Available online at www.sciencedirect.com
ScienceDirect
Computers & Industrial Engineering 54 (2008) 140–154
computers & industrial engineering
www.elsevier.com/locate/dsw

The impact of various activity assumptions on the lead time and resource utilization of resource-constrained projects

Mario Vanhoucke [a,b,*], Dieter Debels [a]

[a] Faculty of Economics and Business Administration, Ghent University, Ghent, Belgium
[b] Operations & Technology Management Centre, Vlerick Leuven Gent Management School, Ghent, Belgium

Received 30 October 2006; received in revised form 3 July 2007; accepted 3 July 2007
Available online 10 July 2007

Abstract

The well-known resource-constrained project scheduling problem (RCPSP) schedules project activities within the precedence and renewable resource constraints while minimizing the total lead time of the project. The basic problem description assumes non-pre-emptive activities with fixed durations, and has been extended to various other assumptions in the literature. In this paper, we investigate the effect of three activity assumptions on the total lead time and the total resource utilization of a project. More precisely, we investigate the influence of variable activity durations under a fixed work content, the possibility of allowing activity pre-emption and the use of fast tracking to decrease a project's duration. We give an overview of the procedures developed in the literature and present some modifications to an existing solution approach to cope with our activity assumptions under study. We present computational results on a generated dataset and evaluate the impact of all assumptions on the quality of the schedule.

Keywords: Project scheduling; Resource-constraints; Lead-time minimization

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 an activity-on-the-node (AoN) format, the set of nodes N are used to represent the n activities (numbered from 1 to n, i.e. $|N| = n$) and a set of pairs of activities A represent the precedence relations between activities, which are assumed to be finish-start relations with a minimal time-lag of zero. 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 $d_i \in \mathrm{IN}$ and requires $r_{ik} \in \mathrm{IN}$ units of resource type k. The dummy start and end activities one and n have zero duration and zero resource usage. A schedule can be defined by an n-vector of start times and implies an n-vector of finish times. A schedule

* Corresponding author. Address: Faculty of Economics and Business Administration, Ghent University, Ghent, Belgium.
E-mail addresses: mario.vanhoucke@ugent.be (M. Vanhoucke), dieter.debels@ugent.be (D. Debels).

0360-8352/$ - see front matter
doi:10.1016/j.cie.2007.07.001

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 is minimized.

Fig. 1 displays a project network example with 9 activities and one resource type with an availability a_1 of 6. This example will be used throughout the remainder of this paper. The duration d_i of each activity i has been displayed above the node, while the resource demand r_{i1} has been shown below the node. The optimal solution with a minimal project duration of 9 has been displayed at the right part of Fig. 1.

In the basic RCPSP formulation, activities are assumed to have a fixed duration and are not allowed to be pre-empted. In this paper, we relax these strict activity assumptions and investigate their impact on the quality of the project schedule. More precisely, we study three extended activity assumptions, i.e. the extension from fixed to variable activity durations, the presence of activity splitting (pre-emption) and the possibility of the parallel execution of subparts of activities (fast tracking). The contribution and purpose of this research is as follows: First, we modify the exact branch-and-bound procedure of Demeulemeester and Herroelen (1992) in order to solve the extended versions of the RCPSP. Second, we generate optimal schedules for various projects and evaluate the impact of these various activity assumptions on the total project lead time as well as on the efficiency of resource use.

The outline of the paper is as follows. In Section 2 we discuss the three activity assumptions in detail. We show that the introduction of these additional activity assumptions does not fundamentally change the original RCPSP formulation and consequently all extended problem formulations can be solved by any RCPSP solution procedure. In Section 3, we propose our adaptations on the well-known branch-and-bound RCPSP procedure of Demeulemeester and Herroelen (1992) in order to cope with most of our new assumptions. Section 4 presents detailed computational results and investigates the impact of the activity assumptions on the quality of the schedule, both from a lead time as from a resource point-of-view. Section 5 presents some overall conclusions and suggestions for future research.

2. Project scheduling under three activity assumptions

Many project scheduling software packages aim at the construction of resource-feasible schedules in order to minimize the total lead time of the project. Hence, an AoN project network with a list of activities with their corresponding precedence relations and resource requirements need to be given as an input. However, various activity assumptions need to be made by the user in order to construct a feasible schedule. We investigate three different activity assumptions, as summarized in Fig. 2. This figure displays the effect of the three assumptions on activity 2 of Fig. 1. These extensions are:

- Fixed duration or fixed work.
- The presence of activity pre-emption.
- The effect of fast tracking.

Fixed duration or fixed work: The basic RCPSP assumes that each activity i consists of a deterministic work content W_{ik} for each resource type k, and imposes a fixed duration d_i and a fixed resource requirements r_{ik} on

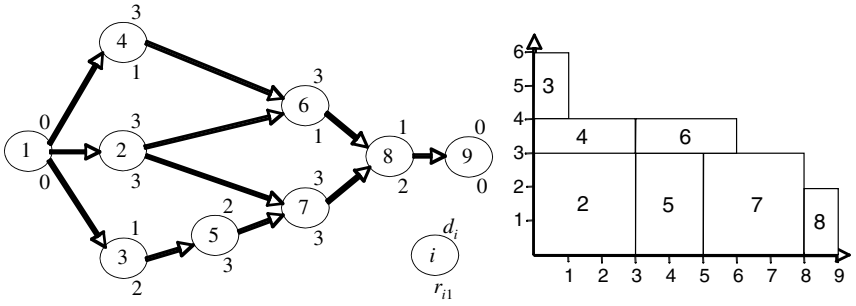

Fig. 1. Example activity network and optimal schedule of the RCPSP.

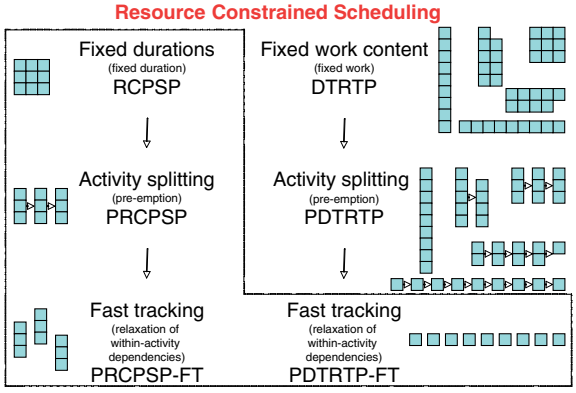

Fig. 2. Resource-constrained project scheduling under various activity assumptions.

its execution. The extension to the discrete time/resource trade-off problem (DTRTP) still assumes a fixed work content but allows variable activity durations. As an example, activity 2 of Fig. 1 still has a fixed work content W_{i1} of 9 for the single resource type one, but can now be executed under different scenarios. Note that many commercial software packages pay a lot of attention to this activity assumption, and call for the well-considered use of this activity option before the construction of a schedule (see e.g Uyttewaal (2005) for examples).

