UNIVERSITEIT GENT

FACULTEIT ECONOMIE EN BEDRIJFSKUNDE

HOVENIERSBERG 24
B-9000 GENT
Tel. : 32 - (0)9 – 264.34.61
Fax. : 32 - (0)9 – 264.35.92

WORKING PAPER

Pre-emptive resource-constrained project scheduling with setup times

Dieter Debels [1]
Mario Vanhoucke [2]

May 2006

2006/391

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

D/2006/7012/36

ABSTRACT

Resource-constrained project scheduling with activity pre-emption assumes that activities are allowed to be interrupted and restarted later in the schedule at no extra cost. In the current paper, we extend this pre-emptive scheduling problem with setup times between activity interruptions and the possibility to fast track pre-emptive subparts of activities.

The contribution of the paper is twofold. First, we present an optimal branch-and-bound procedure for the pre-emptive resource-constrained project scheduling problem with setup times and fast tracking options. Second, we test the impact of these pre-emptive extensions to the quality of the schedule from a lead-time point-of-view.

1 Introduction

The well-known *resource-constrained project scheduling problem* (RCPSP) is one of the most widely studied problems in project scheduling and can be stated as follows. In a project network $G(N,A)$ in activity-on-the-node (AoN) format, we have a set of nodes N representing the n activities (numbered from 1 to n, i.e. $|N| = n$) and a set of pairs of activities A representing the precedence relations between the activities. Furthermore, project execution requires a set of resources R with a constant availability a_k for each resource type $k \in R$ throughout the project horizon. Each activity $i \in N$ is assumed to have a deterministic duration $d_i \in \mathbb{N}$ and requires $r_{ik} \in \mathbb{N}$ units of resource type k. The dummy start and end activities 1 and n have zero duration and zero resource usage. A schedule can be defined by an n-vector of finish times $(f_1, ..., f_n)$, and implies an n-vector of start times $(s_1, ..., s_n)$ such that $s_i + d_i$ equals f_i. A schedule is said to be *feasible* if it is non-pre-emptive and if both the precedence and renewable resource constraints are satisfied, and *optimal* if the project makespan f_n is minimized. Figure 1 displays an example project network that will be used throughout the remainder of this manuscript. Each activity has a fixed duration d_i shown above and a single resource requirement r_{i1} shown below the node. The resource availability a_1 equals 6 units. The optimal schedule is displayed at the right of the figure, and has a minimal makespan of 11 time units.

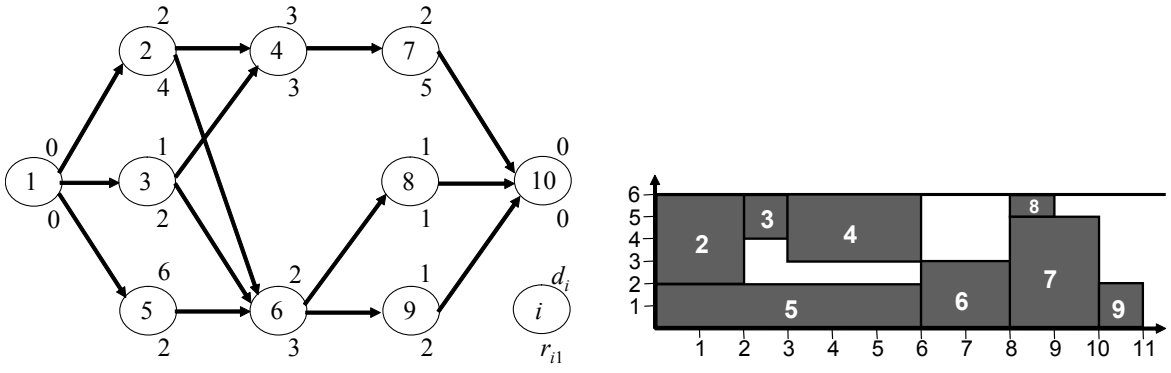

Figure 1. An example project with a corresponding optimal RCPSP schedule

The research on the RCPSP has widely expanded over the last few decades, and reviews can be found in Icmeli et al. (1993), Özdamar and Ulusoy (1995), Herroelen, et al. (1998), Brucker et al. (1999) and Kolisch and Padman (2001). In literature, various RCPSP extensions have been proposed. In this paper, we extend the RCPSP by relaxing two strict activity assumptions, as follows.

1) Activity pre-emption: the activities are allowed to be interrupted during execution
2) Activity fast tracking: The pre-emptive sub-parts of an activity can be executed in parallel

The pre-emptive resource-constrained project scheduling problem (PRCPSP) includes the first relaxation and assumes that activities can be pre-empted at any integer time instant and restarted later on at no additional cost. This problem type has been investigated in literature as an option to reduce the total RCPSP project lead-time. Kaplan (1988, 1991) was the first to study the PRCPSP and presented a solution procedure. However, Demeulemeester and Herroelen (1996a) have found an error in one of her theorems, and presented a correct optimal algorithm for the PRCPSP. Their computational results revealed that activity pre-emption has only a small positive effect on the lead-time of a project schedule. However, Ballestin et al. (2006) show that their heuristic procedure is able to produce high-quality solutions more easily for the PRCPSP than for the RCPSP.

The pre-emptive resource-constrained project scheduling problem with fast tracking (PRCPSP-FT) includes both relaxations and has been proposed by Debels and Vanhoucke (2006). When projects are fast-tracked, it usually indicates the compression of a project schedule by doing certain activities in parallel that would normally be done in a sequence. Hence, fast-tracking violates the precedence relations between activities and implies activity execution at incomplete information. Debels and Vanhoucke (2006) have investigated the impact of within-activity fast tracking, which allows the execution of pre-emptive sub-parts of an activity in parallel. This fast tracking option removes precedence relations between sub-parts of pre-empted activities and increases the number of execution scenarios.

Demeulemeester and Herroelen (1996a) have shown that the PRCPSP can be transformed into an RCPSP network by constructing a sub-activity network that splits each activity i into d_i sub-activities i_s with a duration $d_{i_s} = 1$ and a resource requirement $r_{i_s k} = r_{ik}$. Debels and Vanhoucke (2006) have taken a similar approach for the PRCPSP-FT where all precedence relations between sub-activities i_s of a similar activity i have been removed. Hence, the PRCPSP-FT assumes pre-emptive activities with fixed durations, which results in d_i non-related sub-activities with a duration $d_{i_s} = 1$ and a resource requirement $r_{i_s k} = r_{ik}$.

Demeulemeester and Herroelen (1996a) state that activity-pre-emption seldom has a huge impact on the total project lead-time compared to the RCPSP lead-time. However, Debels and Vanhoucke, (2006) has shown that activity pre-emption *and* activity fast-tracking of pre-empted sub-parts of activities can lead to large lead-time reductions. In the current manuscript, we study the PRCPSP-FT where we only allow activity pre-emption and within-activity fast tracking at an extra setup cost. Hence, our defined activity durations consist of both a setup and a processing time. The setup time component includes activity preparations such as equipping, resetting, changing, positioning, cleaning and warming up (Mika et al., 2006). This setup time is added to the total duration each time activity pre-emption and/or fast tracking occurs. In our problem-statement we assume an activity-dependent setup time t_i that needs to be added to the sub-activity duration at the initial start of the activity as well as for each time the activity is interrupted. The idea of setup time incorporation in project scheduling is not new. Kaplan (1991) has studied a similar approach for the PRCPSP. However, she assumes that a setup time is only required between activity interruptions and not for the initial start-up of an activity. To that purpose, she splits each activity i in d_i sub-activities and shows in a theorem that activity pre-emption is never beneficial for the first $t_i + 1$ sub-activities of each activity i. In our problem formulation we include this theorem in our problem-definition by assuming that each activity automatically requires a setup time from the moment it is started. Note that Demeulemeester and Herroelen (1996b) argue that possibly setup times before the starting of an activity are allowed to overlap with the processing of predecessor activities. For simplicity reasons, we do not add such an overlap to our problem description, although this could be included rather easily by adding minimal (negative) time-lags to the finish-start precedence relations (i, j), corresponding to the setup time of the end node activity j.

