{extra} is equal for each interval l.

Each activity of the example project belongs to one of the following categories. An activity can be the subject to single (activities 12 and 17) or multiple quality-dependent time slots (activities 3, 5, 8, 18 and 20). Activities can also have the requirement to be scheduled as-soon-as-possible (ASAP; activities 2, 4, 6, 7, 9, 10, 11 and 13) or as-late-as-possible (ALAP; activities 14, 15, 16, 19 and 21). This can be incorporated in the network by adding a single quality-dependent time slot with $q_{i1}^- = q_{i1}^+ = es_i$ (ASAP) or $q_{i1}^- = q_{i1}^+ = ls_i$ (ALAP), with es_i (ls_i) the earliest start (finishing) time of activity i. Deviations from these requirements will be penalized by q_{i1}^{extra} per time unit. We assume that the project deadline T equals 44 time units.

Figure 2 displays the schedules found by solving the RCPSP without (i.e. minimization of project time) and with the quality-dependent time slots. The activities highlighted in dark grey are the activities that are scheduled at a different time instance between the two schedules. Activities 3, 6, 8, 14, 10, 17, 13, 18, 12 and 20 have been scheduled later than the classical RCPSP schedule, while activities 5 and 7 have been scheduled earlier.

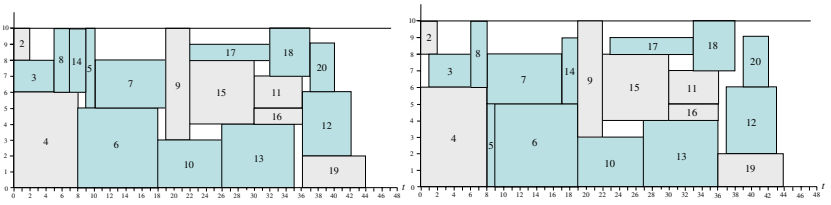

Fig. 2. The RCPSP schedule with minimal time (left) and quality-dependent time slots (right)

4 Computational Experience

In order to validate the efficiency, we have coded our double B&B procedure in Visual C++ Version 6.0 under Windows NT 4.0 on a Compaq personal computer (Pentium 500 MHz processor). We have generated a problem set by the RanGen network generator of [4] with 10, 20 and 30 activities.

In order to test the presence of renewable resources on the performance of our B&B procedure, we have extended each network instance with renewable resources under a pre-defined design. To that purpose, we rely on the resource use *RU* and the resource-constrainedness *RC* which can be generated by RanGen. More precisely, we use 4 settings for the *RU* (1, 2, 3 or 4) and 4 settings for the *RC* (0.25, 0.50, 0.75 or 1).

In order to generate data for the quality-dependent time slots, we need to generate values for the number of time slots $nr(i)$, the start and finishing time instance per time slot l ($\overline{-}_{il}$ and $\overline{-}^{+}_{il}$) and the quality deviation per time unit for each time slot l (q^{extra}_{il}). We used 4 different settings to generate the number of time slots per activity, i.e. $nr(i)$ equals 5, 10, 15 or 20. The start and finishing times of each time slot have been carefully selected between the earliest start time and the latest finish time of each activity, such that the 'spread' of the time slots have been generated under 3 settings, i.e. low (time slots close to each other), average and high (time slots far from each other). The q^{extra}_{il} values have been randomly generated. Using 30 instances for each setting, we have created 17,280 problem instances.

In table 1 we report the computational results for the project scheduling problem with renewable resource constraints and quality-dependent time-slots. To that purpose, we display the average CPU-time in seconds (Avg.CPU), the number of problems solved to optimality within 100 seconds CPU-time (#Solved), the average number of created nodes in the main branch-and-bound tree (#Avg.CN) to resolve resource conflics, the average number of branched nodes for this B&B tree (Avg.BN) and the average number of created nodes (Avg.CN2) in the unconstrained branch-and-bound tree (lower bound calculation) at each node of the main B&B tree. The row labelled *'all instances'* gives the average results over all 17,280 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.

Table 1. Computational results for the RCPSP with quality-dependent time slots

		Avg.CPU	#Solved	Avg.CN	Avg.BN	Avg.CN2
All instances		43.517	10,212	78.924	71.517	22.881
Number of activities	10	0.099	5,760	57.378	44.173	7.825
	20	53.149	3,012	86.021	80.557	26.672
	30	77.304	1,440	93.372	89.821	34.145
RU	1	30.462	3,123	59.043	49.905	28.101
	2	43.544	2,530	80.558	72.422	24.646
	3	48.318	2,348	87.304	80.869	20.121
	4	51.745	2,211	88.791	82.872	18.654
RC	0.25	24.602	3,328	49.583	36.353	25.489
	0.5	48.772	2,338	85.655	79.643	25.258
	0.75	50.547	2,270	89.723	84.356	20.758
	1	50.149	2,276	90.735	85.716	20.017
nr(i)	5	31.851	3,069	76.684	70.006	10.714
	10	45.553	2,476	79.643	72.077	20.211
	15	48.146	2,339	79.423	71.860	28.533
	20	48.519	2,328	79.947	72.125	32.064
Spread	low	39.394	3,640	78.472	71.500	15.385
	mid	44.644	3,333	79.704	72.422	24.841
	high	46.515	3,239	78.596	70.629	28.416