Activity pre-emption. The basic RCPSP assumes that each activity, once started, will be executed until its completion. The extension to the *pre-emptive resource-constrained project scheduling problem* (PRCPSP) allows activities to be pre-empted at any integer time instant and restarted later on at no additional cost, and has been investigated in the literature as an option to further reduce the total project lead time. The literature for the *pre-emptive discrete time/resource trade-off problem* (PDTRTP) is, to the best of our knowledge, completely void. In most project scheduling software packages, the option of activity splitting can be made before the construction of a resource-feasible schedule and has an effect on the number of execution scenarios, as displayed in Fig. 2.

Fast tracking: Fast tracking is a scheduling technique used to reduce the total project lead time during project execution. 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, it violates the precedence relations between activities which implies activity execution at incomplete information. In our paper, we investigate the impact of within-activity fast tracking, which allows the execution of pre-emptive subparts of an activity in parallel. The fast tracking option removes precedence relations between subparts of pre-empted activities and increases the number of execution scenarios. The within-activity fast tracking option is inspired by the idea that activities are often executed by groups of resources (with a fixed availability), but the total work can often be done by multiple groups (in parallel). The *pre-emptive resource-constrained project scheduling problem with fast tracking* (PRCPSP-FT) assumes pre-emptive activities with fixed durations, which results in d_i parallel subactivities each with a resource requirement r_{ik}. The pre-emptive discrete time/resource trade-off problem with fast tracking (PDTRTP-FT) assumes variable activity durations (under a fixed work content) and allows the pre-emptive and parallel execution of each subactivity with a duration and resource requirement equal to one, as shown in the bottom part of Fig. 2. To the best of our knowledge, the literature of resource-constrained project scheduling with a fast tracking option between pre-emptive subparts of activities is completely void.

In the next subsection, we show that the PRCPSP, the PRCPSP-FT and the PDTRTP-FT can be solved by any solution approach for the RCPSP (denoted by the dashed lines in Fig. 2). In Section 2.2, we elaborate on the DTRTP and the PDTRTP.

2.1. The subactivity network for the PRCPSP, PRCPSP-FT and PDTRTP-FT

In this section, we show that the resource-constrained project scheduling problem can be easily extended to cope with three of our activity assumptions, i.e. PRCPSP, PRCPSP-FT and PDTRTP-FT, and hence, these problem instances can be solved by any solution algorithm for the RCPSP.

Kaplan (1988, 1991) was the first to study the PRCPSP, but she did not present a correct exact solution procedure (Demeulemeester & Herroelen, 1996). Ballestin, Valls, and Quintanilla (2006) have developed a meta-heuristic procedure to solve the PRCPSP. Demeulemeester and Herroelen (1996) have translated the RCPSP to the PRCPSP by means of a subactivity project network $G(N', A')$ and developed a branch-and-bound procedure to optimally solve the problem. In a subactivity network, each activity i is split up into d_i subactivities i_s $(s = 1, \ldots, d_i)$ with a subactivity duration $d_{i_s} = 1$ and a corresponding resource requirement $r_{i_s k} = r_{ik}$. The PRCPSP allows activity pre-emption and assumes that the remaining part of the activity is scheduled later in the schedule. Hence, a precedence constraint between each pair (i_s, i_{s+1}) is added in the subactivity network. The complete PRPCSP subactivity network has been displayed in Fig. 3 and splits the 7 non-dummy activities into 16 subactivities. The optimal schedule is displayed in the right part of Fig. 3 and leads to an overall project lead-time reduction from 9 to 8 as a result of the pre-emption of activities 4 and 5.

The option to fast track pre-empted subparts of activities boils down to the option to schedule subactivities of the same activity in parallel, and hence, implies the removal of all precedence relations between subactivities of the same activity. Consequently, the subactivity network for the PRCPSP-FT assumes that each activity i is split up into d_i subactivities i_s $(s = 1, \ldots, d_i)$ with a subactivity duration $d_{i_s} = 1$, resource requirements $r_{i_s k} = r_{ik}$, and no precedence relations between subactivities of the same activity. Fig. 4 represents the subactivity network and corresponding optimal schedule for the PRCPSP-FT. The PRCPSP-FT schedule shows a decreased lead time from 8 to 7 time units as a result of the parallel execution of pre-emptive subpart for activities 2, 4, 5, 6 and 7.

The fast track option for the PDTRTP-FT assumes a subactivity network where each activity i is split up into W_{ik} subactivities i_s $(s = 1, \ldots, W_{ik})$ with a subactivity duration $d_{i_s} = 1$, resource requirements $r_{i_s k} = 1$, and no precedence relations between subactivities of the same activity. Fig. 5 represents the subactivity network and corresponding optimal schedule for the PDTRTP-FT. The subactivity network for the PDTRTP-FT contains 34 non-dummy subactivities, with all durations and resource requirements equal to one. The optimal resource-feasible schedule also has a minimal project lead time of 7.

Since the PRCPSP, PRCPSP-FT and DTRTP-FT can be represented by a subactivity network, these problem types can be solved by any algorithm for the RCPSP. Many exact and (meta-)heuristic RCPSP procedures have been presented in the literature, and overviews can be found in Brucker, Drexl, Möhring, Neumann, and Pesch (1999), Hartmann and Kolisch (2000), Herroelen, De Reyck, and Demeulemeester (1998), Icmeli, Erenguc, and Zappe (1993), Kolisch and Hartmann (2006), Kolisch and Padman (2001) and Özdamar and Ulusoy (1995). In our current paper, we rely on the efficient branch-and-bound procedure of Demeulemeester and Herroelen (1992) to solve various problem instances. We would like to mention that we could have used alternative problem formulations and procedures available in the literature. However, the scope of the current manuscript is not to develop efficient state-of-the-art procedures, but rather to test the impact of the activity assumptions of Fig. 2 on the quality of the optimal schedule, both from a lead time and a resource utilization point-of-view. Hence, the choice of the algorithm of Demeulemeester and Herroelen (1992) has been made by practical reasons (the code was available to us) and allows us to generate, like any other optimal RCPSP procedure, optimal schedules that can be used for comparison purposes. The depth-first approach of Demeulemeester and Herroelen (1992) 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