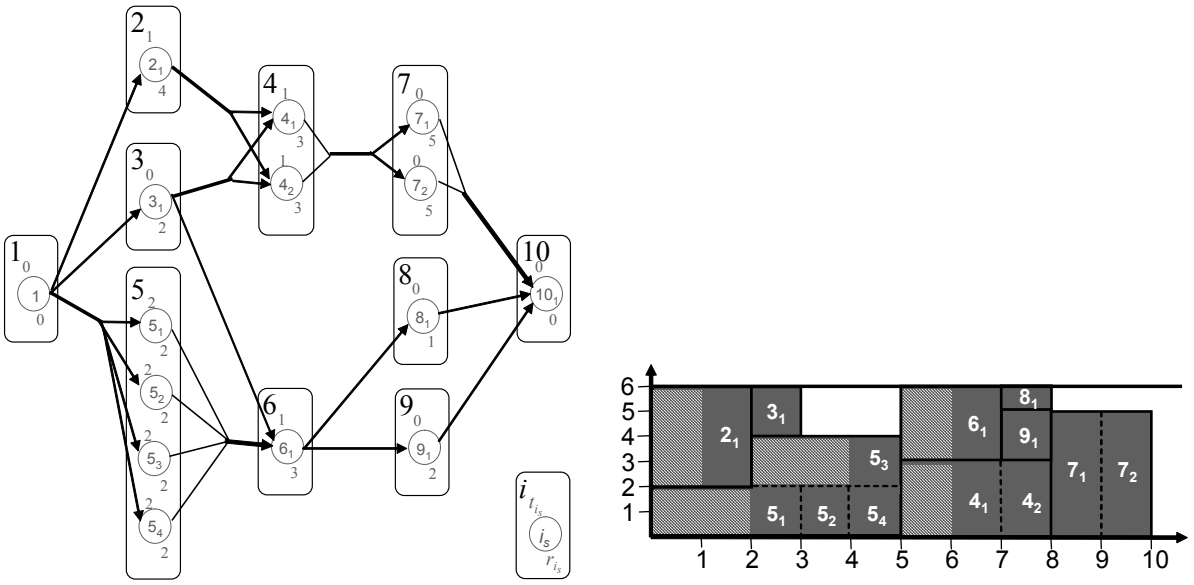

Figure 2. The example sub-activity network
and optimal schedule for the PRCPSP-FT with setup times

Figure 2 displays the sub-activity network for the PRCPSP-FT with a corresponding optimal schedule for the PRCPSP-FT with setup times. We assume the setup times $t_2 = t_4 = t_6 = 1$ and $t_5 = 2$ while all other activities can be pre-empted without extra setup. These setup times have been displayed above the node while resource requirements $r_{i_s 1}$ are shown below the node. All sub-activity durations d_{i_s} are equal to 1 and have not been displayed in the network. Note that we have carefully selected the number of sub-activities i_s for each activity i (denoted by $nrsa_i$), in order to allow a fair comparison with the RCPSP schedule of figure 1, as $nrsa_i = d_i - t_i$. Consequently, the number of sub-activities of activity 2 equals 1. Hence, since the total duration of activity 2 is not pre-empted (and not fast tracked) there is no difference with the RCPSP (i.e. the total duration d_i is equal to 2). However, the number of sub-activities for activity 5 is equal to $nrsa_5 = 4$. In doing so, the activity can be scheduled without pre-emption, resulting in a total duration of $t_5 + nrsa_5 = 2 + 4 = 6$ which is similar to the RCPSP duration d_5 of figure 1. In figure 2, this activity is pre-empted resulting in a total duration of 8 time units instead of 6, due to the extra setup time before subactivity 5_3. The activity labels have been shown in black, while the sub-activity numbers have been displayed in white. The shaded areas represent the use of resources due to setup times. Despite the pre-emptive setup times, the optimal lead-time could be decreased from 11 to 10 time units.

The outline of the paper is as follows. In section 2, we propose a branch-and-bound approach for the PRCPSP-FT with set-up times, which relies on the branching scheme of Demeulemeester and Herroelen (1992). Section 3 presents some specific adaptations to our branch-and-bound algorithm to efficiently cope with setup times. In section 4, the problem example of figure 1 is solved as an illustration. In section 5, we report extensive computational results. We conclude in section 6 with some overall conclusions and suggestions for future research.

2 The general branch-and-bound approach

In this section, we explain the branching scheme of our branch-and-bound procedure, which is an adapted version of the depth-first procedure of Demeulemeester and Herroelen (1992) in order to cope with pre-emptive fast tracking of sub-activities with setup times. To that purpose, we rely on the following definitions:

i_s = sub-activity s of activity i ($s = 1, \ldots, nrsa_i$)
dm = current time instant in our search tree (the decision moment)
S = set of active sub-activities at time instant dm
PS = partial schedule

D^p = set of delaying alternatives at level p of the branch-and-bound tree

D_q^p = delaying alternative q at level p of the branch-and-bound tree ($D_q^p \subset D^p$)

$E_1 \cup E_2$ = the eligible set of activities that can be scheduled at time instant *dm*. This eligible set is divided into two disjoint subsets E_1 and E_2, as follows:

E_1 = set of eligible sub-activities that can be scheduled without setup time at decision moment *dm*

E_2 = set of remaining sub-activities that can only be scheduled with an extra setup time

Hence, the set E_1 contains eligible sub-activities i_s that can be scheduled immediately after the finish of a sub-activity $i_{s'}$. If more than one sub-activity i_s can be scheduled after the finish of sub-activity $i_{s'}$, priority is given to the lowest numbered sub-activity of activity i. Figure 3 displays the sets *PS*, *S*, E_1 and E_2 at decision moment *dm* = 3. Activity 5_3 can be scheduled after activity 5_1 without any setup time, and enters E_1. Therefore, activity 5_4, which has a higher subscript, can not be scheduled after 5_1. All remaining sub-activities (4_1, 4_2 and 5_4) can only be scheduled with an extra setup time, and belong to the set E_2.

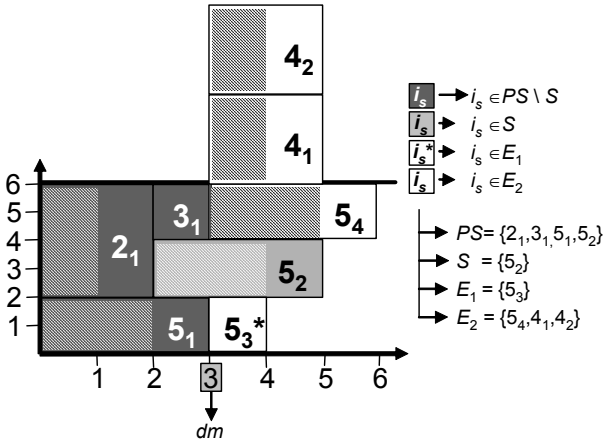

Figure 3. The sets *PS*, *S*, E_1 and E_2 at decision moment *dm*

The depth-first approach builds up partial schedules starting at time 0 and continuing systematically throughout the search process by iteratively adding sub-activities until a complete feasible schedule is obtained. A partial schedule at level p of the search tree will be continued by determining the next decision moment *dm* at which unscheduled activities might start. All unscheduled activities which are a candidate to start at time *dm* are calculated and collected in the set of eligible activities. In order to take the setup times into account, the eligible set is splitted into two disjoint subsets E_1 and E_2 as described earlier. The previously scheduled but at *dm* unfinished activities belong to the set *S* of activities in progress. If scheduling all activities from $E_1 \cup E_2 \cup S$ at *dm* would cause a resource conflict, the procedure starts to branch to the next level $p + 1$ and delays subsets (delaying alternatives D_q^p) of $E_1 \cup E_2 \cup S$ to resolve resource conflicts. The selection of a delaying alternative involves

that only the unselected activities of $E_1 \cup E_2 \cup S$ will be scheduled at *dm* while all previously scheduled activities of S and the activities of $E_1 \cup E_2$ that belong to the alternative are postponed. This process is repeated until a feasible schedule is found, followed by a backtracking mechanism and the algorithm continues as a usual branch-and-bound procedure.

