Available online at www.sciencedirect.com
ScienceDirect
Computers & Industrial Engineering 54 (2008) 1062–1070
computers & industrial engineering
www.elsevier.com/locate/caie

Setup times and fast tracking in resource-constrained project scheduling

Mario Vanhoucke *

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

Received 24 August 2007; received in revised form 15 November 2007; accepted 16 November 2007
Available online 23 November 2007

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 schedule pre-emptive sub-parts of activities in parallel.

The contribution of the paper is twofold. First, we briefly show that an efficient exact branch-and-bound procedure from the literature to solve the resource-constrained project scheduling problem can be easily adapted to cope with our problem extensions. Second, we extensively test the impact of these pre-emptive extensions to the quality of the schedule from a makespan point-of-view.

Keywords: Project scheduling; Setup times

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 finish–start zero-lag precedence relations between the activities. Furthermore, project execution requires a set of renewable 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 per period. 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 start times $(s_1, \ldots, s_n)$, and implies an n-vector of finish times $(f_1, \ldots, f_n)$ 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 sat-

* Address: Faculty of Economics and Business Administration, Ghent University, Ghent, Belgium.
E-mail address: mario.vanhoucke@ugent.be

0360-8352/$ - see front matter © 2007 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cie.2007.11.008

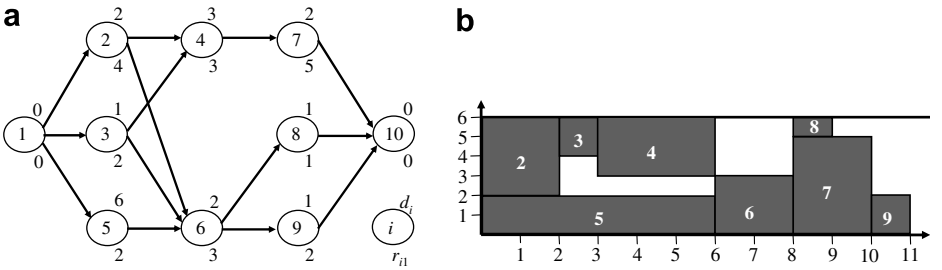

Fig. 1. An example project with a corresponding optimal RCPSP schedule.

isfied, and *optimal* if the project makespan f_n is minimized. Fig. 1a 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 U. The optimal schedule is displayed in Fig. 1b, and has a minimal makespan of 11 time U.

The research on the RCPSP has widely expanded over the last few decades, and reviews can be found in Icmeli, Erenguc, and Zappe (1993), Özdamar and Ulusoy (1995), Herroelen, De Reyck, and Demeulemeester (1998), Brucker, Drexl, Möhring, Neumann, and Pesch (1999) and Kolisch and Padman (2001). In the literature, various RCPSP extensions have been proposed (see e.g., the extensions described in the papers by Brucker et al., 1999; Herroelen, Demeulemeester, & De Reyck, 1999). Recently, Vanhoucke and Debels (2008) have extended the basic RCPSP to the pre-emptive resource-constrained project scheduling problem with fast tracking (PRCPSP-FT). Although the concept of activity pre-emption was not new, their specific definition of activity overlapping or activity fast tracking was never studied before. In their paper, they extended the RCPSP to the *within-activity* fast-tracking option which allows the pre-emptive sub-parts of an activity to be executed in parallel, and hence they did not allow overlaps *between* activities by relaxing their technological precedence relations.

In the current manuscript, we extend this PRCPSP-FT procedure to the presence of setup times between pre-emptive sub-parts of activities. Since Vanhoucke and Debels (2008) obviously reported a clear positive influence of activity pre-emption and fast tracking on the project makespan and resource utilization, we investigate in the current manuscript whether setup times between pre-emptive sub-parts still lead to positive makespan and renewable resource utilization reductions compared to the basic RCPSP. The aim and contribution of this paper is as follows. First, we present a problem description and briefly describe the straightforward extension to the efficient branch-and-bound procedures of Demeulemeester and Herroelen (1992) and Vanhoucke and Debels (2008) in order to solve the PRCPSP-FT with setup times. Our main intention is to show by means of computational experiments that the procedure can produce promising results within an acceptable time limit. We show that, while the positive influence of activity pre-emption and fast traking without setup times on the project makespan is intiutively clear (Vanhoucke & Debels, 2008), the introduction of setup times does not always neutralizes this positive effect. The effect of various settings of setup times is tested on a large set of project instances, and the effect on the project makespan is reported. In the next section, the PRCPSP-FT problem formulation with setup times is given and the modification to the branch-and-bound algorithm of Vanhoucke and Debels (2008) is briefly discussed.