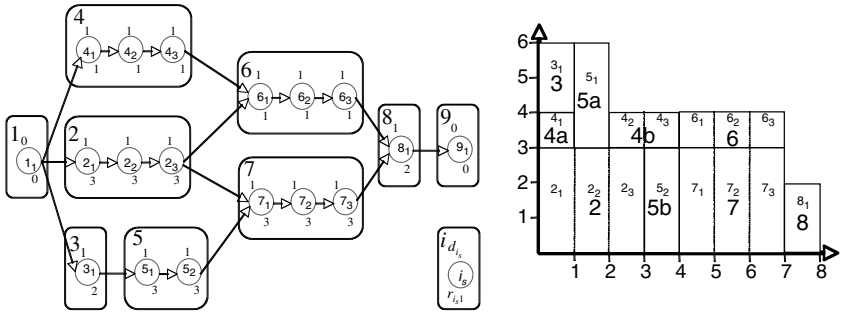

Fig. 3. The subactivity network and corresponding optimal schedule for the PRCPSP.

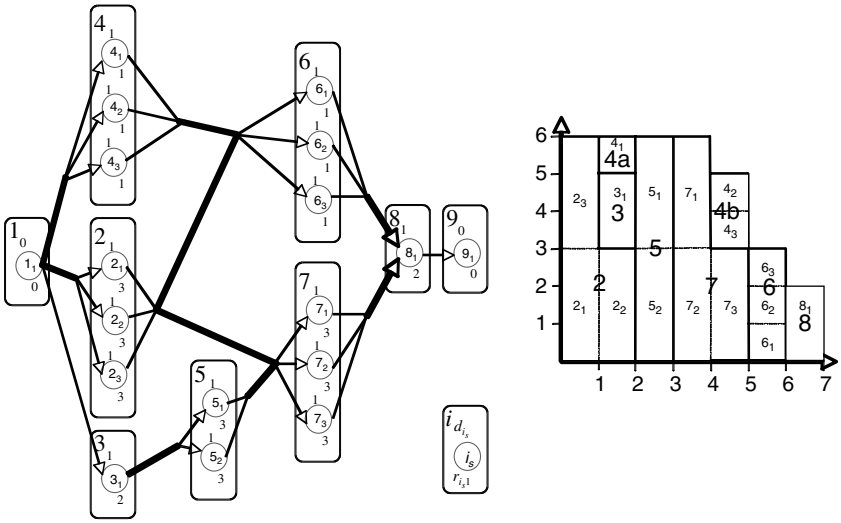

Fig. 4. The subactivity network and corresponding optimal schedule for the PRCPSP-FT.

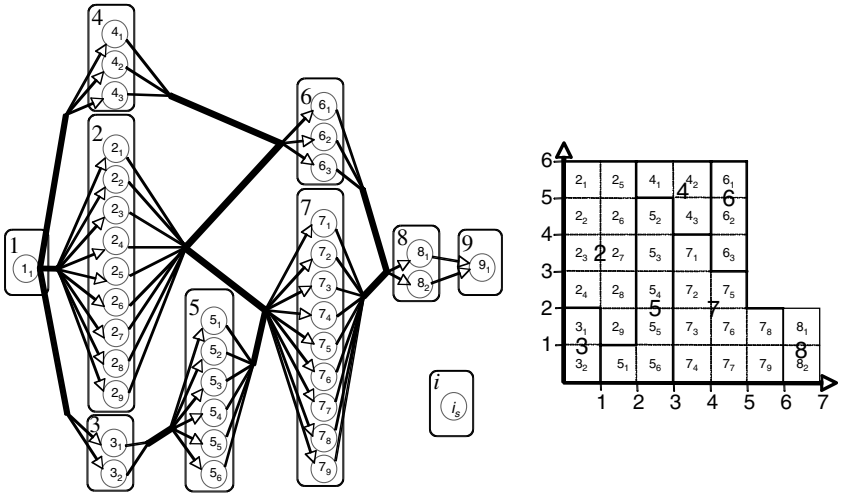

Fig. 5. The subactivity network and corresponding optimal schedule PDTRTP-FT.

obtained. A partial schedule at level p of the search tree will be further built by determining the next decision moment dm at which unscheduled activities might start. All unscheduled activities that are a candidate to start at time dm are calculated and collected in the set E of eligible activities. The previously scheduled but at dm unfinished activities belong to the set S of activities in progress. If scheduling all activities from $E \cup S$ at dm would cause a resource conflict (i.e. a resource over-allocation of c_k units above the a_k limit for resource type k), the procedure starts to branch to the next level $p+1$ and delays subsets (known as delaying alternatives, with D_q^p the q^{th} delaying alternative of level p) of $E \cup S$ to resolve resource conflicts. It has been shown that it is sufficient to limit the search to the *minimal delaying alternatives* D_q^p, which contain no other delaying alternatives $D_{q'}^p$ as a subset. Formally, the set of minimal delaying alternatives $D^p = \{D_q^p \subset (S \cup E) | \sum_{i \in D_q^p} r_{ik} \geqslant c_k$ for each $k \in R$ and there exists no other $D_{q'}^p \in D^p$ for which $D_{q'}^p \subset D_q^p\}$. Then, a minimal delaying alternative needs to be selected, which involves that only the unselected activities of $E \cup S$ will be scheduled at dm while all previously scheduled activities of S and the activities of E 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 branch-and-bound procedure has been made very efficient as a result of a number of dominance rules (probably the best known is the cutset dominance rule) and efficient lower bound calculations.

Fig. 6 briefly illustrates this solution approach on the partial schedule of the project example of Fig. 1. The figure assumes that the algorithm is at decision moment $dm = 1$ and activities 2, 3 and 4 have been scheduled at the previous decision moment. The eligible set $E = \{5\}$ and the set of activities in progress $S = \{2,4\}$. Entering all activities of the eligible set in the partial set results in a resource conflict (total resource consumption of activities of $E \cup S$ equals 7 and, consequently, $c_1 = 7-6 = 1$) and hence the algorithm needs to evaluate all minimal delaying activities at the next level of the branch-and-bound tree. Since the set of minimal delaying alternative contains three sets with a single activity $\{(2),(3),(4)\}$, the next level of the branch-and-bound tree consists of three nodes.