The basic algorithmic steps of the branch-and-bound branching scheme can be summarized along the following 5 steps. In section 3, we explain the bold indicated sub-steps 3.2, 3.4 and 4.4 into detail.

Step 1. Initialisation

1.1 Set the upper bound *UB* on the project duration at ∞
1.2 Set the level *p* of the branch-and-bound tree at 0
1.3 Initialise the decision moment *dm* at -1
1.4 Schedule the dummy start sub-activity: $f_1 = 0$, $PS = \{1\}$ and $S = \{1\}$

Step 2. Increase the decision moment *dm*

2.1 If the dummy end sub-activity has been scheduled, update the upper bound *UB* and go to Step 5 (Backtracking)
2.2 Update the decision moment: $dm = \min_{i_s \in S} f_{i_s}$
2.3 Update the set of sub-activities in progress S and the set of eligible sub-activities E_1 and E_2, as follows: $\forall i_s \in S | f_{i_s} = dm$
- Update $S = S \setminus \{ i_s \}$
- Update E_1 and E_2 by including all successor sub-activities of i_s for which all predecessor sub-activities are element of $PS \setminus S$
  - E_1 contains these sub-activities which can be scheduled immediately after a sub-activity without pre-emptive setup time at *dm*. If there are more possibilities, the lowest indexed sub-activities are put in E_1.
  - E_2 contains the remaining sub-activities

2.4 Store the decision moment *dm*, the sets *PS*, E_1, E_2 and S and the finishing times f_{i_s} of each sub-activity $i_s \in S$ at level *p* of the search tree.

Step 3. Determine the minimal delaying alternatives

3.1 Calculate the excessive resource use c_k for each resource $k \in R$ when all sub-activities of $S \cup E_1 \cup E_2$ are scheduled at *dm*: $c_k = \sum_{i_s \in S \cup E_1 \cup E_2} r_{i_s k} - a_k$
3.2 **Theorems 1/2/3/5 and property 1:** Define the set of minimal delaying alternatives $D^p = \{ D_q^p \subset (S \cup E_1 \cup E_2) \mid \sum_{i_s \in D_q^p} r_{i_s k} \geq c_k$ for each $k \in R$ and there exists no other $D_{q'}^p \in D^p$ for which $D_{q'}^p \subset D_q^p \}$
3.3 Fathom all minimal delaying alternatives D_q^p that satisfy the conditions of **theorem 4**
3.4 $\forall D_q^p \in D^p$: if $LB(D_q^p) \geq UB$, $D^p = D^p \setminus \{ D_q^p \}$
3.5 If $D^p = \varnothing$: go to step 5

Step 4. Select the next minimal delaying alternative

4.1 Select the minimal delaying alternative $D_b^p \in D^p$ with the lowest lower bound $LB(D_b^p)$ and update $D^p = D^p \setminus \{ D_b^p \}$

4.2 If $LB(D_b^p) \geq UB$: Set $D^p = \varnothing$ and go to step 5

4.3 Schedule the sub-activities $i_s \notin D_b^p$ as follows:

- $\forall\, i_s \in ((E_1 \cup E_2) \setminus D_b^p)$
  - $PS = PS \cup \{ i_s \}$, $S = S \cup \{ i_s \}$
  - If $i_s \in E_1$: $f_{i_s} = dm + 1$ and $E_1 = E_1 \setminus \{ i_s \}$
  - If $i_s \in E_2$: $f_{i_s} = dm + t_i + 1$ and $E_2 = E_2 \setminus \{ i_s \}$
- $\forall\, i_s \in (S \cup D_b^p)$
  - Remove i_s from S and PS: $PS = PS \setminus \{ i_s \}$, $S = S \setminus \{ i_s \}$
  - Add i_s to the eligible set $E_1 \cup E_2$

4.4 **Property 2:** Check whether there are sub-activities in progress at dm that have been scheduled with a pre-emptive setup time before time instant dm: if this sub-activity can be scheduled with pre-emptive setup time at time instant dm, remove its setup time and change its finishing time to $dm + 1$

4.5 Update the branching level of the search tree: $p = p + 1$

4.6 Go to step 2

Step 5. Backtracking

5.1 Update the branching level of the search tree: $p = p - 1$

5.2 If the branching level $p < 0$, then STOP

5.3 If $D^p = \varnothing$: repeat Step 5

5.4 Restore the decision moment dm, the sets PS, E_1, E_2 and S and the finishing times f_{i_s} of each sub-activity $i_s \in S$ at level p of the search tree

5.5 Go to step 4

3. Specific adaptations to the PRCPSP-FT with set-up times

In this section, we discuss all dominance rules and lower bound calculations used in the branch-and-bound procedure. Most of them are modified versions of dominance rules and lower bounds presented in Demeulemeester and Herroelen (1992) (further abbreviated as DH92), to cope with setup-times. In section 3.1, we discuss two properties that will be used throughout the remainder of this section. In section 3.2, the various dominance rules are explained in detail. Section 3.3 discusses the calculations of the lower bounds.

3.1 Dominance and lower bound properties

In this section, we briefly explain two algorithmic details that will be used throughout the remainder of our manuscript. They avoid conflicts between dominance rules of section 3.2 and simplify the lower bound descriptions of section 3.3.

Property 1. Sub-activities i_s of each set (S, E_1 or E_2) can only be evaluated as potential candidates for a minimal delaying alternative if all other sub-activities $i_{s'}$ from these sets with a higher numbered subscript $s' > s$ have been evaluated first.

Note that we previously mentioned that entrance of sub-activities in the sets E_1 or E_2 is done according to the lowest numbered subscripts. Possible entrance of these sub-activities into the minimal delaying alternatives is done according to the highest numbered subscripts. In doing so, the algorithm guarantees that only sub-activities are added to the partial set PS when all lower numbered sub-activities already belong to that set.

Property 2. When a sub-activity is moved from the set E_2 to the set S at decision moment dm, the corresponding setup time can still be removed from that sub-activity at a later decision moment $dm' > dm$.

Property 2 removes setup times when a sub-activity $i_s \in S$, that has been scheduled pre-emptively at decision moment dm, can be rescheduled at $dm' > dm$ without pre-emptive setup time immediately after another sub-activity $i_{s'}$ of the set PS. In this case, the initial setup time of sub-activity i_s has become superfluous and can be removed. Both properties will be used throughout the remained of our manuscript, and will be illustrated during the description of the dominance rules and lower bound calculations.

3.2 Dominance rules

In this section, we review all dominance rules of DH92 and adapt them to cope with the PRCPSP-FT with setup times (note that we number these dominance rules from 1 to 4 to be in line with the original paper). Moreover, we extended the branching scheme with a new theorem 5 to further reduce the number of nodes in the branch-and-bound tree.

Demeulemeester and Herroelen (1992) have described two special cases in order to put eligible activities in progress, which have been adapted in the theorems 1 and 2 to cope with setup times, as follows:

Theorem 1 (Put eligible sub-activity i_s in progress): If no activities are in progress at decision moment *dm* of the current branching node and an eligible sub-activity i_s cannot be scheduled together with any other unscheduled sub-activity at any time instant $dm' \geq dm$, then there exists an optimal continuation of the partial schedule with the eligible sub-activity i_s started at *dm*. However, when there exists a non-empty set of sub-activities $K \subset (E_1 \setminus \{ i_s\})$ that contains sub-activities *k'* with $t_{k'} > 0$, then the algorithm also needs to consider all minimal delaying alternatives which do not contain all sub-activities of *K*.