2. Problem formulation

The pre-emptive resource-constrained project scheduling problem with fast tracking (PRCPSP-FT) has been initially proposed by Vanhoucke and Debels (2008) to investigate the influence of within-activity fast tracking on the project makespan. Although Demeulemeester and Herroelen (1996a) have shown that activity pre-emption seldom has a significant impact on the total project makespan compared to the RCPSP makespan, Vanhoucke and Debels (2008) have concluded that activity pre-emption *and* activity fast tracking of these pre-empted sub-parts of activities can lead to large makespan reductions. These authors did not take preparation or setup work into account and simply assumed that activities can be started, pre-empted and/or

fast tracked at no extra time or cost. However, in our daily scheduling consulting activities (mainly in the construction industry), we noticed that most project planners make a distinction in an activity duration between the actual work (in days) and the time needed to prepare the actual work (currently called setup time) such as installation time, releasing resources from other sites, etc. Fig. 2a displays details about activity 5 of Fig. 1 with a total duration d_i of 6 days (horizontal direction of the activity), consisting of 1 setup time day and 5 remaining days of actual work. The project activity needs to be executed by teams of two persons working together (vertical axis of the activity). Fig. 2b shows that activity pre-emption results in a renewed setup time due to the interruption in time. Consequently, the total setup time amounts to two days while the actual work remains 5 days (which is known in literature as pre-emption-resume). The activity fast-tracking option in Fig. 2c is the result of assigning two teams (each containing two persons working in parallel) on the interruptive parts of the activity, and hence, the two pre-emptive parts can now be executed in parallel. Consequently, this fast-tracking option removes precedence relations between sub-parts of pre-empted activities and increases the number of execution scenarios for each activity of the project.

In the current manuscript, we study the PRCPSP-FT as option c of Fig. 2 where activity pre-emptions and/or within-activity fast trackings are only allowed at the expense of an extra setup cost. Hence, the original defined activity durations d_i consist of both a single setup time t_i for starting the activity and a remaining processing time. The setup time component includes activity preparations such as equipping, resetting, changing, positioning, cleaning and warming up (Mika, Waligora, & Weglarz, 2006). This setup time is additionally added to the total duration each time the activity is interrupted. This problem formulation is highly relevant for projects where multiple resource units (e.g., teams of people or a combination of machines) are assigned to project activities. In this case, activity resource requirements are often defined as the minimal amount of resource units required to perform the activity, and hence, the duplication of this minimal amount of resource units allows the fast tracking of sub-parts of activities. Obviously, the duplication and corresponding fast-tracking decision often involves extra setup time, as discussed before. The use of activity fast tracking with setup times can be illustrated by the Westerscheldetunnel project discussed in Vanhoucke (2006, 2007). In this highly complex project, a freezer machine is needed to drill emergency connections between two lanes of a tunnel constructed in The Netherlands to connect two sides of a river. However, multiple freezer machines allow the drilling of the connections starting from both sides (i.e., both lanes) and hence, results in overlaps of sub-parts of activities (in this case, the activity "drill a connection"). Obviously, installing multiple freezer machines at both sides of the new connection between the lanes doubles the original setup times. More information on the Westerscheldetunnel project can be obtained on www.westerscheldetunnel.nl.

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 interrup-

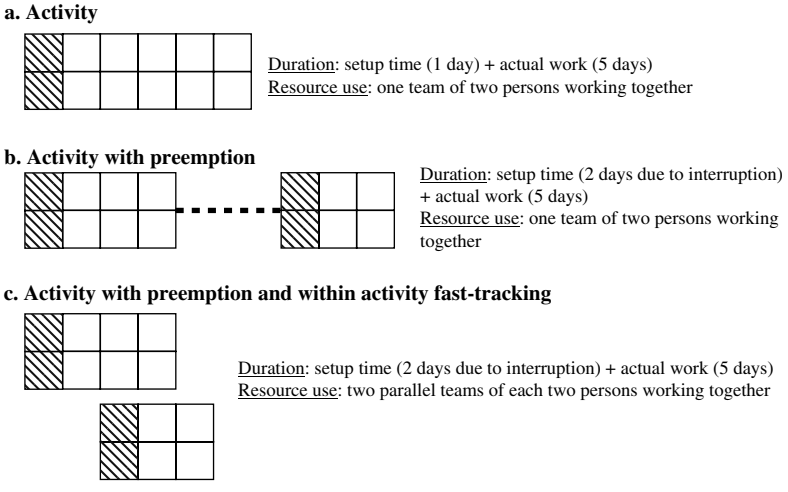

Fig. 2. Activity 5 of Fig. 1 contains minimum 1 day setup time and 5 days remaining actual work.

tions 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.