Demeulemeester and Herroelen (1996) have adapted their original RCPSP branch-and-bound procedure to cope with pre-emptive activities. To that purpose, they rely on the subactivity network (see Fig. 3) with all activity durations equal to one. Indeed, since all subactivities that are scheduled at a decision moment dm automatically end one time unit later, the next decision moment automatically equals $dm + 1$, resulting in an empty set S of activities in progress. The authors observe a clear trade-off between computational effort to solve the PRCPSP and the resulting schedule quality improvements compared to the RCPSP, and show that activity pre-emption has only a small positive effect on a project's lead time. However, Ballestin et al. (2006) recently showed that high-quality heuristic solutions can be obtained more easily for the PRCPSP than for the RCPSP.

In the current paper, we use the original branch-and-bound procedure of Demeulemeester and Herroelen (1992) to solve the RCPSP, and adapt this procedure to make it more efficient for solving the RCPSP-FT and the PDTRTP-FT (see Section 3).

2.2. The solution approach for the DTRTP

The DTRTP assumes variable activity durations and resource requirements with a fixed work content W_{i1} for a single resource type (note that only one resource type is considered, and hence, no resource/resource trade-offs between multiple resources are included). Each activity can be executed according to a set of feasible execution modes M_i. Every mode m represents a combination of duration $d_{i(m)}$and resource requirements $r_{i1}(m)$, for which $d_i(m) * r_{i1}(m) \geqslant W_{i1}$. De Reyck, Demeulemeester, and Herroelen (1998) have shown that it is sufficient to consider only efficient modes for which all other feasible modes are either higher in duration or higher in resource requirements. As an example, set M_2 of Fig. 7 contains 5 efficient modes $m = (d_i(m), r_{i1}(m))$, i.e. $M_2 = \{(1,9),(2,5),(3,3),(5,2),(9,1)\}$. Note that modes (2,5) and (5,2) exceed the minimal work content of 9 by 1 unit, and mode (1,9) is infeasible with respect to the renewable resource constraints. The optimal schedule has a decreased lead time from 9 to 7 time units when shifting from fixed durations (RCPSP) to fixed work content (DTRTP) as a result of the selection of a different mode for activities 4, 6 and 7.

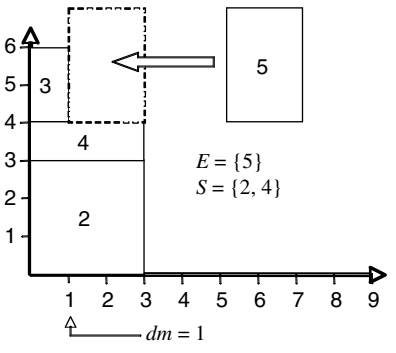

Fig. 6. The Demeulemeester and Herroelen (1992) approach on decision moment one for the project example of Fig. 1.

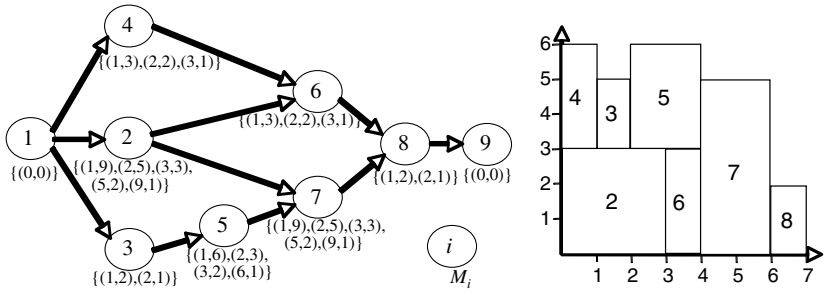

Fig. 7. The activity network and corresponding optimal schedule for the DTRTP.

Demeulemeester, De Reyck, and Herroelen (2000) have presented a branch-and-bound procedure for the DTRTP that relies on the *activity-mode combination* branching strategy as an extension of the minimal delaying alternative branching strategy. Activity-mode combinations are subsets of the candidate activities of set $(E \cup S)$, executed in a specific mode. The authors have shown that only feasible and maximal combinations need to be considered. An activity-mode combination is *feasible* if the activities can be executed in parallel in the specified mode without causing a resource conflict, and *maximal* if no other activity can be added in one of its modes without causing a resource conflict. The authors also mention the importance of efficient resource based lower bounds since the resource utilization for a DTRTP schedule is often much higher than for an RCPSP schedule. The literature for the PDTRTP is, to the best of our knowledge, completely void.

Table 1 displays all activity-mode combinations of the partial schedule of Fig. 7 at decision moment 0. The eligible set E contains activities 2, 3 and 4 which can be scheduled under different modes at the current time instance. The table evaluates all 71 possible mode combinations, and indicates whether a combination is feasible and/or maximal (the 17 combinations in bold are both feasible and maximal).

In the remainder of this paper, we present new exact approaches for the PRCPSP-FT and the PDTRTP-FT, since no dedicated procedures have been presented yet in the literature for these problem types. As they can be represented by subactivity networks, high-quality solution procedures can be constructed rather easily by modifying existing RCPSP procedures. Moreover, we do not present new solution procedures for the DTRTP and the PRCPSP, but rather rely on existing procedures for these problem types to compare their results with the results of the other problems under study. Finally, we do not consider the PDTRTP for two reasons. Firstly, no exact procedures have been presented in the literature and hence, no comparison is possible. Secondly, the problem type can not be transformed to a subactivity network as is the case for the PRCPSP, the PRCPSP-FT and the PDTRTP-FT.

3. A branch-and-bound procedure

In this section we present two adaptations to the branch-and-bound procedure of Demeulemeester and Herroelen (1992) in order to solve the PRCPSP-FT and the PDTRTP-FT more efficiently. Section 3.1 presents an adapted minimal delaying alternative approach to solve the PRCPSP-FT. In Section 2.1, we present adapted lower bound and upper bound calculations for the PDTRTP-FT.