Theorem 2 (Put eligible sub-activities i_s and j_s in progress): If no activities are in progress at decision moment *dm* of the current branching node and an eligible sub-activity i_s can only be scheduled concurrently with one other unscheduled sub-activity $j_s \in (E_1 \cup E_2)$ at any time instant $dm' \geq dm$ such that j_s would not finish later than i_s if both are started at *dm*, then there exists an optimal continuation of the partial schedule in which both sub-activities i_s and j_s are put in progress at time *dm*. However, when there exists a non-empty set of sub-activities $K \subset (E_1 \setminus \{i_s, j_s\})$ that contains sub-activities *k'* with $t_{k'} > 0$, then the algorithm also needs to consider all minimal delaying alternatives which do not contain all sub-activities of *K*.

Theorems 1 and 2 do not differ much from the original versions of DH92. However, the incorporation of setup times has resulted in an adaptation of the theorems when the set E_1 contains sub-activities different from i_s (theorem 1) or $\{i_s, j_s\}$ (theorem 2). These sub-activities can be scheduled without a setup time at time instant *dm* (element of E_1), while delaying these sub-activities will lead to the scheduling at a later time instant with an extra setup time. However, if the algorithm schedules at least one of these sub-activities of E_1, we might prevent a setup time of the remaining sub-activities of E_1 at a later time instant.

Figure 4 displays an illustrative schedule for theorem 1, where the resource availability has been decreased to 5 units. In this schedule, sub-activity 7_1 cannot be scheduled together with any other unscheduled sub-activity. However, there exists a non-empty set of sub-activities $K \subset (E_1 \setminus \{7_1\})$ that contains sub-activity 5_3 with $t_5 = 2$. The algorithm also needs to consider all minimal delaying alternatives (see theorem 3) which do not contain all sub-activities of K and hence the set of minimal delaying alternatives is equal to $\{(7_1, 7_2), (5_3, 5_4, 7_2)\}$ instead of only $\{(5_3, 5_4, 7_2)\}$.

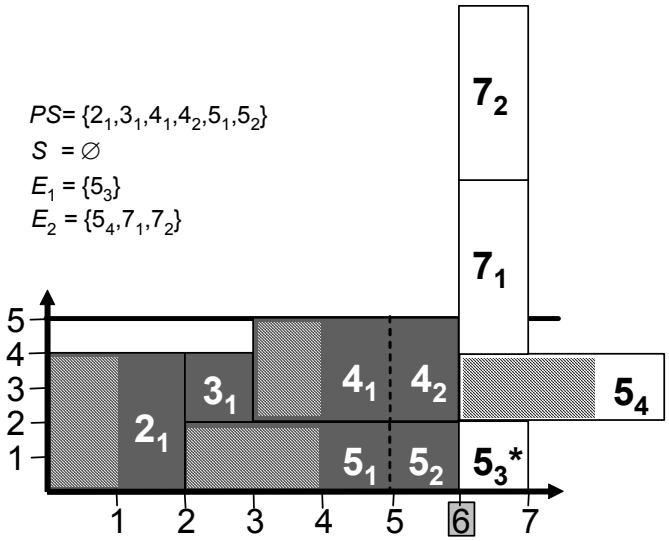

Figure 4. An illustrative partial schedule for theorem 1
(Note that – for illustrative purposes – the resource availability has been changed to 5)

The partial schedule of figure 5 illustrates theorem 2. Sub-activity 7_1 can only be scheduled in parallel with sub-activity 8_1 at any time instant $dm' \geq 8$. Moreover, both sub-activities would finish at time instant 9 if started at $dm = 8$. Since $E_1 = \varnothing$ there is no non-empty set of sub-activities K. Hence, it is sufficient to consider only the minimal delaying alternative $D_q^6 = \{7_2, 9_1\}$ such that sub-activities 7_1 and 8_1 are both scheduled at $dm = 8$.

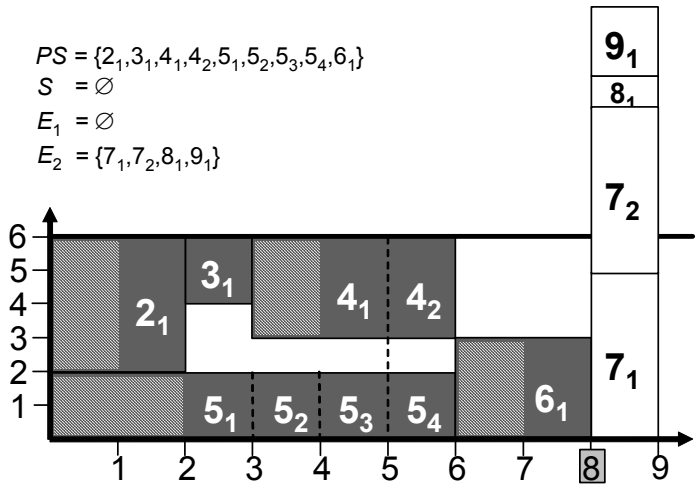

Figure 5. The partial schedule at node 7 of level 6 of the branch-and-bound tree

Theorem 3 (Minimal Delaying Alternatives): In order to define the set of minimal delaying alternatives D_q^p for the PRCPSP-FT with setup times, it is sufficient to define the number of sub-activities e_i for each activity i that should be chosen from the eligible set $E_1 \cup E_2 \cup S$

This theorem has been proposed by Debels and Vanhoucke (2006) for the PRCPSP-FT without setup times as an extension of the minimal delaying alternatives principle of DH92. However, in order to cope with setup times, we implemented a specific ranking between activities of sets E_1, E_2 and S in order to standardize the selection of activities for each set. More precisely, if e_i sub-activities of activity i need to be chosen for a delaying alternative, the algorithm always give priority to sub-activities of set E_2, then to sub-activities of E_1 and finally to sub-activities of S. Hence, the algorithm prefers to delay sub-activities that need to be scheduled with an extra setup time at the decision moment *dm*, followed by sub-activities that start at the decision moment *dm* without setup. The sub-activities of set S have the lowest priority to be delayed since these sub-activities start earlier than the decision moment *dm*, and hence, delaying these sub-activities would release resources at time units earlier than *dm*, which can not be re-filled by other unscheduled activities at a later point in time. Note that within each set, priority has been given to the sub-activities with the highest sub-activity number (see property 1).

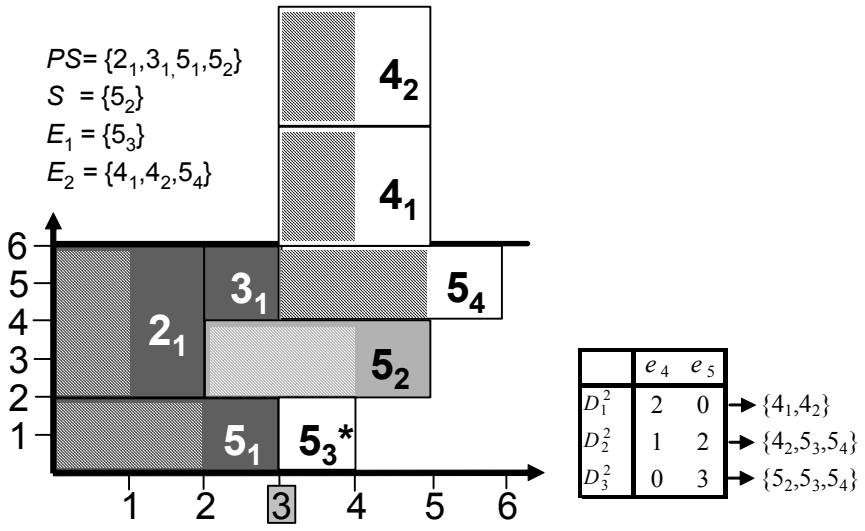

Figure 6. The partial schedule at node 3 of level 2 of the branch-and-bound tree and the corresponding minimal delaying alternatives

In Figure 6 we display the partial schedule of node 3 of the branch-and-bound tree of figure 10. The decision moment *dm* is equal to 3 and the candidate sets to build the delaying alternatives are equal to $S = \{5_2\}$, $E_1 = \{5_3\}$ and $E_2 = \{5_4, 4_1, 4_2\}$. Thanks to theorem 3, the algorithm only needs to determine the number of sub-activities e_4 and e_5 to be selected in the delaying alternatives. Each (e_4, e_5)

combination can be transformed into a minimal delaying set following the priority rules described above. As an example, when $e_4 = 2$ and $e_5 = 0$, the minimal delaying alternative is equal to $D_1^2 = \{4_1, 4_2\}$. The minimal delaying alternative for $e_4 = 1$ and $e_5 = 2$ is equal to $D_2^2 = \{4_2, 5_3, 5_4\}$ and hence no other alternatives (e.g. $\{4_1, 5_3, 5_4\}$) need to be considered.