Demeulemeester and Herroelen (1996a) have already shown that any RCPSP network can be easily transformed to a PRCPSP network by splitting each original activity i into d_i sub-activities i_s with a duration $d_{i_s} = 1$ and a resource requirement $r_{i_s k} = r_{ik}$. A similar transformation holds for the PRCPSP-FT, but all precedence relations between sub-activities i_s of a similar activity i need to be removed. Fig. 3 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 Fig. 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 Fig. 1. In Fig. 3, this activity is pre-empted resulting in a total duration of 8 time U instead of 6, due to the extra setup time before sub-activity 5_3. The activity labels have been shown in black, while the sub-activity numbers have been displayed in grey. The shaded areas represent the use of resources due to setup times. Despite the pre-emptive setup times, the minimum makespan could be reduced from 11 to 10 time U.

The foundation of the solution procedure lies on the depth-first procedure of Demeulemeester and Herroelen (1992) to solve the RCPSP (no pre-emption, no fast tracking and no setup times). Vanhoucke and Debels (2008) have extended this branch-and-bound procedure in order to cope with activity pre-emption and fast tracking (i.e., the PRCPSP-FT without setup times). More precisely, they simplified the minimal delaying alternatives theorem in order to cope with sub-activities and presented two effective lower bounds to improve the efficiency of the procedure. In this section, we briefly discuss the adaptations to the latter procedure in order to cope with setup times. These extensions are often straightforward and boil down to modifications in the existing dominance rules and lower bound calculations.

All dominance rules originally proposed by Demeulemeester and Herroelen (1992) have been used and slightly adapted in order to cope with setup times. We mentioned previously that the minimal delaying alter-

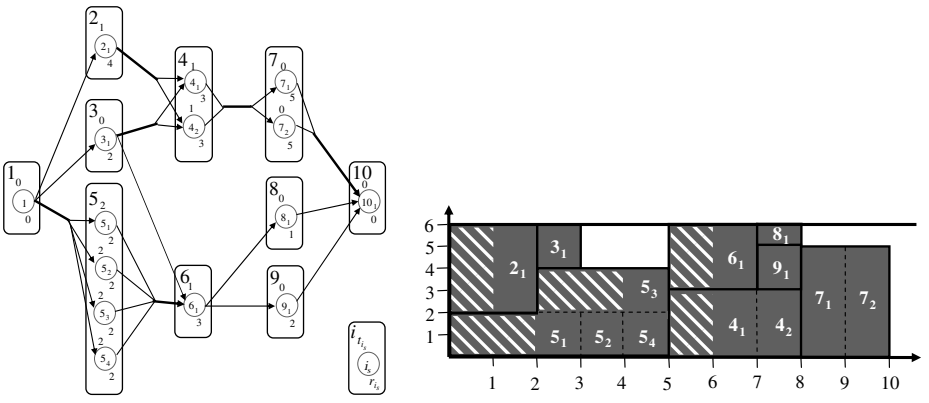

Fig. 3. The example sub-activity network and optimal schedule for the PRCPSP-FT with setup times.

natives theorem has already been simplified by Vanhoucke and Debels (2008) in order to cope with sub-activities. In line with Demeulemeester and Herroelen (1992), the so-called cutset dominance rule still accounts for the largest reduction in the number of nodes of the branch-and-bound tree. The lower bound calculations basically exploit the observation that the presence of fast tracking in resource-constrained project scheduling leads to an efficient use of resources (see Vanhoucke & Debels, 2008). Hence, in this algorithm, we use an extended version of the resource-based lower bound that calculates the minimal remaining time of a partially scheduled project based on both critical path based and renewable resource based information. The latter assumes that resource availability can be used by either activity consumption (as in the original RCPSP and its pre-emptive version) and/or setup times due to pre-emption of activities. Both the branch-and-bound approach of the current manuscript and the procedure of Vanhoucke and Debels (2008) follow the same logic as the original DH92 branch-and-bound procedure, which shows that the latter procedure can be relatively easily and effectively extended to broader problem settings. Algorithmic details can be found in Debels and Vanhoucke (2006) which can be downloaded from www.feb.ugent.be.