3.1. The minimal delaying alternatives for the PRCPSP-FT and the PDTRTP-FT

Demeulemeester and Herroelen (1992) have shown that only minimal delaying alternatives need to be investigated during their branch-and-bound procedure. A minimal delaying alternative is a subset of activities to delay in order to resolve a resource conflict that contains no other delaying alternative as a subset. Since the PRCPSP-FT removes precedence relations between subactivities, all subactivities i_s of an activity i become eligible to be scheduled at the same decision moment, and hence, the number of minimal delaying alternatives at each level of the search tree grows exponentially. However, an extension of the minimal delaying alternative

Table 1
Feasible and maximal mode combinations at decision moment $dm = 0$

Act. 2	Act. 3	Act. 4	Feas?	Max?
(1,9)	–	–	No	–
(2,5)	–	–	Yes	No
(3,3)	–	–	Yes	No
(5,2)	–	–	Yes	No
(9,1)	–	–	Yes	No
–	(1,2)	–	Yes	No
–	(2,1)	–	Yes	No
–	–	(1,3)	Yes	No
–	–	(2,2)	Yes	No
–	–	(3,1)	Yes	No
(1,9)	(1,2)	–	No	–
(1,9)	(2,1)	–	No	–
(2,5)	(1,2)	–	No	–
(2,5)	**(2,1)**	–	**Yes**	**Yes**
(3,3)	(1,2)	–	Yes	No
(3,3)	(2,1)	–	Yes	No
(5,2)	(1,2)	–	Yes	No
(5,2)	(2,1)	–	Yes	No
(9,1)	(1,2)	–	Yes	No
(9,1)	(2,1)	–	Yes	No
(1,9)	–	(1,3)	No	–
(1,9)	–	(2,2)	No	–
(1,9)	–	(3,1)	No	–
(2,5)	–	(1,3)	No	–
(2,5)	–	(2,2)	No	–
(2,5)	–	**(3,1)**	**Yes**	**Yes**
(3,3)	–	**(1,3)**	**Yes**	**Yes**
(3,3)	–	(2,2)	Yes	No
(3,3)	–	(3,1)	Yes	No
(5,2)	–	(1,3)	Yes	No
(5,2)	–	(2,2)	Yes	No
(5,2)	–	(3,1)	Yes	No
(9,1)	–	(1,3)	Yes	No
(9,1)	–	(2,2)	Yes	No
(9,1)	–	(3,1)	Yes	No
–	(1,2)	(1,3)	Yes	No
–	(1,2)	(2,2)	Yes	No
–	(1,2)	(3,1)	Yes	No
–	(2,1)	(1,3)	Yes	No
–	(2,1)	(2,2)	Yes	No
–	(2,1)	(3,1)	Yes	No
(1,9)	(1,2)	(1,3)	No	–
(1,9)	(1,2)	(2,2)	No	–
(1,9)	(1,2)	(3,1)	No	–
(1,9)	(2,1)	(1,3)	No	–
(1,9)	(2,1)	(2,2)	No	–
(1,9)	(2,1)	(3,1)	No	–
(2,5)	(1,2)	(1,3)	No	–
(2,5)	(1,2)	(2,2)	No	–
(2,5)	(1,2)	(3,1)	No	–
(2,5)	(2,1)	(1,3)	No	–
(2,5)	(2,1)	(2,2)	No	–
(2,5)	(2,1)	(3,1)	No	–
(3,3)	(1,2)	(1,3)	No	–
(3,3)	(1,2)	(2,2)	No	–
(3,3)	**(1,2)**	**(3,1)**	**Yes**	**Yes**
(3,3)	(2,1)	(1,3)	No	–

(*continued on next page*)

Table 1 (*continued*)

Act. 2	Act. 3	Act. 4	Feas?	Max?
(3,3)	**(2,1)**	**(2,2)**	**Yes**	**Yes**
(3,3)	**(2,1)**	**(3,1)**	**Yes**	**Yes**
(5,2)	(1,2)	(1,3)	No	–
(5,2)	(1,2)	**(2,2)**	**Yes**	**Yes**
(5,2)	(1,2)	**(3,1)**	**Yes**	**Yes**
(5,2)	**(2,1)**	**(1,3)**	**Yes**	**Yes**
(5,2)	(2,1)	**(2,2)**	**Yes**	**Yes**
(5,2)	(2,1)	**(3,1)**	**Yes**	**Yes**
(9,1)	**(1,2)**	**(1,3)**	**Yes**	**Yes**
(9,1)	(1,2)	**(2,2)**	**Yes**	**Yes**
(9,1)	(1,2)	**(3,1)**	**Yes**	**Yes**
(9,1)	**(2,1)**	**(1,3)**	**Yes**	**Yes**
(9,1)	(2,1)	**(2,2)**	**Yes**	**Yes**
(9,1)	(2,1)	**(3,1)**	**Yes**	**Yes**

principle limits the search of delaying alternatives to subsets of the eligible activities of set E, and dominates many nodes in the branch-and-bound tree, as follows:

Theorem: In order to define the set of minimal delaying alternatives for the PRCPSP-FT and the PDTRTP-FT, it is sufficient to define the number of subactivities e_i for each activity i that should be chosen from the eligible set E.

The theorem implies that it is not required to define which subactivities should be selected from the eligible set for entrance in each minimal delaying alternative. All subactivities have a duration of one and the set S of activities in progress is always empty at the decision moment dm. Hence, if a minimal delaying alternative selects e_i subactivities of activity i from the eligible set E, then every other combination of e_i subactivities of i in E will lead to an equivalent schedule. In our specific implementation, we always select the e_i highest numbered subactivities of activity i to enter the minimal delaying alternative, such that the remaining lower numbered subactivities are scheduled at dm.

Table 2 illustrates the theorem at the initial decision moment 0 for the example PRCPSP-FT problem of Fig. 4. The set E of eligible subactivities contains all subactivities of activities 2, 3 and 4, i.e. $E = \{2_1, 2_2, 2_3, 3_1, 4_1, 4_2, 4_3\}$. Scheduling all subactivities in parallel results in a total resource demand of 14 units, which exceeds the availability of 6. In order to solve this resource conflict, the branch-and-bound

Table 2
The minimal delaying alternatives at the initial level of the example problem

Activity	$i=2$			$i=3$	$i=4$			e_2	e_3	e_4	Selected
Subactivity	1	2	3	1	1	2	3				
Alternative											
1			x	x	x	x	x	1	1	3	Yes
2		x		x	x	x	x	1	1	3	No
3	x			x	x	x	x	1	1	3	No
4		x	x			x	x	2	0	2	Yes
5		x	x		x		x	2	0	2	No
6		x	x		x	x		2	0	2	No
7	x		x			x	x	2	0	2	No
8	x		x		x		x	2	0	2	No
9	x		x		x	x		2	0	2	No
10	x	x				x	x	2	0	2	No
11	x	x			x		x	2	0	2	No
12	x	x			x	x		2	0	2	No
13		x	x	x				2	1	0	Yes
14	x		x	x				2	1	0	No
15	x	x		x				2	1	0	No
16	x	x	x					3	0	0	Yes

procedure of Demeulemeester and Herroelen (1992) generates 16 minimal delaying alternatives. The theorem selects only one delaying alternative for each combination e_2, e_3 and e_4, and hence, only alternatives 1, 4, 13 and 16 need to be considered in the tree.