Figure 7 displays the resulting partial schedule under the assumption that the minimal delaying alternative D_2^2 has been chosen. This means that sub-activities 4_1 and 5_2 have been scheduled at decision moment $dm = 3$ while sub-activities 5_3 and 5_4 have been delayed. However, note that the setup time of sub-activity 5_2 needs to removed (property 2) since this sub-activity has been originally scheduled at time instant 2 with a setup time of 1, but can now also be considered as a non-pre-emptive successor of sub-activity 5_1 and hence, no setup time is required.

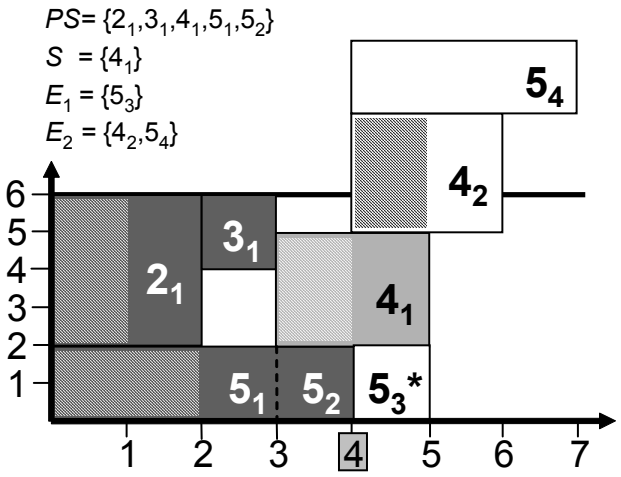

Figure 7. The partial schedule at node 4 of level 3 of the branch-and-bound tree

The fourth theorem is based on the well-known cutset dominance rule of DH92. A cutset C^x at node x of the branch-and-bound tree with a corresponding decision moment dm^x is defined as the set of unscheduled sub-activities for which all predecessor sub-activities belong to the partial schedule PS^x. Furthermore, we refer to E_1^x, E_2^x and PS^x as the eligible sets E_1 and E_2 and the partial set PS at node x of the branch-and-bound tree.

Theorem 4 (Cutset Dominance Rule): Consider a cutset C^y which contains the same sub-activities as a cutset C^x, which was previously saved during the search of another path in the search tree.
If

- $dm^x \leq dm^y$

- All sub-activities in progress at time dm^x do not finish later than the maximum of dm^y and the finish time of the corresponding sub-activities in PS^y
- All sub-activities $i_s \in (E_2^x \cap E_1^y)$ do not finish earlier if scheduled at dm^y in node y then if scheduled at dm^x in node x

then the current partial schedule PS^y is dominated.

This dominance rule differs from the DH92 cutset dominance rule in the third condition. All sub-activities $i_s \in E_1^y$ can be scheduled without setup time at dm^y in node y while all sub-activities $i_s \in E_2^x$ can only be scheduled with an extra setup time at dm^x in node x. Hence, since $dm^x \leq dm^y$, the $E_2^x \cap E_1^y$ sub-activities might finish later at node x (since a setup time is required) than when scheduled at node y (without a setup time) such that PS^y can not be dominated by PS^x. Hence, the extra restriction that these sub-activities $E_2^x \cap E_1^y$ may not finish earlier when scheduled at dm^y (node y) than when scheduled at dm^x (node x) is necessary to conclude that the set PS^y can be dominated.

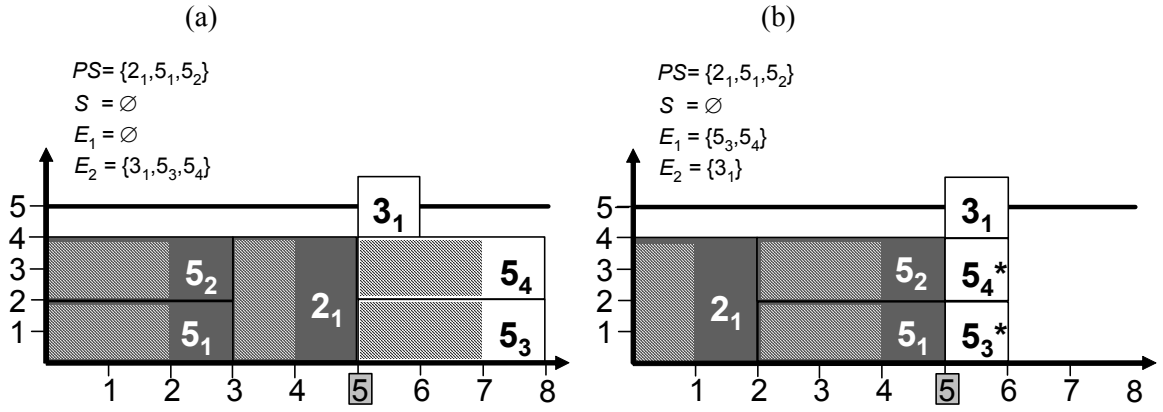

Figure 8. An illustrative partial schedule for theorem 4
(Note that – for illustrative purposes – the resource availability has been changed to 5)

Figure 8 displays two illustrative schedules where the resource availability has been decreased to 5 units. If we ignore the third condition, schedule (a) is dominated by schedule (b) and vice versa. However, the third condition clearly illustrates that schedule (b) is no longer dominated by schedule (a) since the sub-activities $E_2^a \cap E_1^b = \{5_3, 5_4\}$ finish earlier in schedule (b) than in schedule (a).

In order to improve performance, we have added one extra dominance rule to cope with the PRCPSP-FT with setup times. To that purpose, we define the earliest finishing moment efm_i of each activity i as the minimum of the finishing times of all its sub-activities i_s that belong to the sets $E_1 \cup E_2 \cup S$. This theorem automatically selects sub-activities from E_2 that must belong to each minimal delaying alternative, as follows:

Theorem 5 (Deviation from the minimal delaying alternatives principle): If there is a sub-activity $i_s \in E_2$ for which $s \geq efm_i + d_i - dm - 2t_i$ then all sub-activities (i_s, i_{s+1},…, $i_{d_i - t_i}$) should belong to each minimal delaying alternative.

The formula $s \geq efm_i + d_i - dm - 2t_i$ is a simplification of the form $d_i - t_i - s + 1 \leq dm + t_i + 1 - efm_i$ and reads as follows: $d_i - t_i - s + 1$ is equal to the number of sub-activities of activity i with a subscript higher than or equal to s (i.e. i_s, i_{s+1},…, $i_{d_i - t_i}$). $dm + t_i + 1$ is equal to the finishing time of sub-activity i_s when scheduled at decision moment dm. Consequently, $dm + t_i + 1 - efm_i$ is the difference between the finishing time of sub-activity i_s when scheduled at dm and the next time instant efm_i at which sub-activity i_s can be scheduled without pre-emptive setup time. When this difference is larger than the number of unscheduled sub-activities $d_i - t_i - s + 1$, then scheduling all these sub-activities in series at decision moment efm_i will result in a finishing time smaller than $dm + t_i + 1$ and hence, these sub-activities should be delayed.