3. Computational results

In this section, we report computational results for our procedure programmed in Visual C++ 6.0 and tested on an Acer Travelmate 634LC with a Pentium IV 1.8 GHz processor. We present computational results on a randomly generated dataset with up to 30 project activities and show the relative decrease in the project makespan when allowing activity pre-emption and fast tracking. The testset has been generated by RanGen (Demeulemeester, Vanhoucke, & Herroelen, 2003) and contains projects with 10, 15, 20, 25 or 30 non-dummy activities with an order strength (Mastor, 1970) and a resource-constrainedness (Patterson, 1976) varying from 0.25, 0.50 to 0.75. All project instances require four renewable resource types with an availability of 10 U and the activity durations have been chosen randomly between 1 and 10. 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 sub-activity. We have generated activity setup times under five settings as 0%, 25%, 50%, 75% or 100% of the original activity duration minus one. Consequently, the activity setup times and remaining activity durations have been calculated as follows: we subtract the generated setup time from the original activity duration to calculate the remaining activity duration. Hence, the sum of the activity setup time and its remaining duration is always equal to the original duration of the project network instance and the remaining duration is minimum one. This approach allows us to measure the impact of pre-emptive fast tracking with setup times on the schedule quality by comparing it with the RCPSP makespan. The testset leads to three different scenarios, 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 Vanhoucke and Debels (2008).
(2) If the setup times for the activities are set at a value t from 25% to 75%, then the remaining durations of the activities is greater than 1 and lower than its original duration. The minimal makespan will lie between the RCPSP and the PRCPSP-FT minimal makespan.
(3) If the setup time of each activity is set at $t = 100\%$, then each remaining activity duration is equal to 1. In this case, activity pre-emption is impossible, and hence, the problem boils down to the basic RCPSP.

Since each setting contains 10 problem instances, the problem set contains $5 * 3 * 3 * 5 * 10 = 2250$ project instances. Table 1 displays detailed results for all problem instances with the different settings for the setup time and the number of project activities, each split up for the different values of the order strength and the resource constrainedness. Each cell contains three relevant numbers, representing the average CPU time (Avg.CPU, in seconds), the percentage of problems solved to optimality (%Opt, truncated after 1 min CPU time) and the average decrease of the total project makespan compared to the minimal RCPSP makespan (Avg.RD, obtained by the DH92 procedure). The last Avg.RD number measures the average relative project makespan improvement when introducing activity pre-emption and fast tracking with setup times.

Table 1
Computational results for various problem types

RC/OS	10			15			20			25			30			Global	
	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75		
$t = 0\%$																	
0.25	0.169	0.002	0.002	2.149	1.326	0.004	36.480	7.051	0.162	54.144	30.312	0.363	60.000	42.244	14.774	16.612	Avg.CPU
	100%	100%	100%	100%	100%	100%	40%	90%	100%	10%	60%	100%	0%	30%	80%	74%	%Opt
	9.22%	20.61%	28.56%	8.46%	13.88%	32.41%	4.58%	9.32%	23.05%	2.05%	8.17%	22.41%	1.03%	4.23%	16.74%	13.65%	Avg.RD
0.50	0.000	0.000	0.000	0.040	0.002	0.000	0.338	0.002	0.000	6.219	0.017	0.000	42.035	6.510	0.004	3.678	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	90%	100%	100%	30%	90%	100%	94%	%Opt
	1.66%	2.20%	3.92%	0.78%	1.52%	3.59%	1.15%	2.05%	2.26%	1.18%	2.20%	2.25%	0.63%	1.72%	2.87%	2.00%	Avg.RD
0.75	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	%Opt
	1.19%	1.93%	2.10%	0.00%	0.85%	0.72%	0.87%	0.41%	1.40%	0.51%	0.86%	1.39%	0.11%	0.20%	0.74%	0.89%	Avg.RD
$t = 25\%$																	
0.25	0.005	0.002	0.002	0.136	0.816	0.002	28.739	11.919	0.749	48.150	36.797	0.872	58.814	52.372	7.376	16.450	Avg.CPU
	100%	100%	100%	100%	100%	100%	70%	90%	100%	20%	60%	100%	10%	20%	100%	78%	%Opt
	2.48%	10.74%	17.37%	2.06%	6.61%	19.58%	0.43%	2.29%	11.43%	0.00%	1.83%	12.23%	0.00%	0.00%	8.92%	6.40%	Avg.RD
0.50	0.002	0.000	0.000	0.014	0.000	0.000	0.366	0.000	0.000	18.281	0.112	0.002	18.360	0.493	0.002	2.509	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	70%	100%	100%	70%	100%	100%	96%	%Opt
	1.01%	1.83%	2.35%	0.78%	0.76%	2.11%	0.39%	1.13%	1.00%	0.34%	1.32%	1.13%	0.27%	0.93%	1.96%	1.15%	Avg.RD
0.75	0.000	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.000	0.002	0.002	0.000	0.000	0.002	0.000	0.001	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	%Opt
	0.86%	0.81%	1.04%	0.00%	0.67%	0.37%	0.58%	0.25%	1.09%	0.24%	0.38%	0.75%	0.00%	0.00%	0.65%	0.51%	Avg.RD
$t = 50\%$																	
0.25	0.002	0.002	0.002	0.106	0.150	0.000	16.133	1.494	0.220	42.945	22.603	0.040	50.948	36.975	0.668	11.486	Avg.CPU
	100%	100%	100%	100%	100%	100%	90%	100%	100%	30%	70%	100%	20%	50%	100%	84%	%Opt
	0.91%	5.70%	9.61%	0.67%	4.00%	12.50%	0.96%	0.89%	5.71%	0.00%	1.48%	6.39%	0.00%	0.36%	5.33%	3.63%	Avg.RD
0.50	0.002	0.000	0.000	0.005	0.000	0.000	0.025	0.000	0.000	0.173	0.011	0.002	12.052	0.055	0.000	0.822	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	80%	100%	100%	99%	%Opt
	0.30%	0.00%	0.85%	0.24%	0.23%	0.95%	0.18%	0.34%	0.32%	0.15%	0.41%	0.41%	0.00%	0.40%	0.99%	0.38%	Avg.RD