3.2. The lower and upper bound calculations for the PDTRTP-FT

The PDTRTP-FT assumes subactivities with all durations and resource requirements equal to one and no precedence relations between within-activity subactivities. Hence, the problem type is a strong relaxation of the RCPSP for which many alternative optimal schedules exist. In this section, we present straightforward yet efficient lower and upper bounds that dramatically improve the efficiency of the adapted branch-and-bound algorithm of Section 2.1.

3.2.1. Lower bound LB_p

The algorithm calculates at each node of the branch-and-bound tree *the minimal remaining duration* L_{i_s} of each subactivity i_s of the eligible set E (i.e. ready to be scheduled) at decision moment dm. Hence, the lower bound LB_p at level p of the tree equals $dm + \max_{i_s \in E}(L_{i_s})$ and is based on the backward calculations (from the dummy end node to the dummy start node) of L_{i_s}, as follows:

$$L_{i_s} = \max\left(\left\lceil \frac{|S_{i_s}|}{a_1} \right\rceil, \max_{j_s \in S_{i_s}}(L_{j_s})\right) + 1 + \left\lfloor \frac{u_{i_s}}{a_1} \right\rfloor$$

where S_{i_s} is used to denote the set of all (immediate and transitive) non-dummy successor subactivities of subactivity i_s and $|S_{i_s}|$ is used to represent the number of subactivities in this set. Moreover, u_{i_s} is used to represent the number of subactivities of activity i with a higher subscript than the subscript s of subactivity i_s (these are subactivities $i_{s+1}, i_{s+2}, \ldots, i_{W_{ik}}$). Note that all higher numbered subactivities can not be scheduled earlier than subactivity i_s due to our specific delaying alternative approach of the theorem of previous section. The lower bound calculates the minimal remaining length of each activity as the maximum of the resource based remaining schedule length $\left\lceil \frac{|S_{i_s}|}{a_1} \right\rceil$ and the minimal remaining length of its successors $\max_{j_s \in S_{i_s}}(L_{j_s})$, increased by a factor $1 + \left\lfloor \frac{u_{i_s}}{a_1} \right\rfloor$ to represent the minimal required extra time needed to schedule i_s and its u_{i_s} higher subscripted subactivities of the same activity i.

3.2.2. Lower bound LB_0

At the initial node of the branch-and-bound tree, the algorithm determines the earliest possible start time EST_{i_s} of each subactivity i_s, and hence, the initial lower bound LB_0 can be calculated as $LB_0 = \max_{\forall i_s \in N}(EST_{i_s} + L_{i_s})$, with

$$EST_{i_s} = \max\left(\left\lceil \frac{|P_{i_s}|}{a_1} \right\rceil, \max_{j_s \in P_{i_s}}\left(EST_{j_s}\right) + 1\right) + \left\lfloor \frac{l_{i_s}}{a_1} \right\rfloor$$

where P_{i_s} is used to denote the set of all (immediate and transitive) non-dummy predecessor subactivities of subactivity i_s and $|P_{i_s}|$ is used to represent the number of subactivities in this set. Moreover, l_{i_s} is used to represent the number of subactivities of activity i with a lower subscript than the subscript s of activity i_s (these are subactivities $i_1, i_2, \ldots, i_{s-1}$).

3.2.3. Upper bound UB_0

At the start of the search process, the algorithm calculates an initial resource feasible upper bound solution based on a straightforward priority rule based forward scheduling technique. More precisely, the algorithm constructs the priority rule according to decreasing values for L_{i_s} and generates a resource-feasible schedule using the well-known serial schedule generation scheme (see e.g. Kolisch & Padman, 2001). When tie-breaks occur (i.e. activities with identical L_{i_s} priority values), the schedule generation scheme first selects the activities with the maximum $|S_{i_s}|$ value and a higher subactivity index to enter the schedule (i.e. the tie-breaking rules).

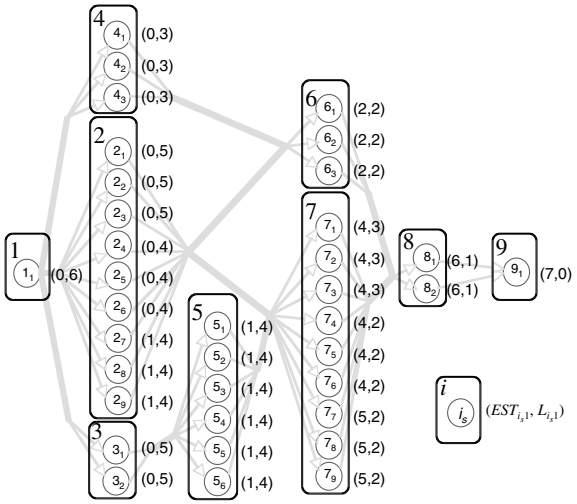

Fig. 8. The subactivity network for the PDTRTP-FT with EST_{i_s} and L_{i_s} values.

Fig. 8 displays the subactivity network of Fig. 1 for the PDTRTP-FT, with the values of EST_{i_s} and L_{i_s}. The lower bound LB_0 equals 7 (see nodes 7_7, 7_8, 7_9, 8_1, 8_2 and 9_1) and the upper bound UB_0 has a total duration of 7 and corresponds to the schedule of Fig. 5. Hence, the resource-feasible schedule constructed for the upper bound is optimal, and no branching is needed.

4. Computational results

In this section, we test the impact of the three activity assumptions on the problem CPU time requirements and the schedule quality, based on 1920 randomly generated project networks generated by RanGen (Demeulemeester, Vanhoucke, & Herroelen, 2003). The number of non-dummy activities ($n - 2$) has been set at 10, 12, 14, 16, 18 and 20 with an order strength OS (defined as the number of precedence relations, including the transitive ones but not including the arcs connecting the dummy start or end activity, divided by the theoretical maximum number of precedence relations, see Mastor, 1970) and a resource-constrainedness RC (defined as the average quantity of a resource demanded by the activities divided by its resource availability, see Patterson, 1976) fixed at 0.2, 0.4, 0.6 and 0.8. Since the formulation of the DTRTP and the PDTRTP-FT does not allow multiple resources, all project instances contain only a single resource type with an availability of 10 units. The activity durations have been chosen randomly between 1 and 5. Using 20 instances for each problem setting, we obtain a problem set of $6 * 4 * 4 * 20 = 1920$ network instances that can also be found at http://www.project-management.ugent.be/Downloads.php. In Section 4.1, we analyse CPU time requirements for the RCPSP procedure applied to the pre-emptive problem types, and compare them with dedicated procedures. Section 4.2 investigates the impact of all assumptions on the total project lead time and the utilization of resources.