Theorem 5 can be illustrated on sub-activity 5_4 of the partial schedule of figure 6. Since $s \geq efm_5 + d_5 - dm - 2t_5$ ($4 \geq 5 + 6 - 4 - 2 * 2$), it is beneficial to delay sub-activity 5_4 such that it can be scheduled later without pre-emptive setup time (immediately after sub-activity 5_3). Consequently, sub-activity 9 needs to be added in every delaying alternative, and D_1^2 is extended to $\{4_1, 4_2, 5_4\}$.

3.3 Lower bound calculations

In this section, we propose two lower bounds that are calculated for each delaying alternative at every node in the tree. When the current best solution is lower than the calculated lower bound, the minimal delaying alternative can be pruned from the set of delaying alternatives.

Critical path based lower bound: $LB_0(D_q^p)$**:** This lower bound is calculated as next decision moment if D_q^p is selected, increased by the maximum remaining critical path length L_{i_s} of all sub-activities $i_s \in D_q^p$, i.e. $LB_0(D_q^p) = dm_{next}(D_q^p) + \max_{i_s \in D_q^p}(L_{i_s})$. The first term $dm_{next}(D_q^p)$ is equal to the earliest finishing time of all sub-activities $i_s \notin D_q^p$ in progress at dm while the L_{i_s} is calculated by straightforward forward calculations where each unscheduled sub-activity i_1 is increased by its setup time t_i.

Resource based lower bound: $LB_r(D_q^p)$: The presence of fast tracking in resource-constrained project scheduling leads to an efficient use of resources, as illustrated by Debels and Vanhoucke (2006). Hence, the use of efficient resource-based lower bounds could dramatically improve the performance of our branch-and-bound algorithm. Therefore, the algorithm calculates a lower bound for each minimal delaying alternative $LB_r^k(D_q^p) = dm + \max_k \left\lceil \frac{extra^k}{a_k} \right\rceil$ where $extra^k$ can be calculated as the sum of:

1) Minimal resource use after *dm* by all unscheduled sub-activities $i_s \notin PS \setminus S$

$$\sum_{i_s \notin PS \setminus S} r_{i_s k}$$

This measures the minimal (i.e. excluding setup time) resource use of all sub-activities that will be scheduled after the decision moment *dm*.

2) Unused resource units due to the minimal setup time of sub-activities

$$\sum_{i_1 \notin PS} t_i * r_{i_1 k} + \sum_{i_1 \in S} (f_{i_1} - 1 - dm) * r_{i_1 k} + \sum_{\substack{i_s \in F_1 \\ s>1}} t_i * r_{i_s k} + \sum_{\substack{i_s \in F_2 \\ s>1}} (f_{i_s} - 1 - dm) * r_{i_s k}$$

The first and second terms calculate the minimal setup time of all initial sub-activities i_1. The first term incorporates the complete setup time for these sub-activities that are neither in progress nor in the partial set. The second term incorporates the minimal remaining setup time $f_{i_1} - 1 - dm$ at *dm* of all initial sub-activities that are in progress at *dm* (only part of the setup time after *dm* needs to be taken into account).

F_1 is used to denote the set of unscheduled sub-activities $i_s \in E_1 \cup E_2 \cup D_q^p$ for which all lower labelled sub-activities are already finished at (if $i_s \in E_1 \cup D_q^p$) or before (if $i_s \in E_2$) decision moment *dm*. These sub-activities can only be scheduled with a pre-emptive setup time. Note that, thanks to property 1, the higher labelled sub-activities can be left out of consideration.

The set F_2 ($\subset S$) contains the sub-activities that are in progress at *dm* and for which all lower labelled sub-activities of the same activity finish before *dm*. These sub-activities can only be scheduled with a pre-emptive setup time at *dm* or later (and hence, the minimal remaining setup time at *dm, i.e.* $f_{i_s} - 1 - dm$ *,* needs to be incorporated)

Assume a project schedule for our example project as displayed in figure 9. The resource based lower bound $LB_r^k(D_q^p) = 5 + \left\lceil \frac{extra^1}{6} \right\rceil$ = 10 with $extra^1$ equal to 28 as given in the right part of figure 9

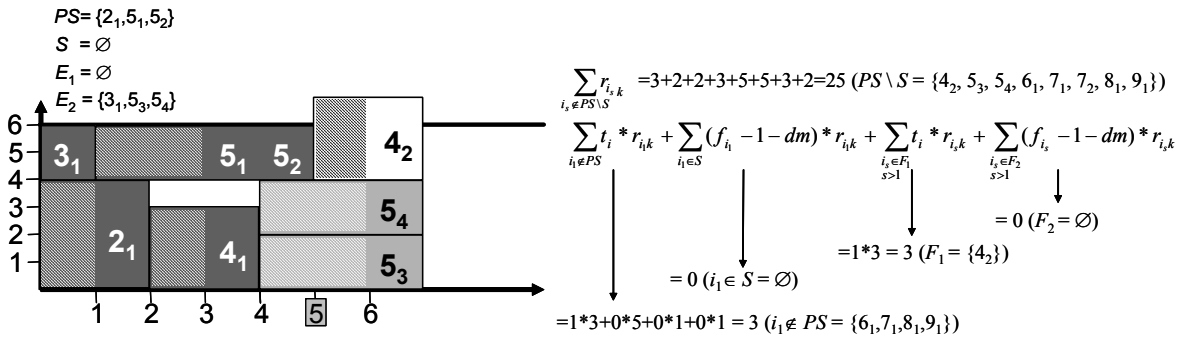

Figure 9. The resource based lower bound for the example project at time instant 5

4 An example branch-and-bound tree

Figure 10 displays the branch-and-bound tree for the example project of figure 2 consisting of 21 nodes. The algorithm starts at level 0 with an empty partial set at $dm = 0$ and calculates D^0 that consists of four minimal delaying alternatives $D_1^0 = \{3_1, 5_2, 5_3, 5_4\}$, $D_2^0 = \{5_1, 5_2, 5_3, 5_4\}$, $D_3^0 = \{2_1, 5_3, 5_4\}$ and $D_4^0 = \{2_1, 3_1, 5_4\}$. The algorithm first selects the alternative $D_1^0 = \{3_1, 5_2, 5_3, 5_4\}$ with the lowest lower bound $LB(D_1^0) = 9$, and schedules sub-activities 2_1 and 5_1 resulting in the partial schedule of node 1. The algorithm continues this way and finds the first complete schedule at node 10. This solution is saved and the upper bound is updated to 11, followed by a backtracking step towards level 2 of the tree. The algorithm then selects D_2^2 from the two remaining delaying alternatives $D_2^2 = \{4_1, 4_2, 5_4\}$ and $D_3^2 = \{5_2, 5_3, 5_4\}$ and continues. The optimal solution can be found at node 16 with a total project lead-time of 10.