As expected, the *RU* and the *RC* are positively correlated with the problem complexity. Indeed, the more resources in the project and the tighter their constrainedness, the higher the probability for a resource conflict to occur. These effects are completely in line with literature (see e.g. [13]). The number of time slots is positively correlated with problem complexity. The spread of the time slots is positively correlated with the problem complexity. When time slots are close to each other, the selection of an activity time slot is rather straightforward since only one (or a few) are relevant.

5 Conclusions and Areas for Future Research

In this paper we presented a double branch-and-bound procedure for the resource-constrained project scheduling problem with quality-dependent time slots. The depth-first branch-and-bound strategy to solve renewable resource conflicts makes use of another branch-and-bound procedure to calculate the lower bounds at each node. The branching scheme has been extended with the subset dominance rule to prune the search tree considerably. The incorporation of quality-dependent time slots in project scheduling is, to the best of our knowledge, completely new.

It is in our future intentions to broaden the research efforts of quality-dependent time slot selection in project scheduling. More precisely, the introduction of *dynamic quality-dependent time slots* should open the door to many other applications. In this case, each activity can be executed several times, preferably within the pre-defined time slots. Moreover, the different time slots per activity depend on each other, in the sense that the time interval $\left[\bar{q}_{il}^{-}, \bar{q}_{il}^{+}\right]$ depends on the finishing time of the activity in or around the previous time slot = $\left[\bar{q}_{il-1}^{-}, \bar{q}_{il-1}^{+}\right]$. A typical example is a maintenance

operation that needs to be done within certain time limits. A second maintenance operation depends on the execution of the first one, resulting in a second time slot that depends on the starting time of the activity around the first time slot. The development of heuristic solution methods to solve larger, real-life problems instances also lies within our future research intentions.

References

1. Brücker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: notation, classification, models and methods. European Journal of Operational Research. 112 (1999) 3-41
2. Chen, Y.L., Rinks, D., Tang, K.: Critical path in an activity network with time constraints. European Journal of Operational Research. 100 (1997) 122-133
3. Demeulemeester E., Herroelen, W.: A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Management Science. 38 (1992) 1803-1818
4. Demeulemeester, E., Vanhoucke, M., Herroelen, W.: A random network generator for activity-on-the-node networks. Journal of Scheduling. 6 (2003) 13-34
5. De Reyck, B., Herroelen, W.: An optimal procedure for the resource-constrained project scheduling problem with discounted cash flows and generalized precedence relations. Computers and Operations Research. 25 (1998) 1-17
6. Falk, J.E., Soland, R.M.: An algorithm for separable nonconvex programming problems. Management Science. 15 (1969) 550-569
7. Horst, R.: Deterministic methods in constrained global optimization: Some recent advances and new fields of application. Naval Research Logistics. 37 (1990) 433-471
8. Icmeli, O., Erengüç, S.S.: A branch-and-bound procedure for the resource-constrained project scheduling problem with discounted cash flows. Management Science. 42 (1996) 1395-1408
9. Icmeli-Tukel, O., Rom, W.O.: Ensuring quality in resource-constrained project scheduling. European Journal of Operational Research. 103 (1997) 483-496
10. Möhring, R.H., Schulz, A.S., Stork, F., Uetz, M.: On project scheduling with irregular starting time costs. Operations Research Letters. 28 (2001) 149-154
11. Schwindt, C.: Minimizing earliness-tardiness costs of resource-constrained projects. In: Inderfurth, K., Schwoediauer, G., Domschke, W., Juhnke, F., Kleinschmidt, P., Waescher, G. (eds.). Operations Research Proceedings. Springer. (1999) 402-407
12. Vanhoucke, M., Demeulemeester, E., Herroelen, W.: An exact procedure for the resource-constrained weighted earliness-tardiness project scheduling problem. Annals of Operations Research. 102 (2000) 179-196
13. Vanhoucke, M., Demeulemeester, E., Herroelen, W.: On maximizing the net present value of a project under renewable resource constraints. Management Science. 47 (2001) 1113-1121
14. Vanhoucke, M., Demeulemeester, E., Herroelen, W.: Progress payments in project scheduling problems. European Journal of Operational Research. 148 (2003) 604-620
15. Yang, H.H., Chen, Y.L.: Finding the critical path in an activity network with time-switch constraints. European Journal of Operational Research. 120 (2000) 603-613