4.1. Impact of the activity assumptions on the problem complexity

In this section, we report computational results for the PRCPSP, the PRCPSP-FT and the PDTRTP-FT. These pre-emptive problem types can be solved by the efficient RCPSP procedure of Demeulemeester and Herroelen (1992) as a result of the transformation to subactivity networks. Moreover, the contribution of the adaptations of Section 3 will be tested for the PRCPSP-FT and PDTRTP-FT.

The results have been displayed in Table 3. The abbreviation 'DH92' refers to the branch-and-bound procedure of Demeulemeester and Herroelen (1992) while the abbreviation 'DH96' refers to the branch-and-bound procedure of Demeulemeester and Herroelen (1996) for the PRCPSP. The adapted branch-and-bound approach as discussed in Section 3 will be abbreviated with 'VD06'. The rows labeled "subactivities" displays the average number of subactivities in the subactivity network. The row "Avg. OS" displays the average value

Table 3
The RCPSP and the impact of the pre-emption (PRCPSP) and fast tracking (PRCPSP-FT and PDTRTP-FT) on problem complexity

Number of activities	10	12	14	16	18	20	Total
RCPSP							
Subactivities	10	12	14	16	18	20	
Avg. OS	0.5	0.5	0.5	0.5	0.5	0.5	0.5
DH92							
Avg. CPU	0.00	0.00	0.00	0.00	0.00	0.01	0.00
%Opt	100%	100%	100%	100%	100%	100%	100%
PRCPSP							
Subactivities	31.83	37.82	44.21	50.21	55.75	61.85	
Avg. OS	0.55	0.53	0.53	0.53	0.53	0.52	0.53
DH92							
Avg. CPU	0.00	0.00	0.04	0.09	0.85	3.67	0.77
%Opt	100%	100%	100%	100%	99%	97%	99%
DH96							
Avg. CPU	0.00	0.00	0.03	0.06	0.57	3.20	0.64
%Opt	100%	100%	100%	100%	100%	97%	99%
PRCPSP-FT							
Subactivities	31.83	37.82	44.21	50.21	55.75	61.85	
Avg. OS	0.46	0.46	0.46	0.47	0.48	0.48	0.47
DH92							
Avg. CPU	0.43	2.64	7.59	11.67	16.95	23.82	10.52
% Opt	100%	98%	93%	90%	86%	80%	91%
VD06							
Avg. CPU	0.00	0.00	0.05	0.22	1.22	3.96	0.91
% Opt	100%	100%	100%	100%	99%	97%	99%
PDTRTP-FT							
Subactivities	151.56	181.13	212.33	243.00	269.80	300.25	
Avg. OS	0.41	0.41	0.43	0.44	0.44	0.45	0.43
DH92							
Avg. CPU	97.04	98.46	99.88	100.00	100.00	100.00	99.23
%Opt	3%	2%	0%	0%	0%	0%	1%
VD06							
Avg. CPU	0.04	0.36	0.40	0.35	1.30	0.50	0.49
% Opt	100%	100%	100%	100%	99%	100%	100%
$LB_0 = UB_0$	91%	92%	94%	92%	92%	93%	92%

Note: DH92, procedure used for RCPSP, PRCPSP, PRCPSP-FT and PDTRTP-FT; DH96, procedure used for the PRCPSP; VD06, procedure for PRCPSP-FT and PDTRTP-FT.

for the order strength OS, defined as the number of immediate and transitive precedence relations between the $(n-2)$ non-dummy (sub)activities in relation to a maximal number $((n-2)*(n-3))/2$ of precedence relations between the (sub)activities. The average OS value equals 0.5 for the original problem instances, which is an average of our RanGen input settings 0.2, 0.4, 0.6 and 0.8. The row "Avg. CPU" displays the average CPU time (in seconds) needed to solve a problem instance and the row "% Opt" reports the number of problem instances that could be solved optimally within a time limit of 100 s.

The results can be summarized as follows. First, the table clearly reveals the increase in CPU time requirements as we move further from the basic assumptions of the RCPSP. All RCPSP relaxations lead to an increase in the number of subactivities (e.g. an increase from 20 to approximately 61.85 subactivities for the PRCPSP and PRCPSP-FT and to 300.25 subactivities for the PDTRTP-FT). However, note that the PRCPSP is still easier to solve than the PRCPSP-FT, although both rows show an equal number of subactivities. The order strength, that measures the presence of precedence relations in the subactivity network, has decreased from 0.5 to approximately 0.46 for the PRCPSP-FT and increased to approximately 0.53 for the PRCPSP. This changed OS value is responsible for the difference in problem complexity between the PRCPSP and the PRCPSP-FT, which is completely in line with the negative effect of the OS on CPU time requirements as mentioned in the literature (Herroelen & De Reyck, 1999).

Second, the table shows that dedicated and problem specific algorithms always perform better than the RCPSP branch-and-bound procedure applied on the subactivity networks. Though the DH92 procedure shows relatively good results for the PRCPSP, the dedicated DH96 clearly outperforms it. The PRCPSP-FT and the PDTRTP-FT could not be solved efficiently by the DH92 procedure. The VD06 adaptations clearly improve the results, both from a CPU time as from a percentage solved to optimality point-of-view. The VD06 procedure is able to solve all PRCPSP-FT problem instances with up to 16 non-dummy activities, and can solve 309 20-activity problem instances within the pre-specified limit of 100 s. The average CPU times decrease drastically compared to the DH92 procedure from 23.82 to less than 1 s for the 20-activity instances. The PDTRTP-FT instances could not be solved by the DH92 procedure, but show excellent results for the VD06 procedure. Almost all problem instances could be solved to optimality at very low CPU time requirements. As a result of the removal of all within-activity precedence relations (fast tracking) as well as the presence of variable durations (fixed work), optimal schedules often utilize all available resources almost completely. Hence, the initial lower bound LB_0 is often equal to the initial upper bound UB_0, such that no branching is needed.

4.2. Impact of the activity assumptions on project lead time and resource utilization

In this section, we report results for the various activity assumptions and their impact on the quality of the schedule. To that purpose, we rely on the dedicated algorithms for the problem types under study: The DH92 procedure for the RCPSP, the DH96 procedure for the PRCPSP, the VD06 procedures of Section 3 for the PRCPSP-FT and the PDTRTP-FT and the Demeulemeester et al. (2000) procedure (DDH00) for the DTRTP.