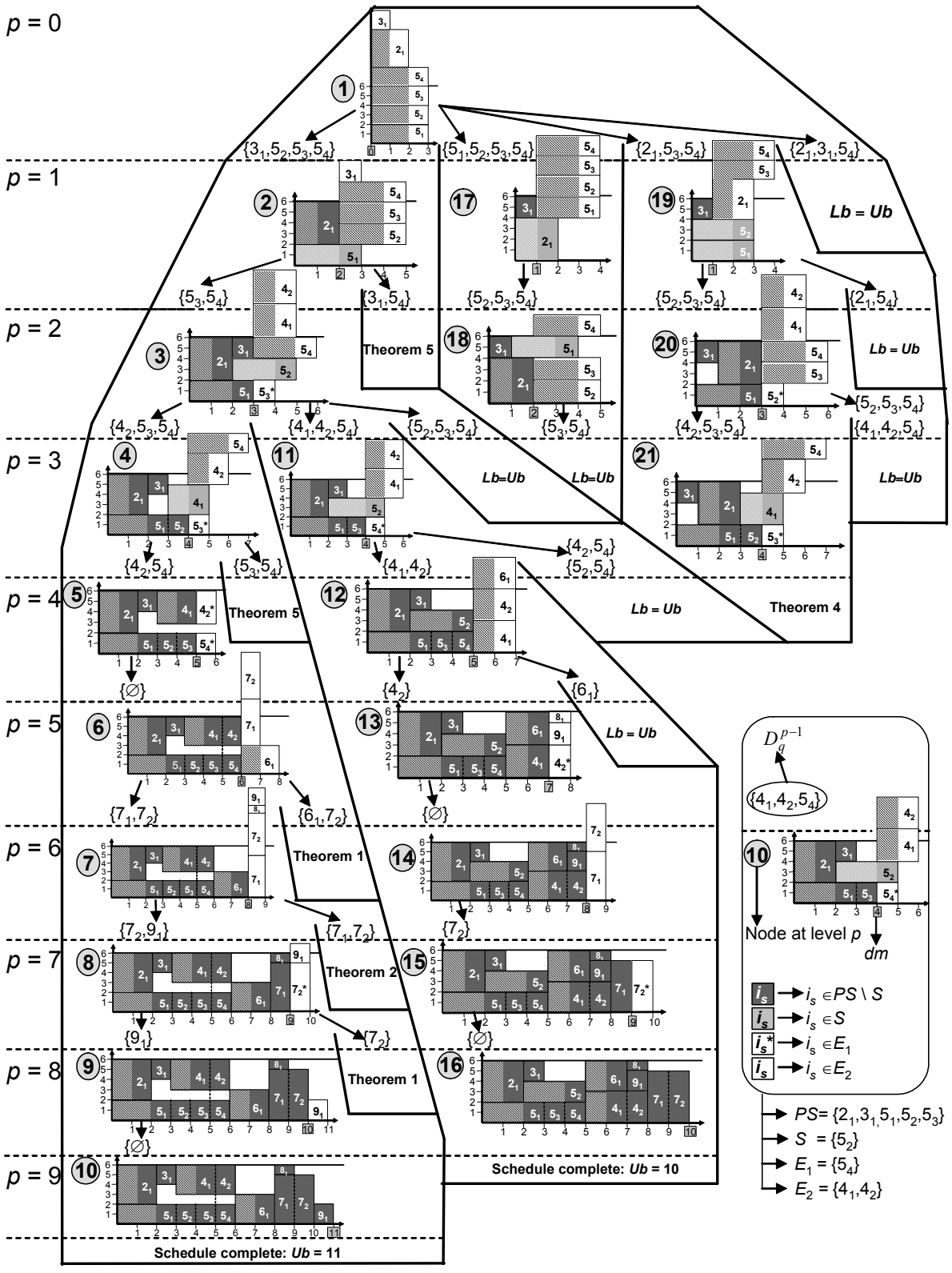

Figure 10. The branch-and-bound tree for the example project of figure 2

5 Computational results

In order to test the performance of our branch-and-bound procedure as well as the impact of pre-emptive fast tracking with setup times on the lead-time, we test the algorithm on the 1,920 problem instances generated by RanGen (Demeulemeester et al., 2003) and presented by Debels and Vanhoucke (2006). The number of non-dummy activities ($n - 2$) has been set at 10, 12, 14, 16, 18 and 20 with an order strength OS (Mastor, 1970) and a resource-constrainedness RC (Patterson, 1976) fixed at 0.2, 0.4, 0.6 and 0.8. All project instances require a single resource type with an availability of 10 units. The activity durations have been chosen randomly between 1 and 5. Since each setting contains 20 problem instances, the problem set contains 6 * 4 * 4 * 5 * 20 = 1920 network instances. To compare the resulting schedules with the optimal RCPSP schedules, we have assumed that part of the activity durations can be considered as the unavoidable setup time before the initial subactivity. We consider, without loss of generality, activity-independent setup times (i.e. the setup times t_i are the same for each activity) under 5 settings $t_i = 0$, 1, 2, 3 or 4. Consequently, the activity setup times and remaining activity durations have been calculated as follows: we subtract the generated setup time (0, 1, 2, 3 or 4) from the original activity duration to calculate the remaining activity duration. In case that the generated setup time setting is larger than the original activity duration, we reduce the setup time for that activity to the original duration - 1, such that the remaining duration equals 1. Hence, the sum of the activity setup time and its remaining duration is always equal to the original duration of the project network instance. This approach allows us to measure the impact of pre-emptive fast tracking with setup times on the schedule quality and leads to 3 different scenario's, as follows:

1) If the setup time of each activity is set at $t = 0$, then the remaining duration for each activity is equal to the duration of the RCPSP instances. Since there are no setup times, the problem boils down to the PRCPSP-FT described in Debels and Vanhoucke (2006).
2) If the setup times for the activities are set at a value t between 0 and 4, then the remaining durations of the activities lie between 1 and $5 - t$. The minimal lead-time will lie between the RCPSP and the PRCPSP-FT minimal lead-time.
3) If the setup time of each activity is set at $t = 4$, then each remaining activity duration is equal to 1. In this case, activity pre-emption can never lead to a lead-time reduction, and hence, the problem boils down to the basic RCPSP.

Scenario	1+2	4	5		$t = 0$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
1				Avg.CPU	13.89	5.06	2.92	1.53	0.67
				%Opt	87%	96%	98%	99%	99%
2	X			Avg.CPU	11.24	4.47	2.56	1.30	0.35
				%Opt	90%	97%	98%	99%	100%
3		X		Avg.CPU	3.35	2.45	0.97	0.24	0.08
				%Opt	97%	98%	99%	100%	100%
4			X	Avg.CPU	14.01	4.84	2.63	1.36	0.68
				%Opt	87%	96%	98%	99%	99%
5	X	X		Avg.CPU	2.70	2.11	0.92	0.24	0.08
				%Opt	98%	98%	99%	100%	100%
6	X		X	Avg.CPU	11.24	4.25	2.33	1.15	0.36
				%Opt	90%	97%	98%	99%	100%
7		X	X	Avg.CPU	3.35	2.43	0.89	0.22	0.08
				%Opt	97%	98%	99%	100%	100%
8	X	X	X	Avg.CPU	2.67	1.95	0.87	0.20	0.08
				%Opt	98%	98%	99%	100%	100%

Table 1. Performance of the dominance rules of section 3.2

Table 1 displays the impact of each theorem on the performance of our branch-and-bound algorithm. To that purpose, we have tested 8 different scenarios, corresponding to various combinations of dominance rules. Note that we never excluded the minimal delaying alternative theorem (theorem 3) since it has a major beneficial effect on the performance of the solution procedure. Moreover, we have combined theorems 1 and 2 into one scenario since they can be considered as two versions of the same dominance principle. The columns of the table represent the different setup time settings, varying from 0 to 4. The row labeled "Avg. CPU" displays the average time (in seconds) needed to solve the problem instances and the row labeled "% Opt" displays the percentage of problem instances that have been optimally solved within a pre-specified time limit of 100 seconds.

The table reveals that all dominance rules have a positive effect on the performance of the algorithm. For setup times equal to 1, 2 or 3, the inclusion of a dominance rule always improves the performance, both in terms of computational effort and percentage of problems solved to optimality. The cutset dominance rule (theorem 4) seems to have the most beneficial effect on the performance, which is completely in line with literature. The size of the setup times has a clear and positive effect on the problem complexity. An increasing setup time results in a lower remaining activity duration and hence in a lower number of sub-activities. This results in a smaller search and reduces complexity of the problem instance.

Table 2 reports results for experiments on the performance of the lower bounds. The critical path based lower bound LB_0 and the resource based lower bound LB_r have been tested individually (scenarios 1 and 2) and in combination (scenario 3). Scenario 3 outperforms scenarios 1 and 2,

showing that both dominance rules have a positive effect on the performance of the algorithm. LB_r is more effective than LB_0, which confirms the results of Debels and Vanhoucke (2006) that fast tracking leads to schedules in which the available resources are used efficiently. This increases the importance of resource-based lower bounds.