The impact of the activity assumptions on the schedule quality has been measured both from a project lead time and a resource utilization point-of-view. The row "Avg. LT" of Table 4 displays the average decrease of total project duration compared to the minimal makespan found by solving the RCPSP problem. The row "Avg. Res" displays the *average resource utilization ratio* (ARUR), defined as the *resource utilization ratio* (Valls, Ballestin, & Quintanilla, 2002) for all resource types averaged over the complete scheduling horizon, i.e.$\mathrm{ARUR} = \frac{1}{f_n|R|}\sum_{k=1}^{|R|}\sum_{i=1}^{n}\frac{W_{ik}}{a_k}$. As an example, the average resource utilization ratio equals $(6+4+4+4+4+4+4+3+3+2)/(9*6) = 62.9\%$ (only one resource type) for the RCPSP schedule of Fig. 1 and 70.8% (80.9%) for the optimal schedule for the PRCPSP (PRCPSP-FT and PDTRTP-FT) of Fig. 3 (4 and 5).

Table 4
The schedule quality for the RCPSP, PRCPSP, PRCPSP-FT, DTRTP and the PDTRTP-FT

Number of activities	10	12	14	16	18	20	Total
RCPSP							
Avg. LT	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
Avg. Res	69.66%	72.30%	75.50%	76.55%	78.00%	79.25%	75.21%
PRCPSP							
Avg. LT	0.47%	0.47%	0.59%	0.51%	0.49%	0.49%	0.50%
Avg. Res	70.05%	72.60%	76.01%	76.98%	78.44%	79.67%	75.63%
PRCPSP-FT							
Avg. LT	18.91%	17.01%	14.76%	14.00%	12.75%	11.85%	14.88%
Avg. Res	85.63%	87.44%	88.55%	89.08%	89.36%	89.60%	88.28%
DTRTP							
Avg. LT	25.25%	23.13%	20.81%	20.20%	19.10%	18.06%	21.09%
Avg. Res	92.18%	93.71%	94.76%	95.61%	95.97%	96.31%	94.76%
PDTRTP-FT							
Avg. LT	26.19%	23.85%	21.37%	21.08%	19.65%	18.72%	21.81%
Avg. Res	93.43%	94.66%	95.99%	96.49%	96.63%	97.18%	95.73%

The results can be summarized as follows. First, allowing pre-emption in the RCPSP has almost no effect on both the lead time and the resource utilization. Hence, the 'task splitting' option of project scheduling software, which results in pre-emptive and often less clear schedules, is no good alternative to improve the schedule quality. Second, the shift from fixed duration activities to fixed work content activities (DTRTP), however, has a major effect on both the lead time (an improvement with approximately 20%) and the resource utilization (from approximately 75% to 92% or more). Hence, the 'fixed work' option should be carefully considered as a default option, since – although resulting in increased CPU time requirements – it has a major beneficial effect on the schedule quality. Third, 'within-activity fast tracking' turns out to have a beneficial effect on the fixed duration activities (PRCPSP-FT), leading to approximately 15% lead time improvement and an 88% resource utilization, but the extra benefits when using fixed work activities (PDTRTP-FT) are relatively small compared to the very efficient schedules found by the DTRTP. Hence, allowing fixed work activities already results in a very efficient schedule, making the within-activity fast tracking a redundant alternative to improve schedule quality.

5. Conclusions

In this paper, we provided computational experiments to measure the impact of three different activity assumptions on the overall quality of the schedule. The three activity assumptions, fixed duration or fixed work, activity pre-emption and fast tracking, have interesting practical applications and are incorporated in most project scheduling software packages. The schedule quality has been measured from both a lead time and a resource utilization point-of-view. All activity assumptions can be considered as relaxations from the strict assumptions of the well-known resource-constrained project scheduling problem (RCPSP).

In this paper, we restricted our research to single resource type projects in order to be in line with the problem formulation of the DTRTP. The results show that all relaxations lead to an increase of the CPU time requirements, which clearly shows that problem specific procedures will be needed to solve practical large-sized problem instances to optimality. Activity pre-emption has only a small positive effect on the schedule quality. The extension to fixed work and/or fast tracking has a major effect on both the project lead time and the resource utilization. The additional effect of fast tracking on the DTRTP is negligible. We believe that the provided insights are valuable for project managers when using commercial project scheduling software packages to help them choosing the activity options and carefully balancing on the trade-off between complexity and schedule quality impact.

Our future intensions are twofold. First, we aim at the construction of efficient meta-heuristic solution procedures to solve the PRCPSP-FT and the PDTRTP-FT where set-up times are incorporated between pre-emptive subactivities. These solution procedures should be able to cope with large-sized and realistic problem settings. Second, we want to extend this approach to a *flexible activity assumptions* problem setting. A problem definition where the activity assumptions can vary for each activity individually would be more realistic. Incorporating more *flexible resource specifications*, such as an option to include renewable resource types with variable availabilities and non-renewable resource types also lies in our future research interest. We would like to note that the goal of the current manuscript was not to present efficient state-of-the-art algorithms for the various RCPSP extensions but rather to show the beneficial impact of activity assumptions on a project's schedule quality by carefully changing the activity assumptions. Hence, we believe that the aforementioned future research will further tighten the bridge between the project scheduling research and the practical algorithms implemented in classical scheduling software.

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.

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. (1996). An efficient optimal solution procedure for the pre-emptive resource-constrained project scheduling problem. *European Journal of Operational Research, 90*, 334–348.

Demeulemeester, E., De Reyck, B., & Herroelen, W. (2000). The discrete time/resource trade-off problem in project networks: A branch-and bound approach. *IIE Transactions, 32*(11), 1059–1069.

Demeulemeester, E., Vanhoucke, M., & Herroelen, W. (2003). A random network generator for activity-on-the-node networks. *Journal of Scheduling, 6*, 13–34.

De Reyck, B., Demeulemeester, E., & Herroelen, W. (1998). Local search methods for the discrete time/resource trade-off problem in project networks. *Naval Research Logistics, 45*(6), 553–578.

Hartmann, S., & Kolisch, R. (2000). Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem. *European Journal of Operational Research, 127*, 394–407.

Herroelen, W., & De Reyck, B. (1999). Phase transitions in project scheduling. *Journal of Operational Research Society, 50*, 148–156.

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 Ph.D. 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., & Hartmann, S. (2006). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research, 174*, 23–37.

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

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

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.

Uyttewaal, E. (2005). *Dynamic scheduling with Microsoft Office Project 2003 – the book by and for professionals*. Florida: J. Ross Publishing.

Valls, V., Ballestin, F., & Quintanilla, S. (2002). A hybrid genetic algorithm for the resource-constrained project scheduling problem with the peak crossover operator. In *Eighth international workshop on project management and scheduling*, pp. 368–371.