Scenario	Lower bounds			$t = 0$	$t = 1$	$t = 2$	$t = 3$	$t = 4$
	LB_0	LB_r						
1	X		Avg.CPU	3.88	17.13	9.55	2.95	1.17
			%Opt	96%	85%	92%	98%	99%
2		X	Avg.CPU	2.69	2.46	2.00	0.87	0.44
			%Opt	98%	98%	98%	99%	100%
3	X	X	Avg.CPU	2.67	1.95	0.87	0.20	0.08
			%Opt	98%	98%	99%	100%	100%

Table 2. Performance of the lower bounds of section 3.3

Table 3 displays more detailed results for all problem instances with the different settings for the setup time and the number of project activities. Moreover, the solutions have been compared with the results obtained by the dedicated algorithms for the RCPSP (Demeulemeester and Herroelen, 1992), the PRCPSP (Demeulemeester and Herroelen, 1996a) and the PRCPSP-FT (Debels and Vanhoucke, 2006). The row labeled "Avg.improvement" measures the average decrease of the total project lead-time compared to the minimal RCPSP lead-time.

Non-dummy activities	10	12	14	16	18	20	Global
PRCPSP-FT with t = 0							
Avg.CPU	0.01	0.02	1.34	2.29	2.78	9.57	2.67
%Opt	100%	100%	99%	98%	98%	91%	98%
Avg.improvement	19%	17%	15%	14%	13%	11%	15%
PRCPSP-FT with t = 1							
Avg.CPU	0.00	0.03	0.07	1.93	1.66	7.98	1.95
%Opt	100%	100%	100%	98%	99%	93%	98%
Avg.improvement	12%	11%	9%	9%	8%	7%	9%
PRCPSP-FT with t = 2							
Avg.CPU	0.00	0.02	0.04	0.41	0.50	3.88	0.87
%Opt	100%	100%	100%	100%	100%	97%	99%
Avg.improvement	6%	6%	5%	5%	5%	4%	5%
PRCPSP-FT with t = 3							
Avg.CPU	0.00	0.00	0.01	0.02	0.03	1.13	0.20
%Opt	100%	100%	100%	100%	100%	99%	100%
Avg.improvement	3%	2%	2%	2%	2%	2%	2%
PRCPSP-FT with t = 4							
Avg.CPU	0.00	0.00	0.00	0.01	0.01	0.45	0.08
%Opt	100%	100%	100%	100%	100%	100%	100%
Avg.improvement	0%	0%	0%	0%	0%	0%	0%
RCPSP							
Avg.CPU	0.00	0.00	0.00	0.00	0.00	0.01	0.00
%Opt	100%	100%	100%	100%	100%	100%	100%
Avg.improvement	0%	0%	0%	0%	0%	0%	0%
PRCPSP							
Avg.CPU	0.00	0.00	0.02	0.06	0.57	3.20	0.64
%Opt	100%	100%	100%	100%	100%	97%	99%
Avg.improvement	0%	0%	1%	1%	0%	0%	1%
PRCPSP-FT							
Avg.CPU	0.00	0.00	0.05	0.22	1.22	3.96	0.91
%Opt	100%	100%	100%	100%	99%	97%	99%
Avg.improvement	19%	17%	15%	14%	13%	12%	15%

Table 3. Computational results for various problem types

The results in the table can be summarized as follows. First, the table confirms the results of table 1 that the size of the setup times has a positive effect on the problem complexity. The higher the value for the setup times, the less beneficial it is to pre-empt activities and hence, the closer the problem resembles to the basic RCPSP. Second, the table reveals that for t = 0 and t = 4, the dedicated procedures for the RCPSP and the PRCPSP-FT without setup times outperform our branch-and-bound procedure. If t = 0, the problem boils down to the PRCPSP-FT without setup times, that can be solved by the Debels and Vanhoucke (2006) procedure, leading to an average CPU-time reduction from 2.333 to 0.911 seconds. Likewise, if t = 4, the RCPSP instances that can be solved faster by the DH92 procedure, resulting in an average CPU-time decrease from 0.077 to 0.002 seconds. Last, it is worth mentioning that the option to fast track has a major effect on the project lead-time, even with high values for the setup times. As an example, the PRCPSP without setup times results only in an average

of 1% lead-time improvement, while the PRCPSP-FT with setup times of 3 still lead to an average improvement of 2%. This illustrates that setup times do not prevent the PRCPSP-FT to find schedule improvements. Hence, if technical restriction allow a within-activity fast tracking, even within the presence of relatively high setup costs, it is still beneficial to allow activity pre-emption as a technique to reduce the project lead-time.

6 Conclusions

The previous research of the preemptive resource-constrained project scheduling problem (PRCPSP) has shown that activity pre-emption drastically increases the problem complexity and might lead to only a small decrease in the total project lead-time. However, a recently studied pre-emptive extension, known as the pre-emptive resource-constrained project scheduling problem with fast tracking (PRCPSP-FT, Debels and Vanhoucke (2006)), allows that these pre-emptive sub-activities can be executed in parallel, and leads to a major decrease in the total project lead-time.

In this paper, we have extended the pre-emptive resource-constrained project scheduling problem with setup times and a fast tracking option between pre-emptive sub-parts of activities. We have presented a branch-and-bound procedure, based on the principles of the RCPSP procedure of Demeulemeester and Herroelen (1992), to cope with the new problem type and reported detailed computational experience.

Our experiments revealed that the incorporation of setup times further increases the complexity of the PRCPSP-FT. However, the improvement in the project lead-time, compared to the basic RCPSP, shows that the trade-off between problem complexity and the resulting schedule quality is worth investigating. Consequently, it lies in our future intensions to develop meta-heuristic procedures in order to solve more challenging and realistic problem instances where setup times can be incorporated when activities are pre-empted and these pre-emptive sub-activities can be fast tracked.

7 References

Ballestin, F., Valls, V., Quintanilla, S., 2006, Pre-emption in resource-constrained project scheduling, working paper, Universidad Publica de Navarra, Spain.

Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E., 1999. Resource-constrained project scheduling: notation, classification, models and methods, European Journal of Operational Research, 112, 3-41.

Debels, D., Vanhoucke, M., 2006, The impact of various activity assumptions on the lead-time and resource utilization of resource-constrained projects, working paper, Ghent University, Gent.

Demeulemeester, E., Herroelen, W., 1992, A branch-and-bound procedure for the multiple resource-constrained project scheduling problem, Management Science, 38 (12), 1803-1818.

Demeulemeester, E., Herroelen, W., 1996a, An efficient optimal solution procedure for the preemptive resource-constrained project scheduling problem, European Journal of Operational Research, 90, 334-348.

Demeulemeester, E., Herroelen, W., 1996b, Modelling setup times, process batches and transfer batches using activity network logic, European Journal of Operational Research, 89, 355-365.

Demeulemeester, E., Vanhoucke, M. and Herroelen, W., 2003. A random generator for activity-on-the-node networks. Journal of Scheduling, 6, 13-34.

Herroelen, W., De Reyck, B., Demeulemeester, E., 1998. Resource-constrained project scheduling: a survey of recent developments, Computers and Operations Research, 25 (4), 279-302.

Icmeli, O., Erenguc, S.S., Zappe, C.J., 1993. Project scheduling problems: a survey, International Journal of Operations and Productions Management, 13 (11), 80-91.

Kaplan, L., 1988, Resource-constrained project scheduling with pre-emption of jobs, Unpublished Phd Dissertation, University of Michigan.

Kaplan, L., 1991, Resource-constrained project scheduling with setup times", Unpublished paper, Department of Management, University of Tenessee, Knoxville.

Kolisch, R., Padman, R., 2001. An integrated survey of deterministic project scheduling, Omega, 49 (3), 249-272.

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

Mika, M., Waligora, G., Weglarz, J., 2006. Precedence-independent and precedence-dependent setup times in project scheduling problems, Tenth International Workshop on Project Management and Scheduling, 248-253.

Özdamar, L., Ulusoy, G., 1995. A survey on the resource-constrained project scheduling problem, IIE Transactions, 27, 574-586.

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