0.75	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	%Opt
	0.60%	0.00%	0.25%	0.00%	0.33%	0.17%	0.28%	0.13%	0.61%	0.13%	0.38%	0.50%	0.00%	0.00%	0.32%	0.25%	Avg.RD
$t = 75\%$																	
0.25	0.000	0.000	0.000	0.030	0.003	0.000	1.872	0.076	0.000	29.989	7.898	0.002	44.386	19.859	0.025	6.943	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	60%	90%	100%	40%	80%	100%	91%	%Opt
	0.00%	2.34%	4.14%	0.67%	2.00%	5.30%	0.43%	0.42%	1.52%	0.00%	0.36%	2.94%	0.00%	0.36%	2.54%	1.53%	Avg.RD
0.50	0.000	0.002	0.000	0.002	0.000	0.000	0.006	0.000	0.002	0.032	0.004	0.000	0.081	0.019	0.000	0.010	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	%Opt
	0.30%	0.00%	0.26%	0.24%	0.00%	0.45%	0.00%	0.34%	0.14%	0.00%	0.26%	0.14%	0.00%	0.14%	0.33%	0.17%	Avg.RD
0.75	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	%Opt
	0.26%	0.00%	0.25%	0.00%	0.16%	0.00%	0.14%	0.13%	0.29%	0.00%	0.00%	0.11%	0.00%	0.00%	0.32%	0.11%	Avg.RD
$t = 100\%$																	
0.25	0.000	0.000	0.000	0.010	0.000	0.000	0.644	0.024	0.000	22.176	2.091	0.000	41.550	7.752	0.002	4.950	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	70%	100%	100%	40%	100%	100%	94%	%Opt
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	Avg.RD
0.50	0.000	0.000	0.000	0.002	0.002	0.000	0.004	0.000	0.000	0.009	0.002	0.000	0.022	0.004	0.000	0.003	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	%Opt
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0 00%	Avg.RD
0.75	0.002	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	Avg.CPU
	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	100%	%Opt
	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0. 00%	0.00%	0.00%	0.00%	0.00%	0.00%	Avg.RD
Global	0.012	0.001	0.000	0.166	0.153	0.000	5.641	1.371	0.076	14.808	6.657	0.085	21.883	11.086	1.523	4.231	Avg.CPU
	100%	100%	100%	100%	100%	100%	93%	99%	100%	77%	92%	100%	66%	85%	99%	94%	%Opt
	1.25%	3.08%	4.71%	0.93%	2.07%	5.21%	0.67%	1.18%	3.25%	0.31%	1.18%	3.38%	0.14%	0.56%	2.78%	2.05%	Avg.RD

The results in the table can be summarized as follows. First, the table shows 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 the basic RCPSP. Moreover, the table shows that many instances with up to 30 activities can be solved within reasonable amount of time to optimality. Second, the table reveals the positive effect on the project makespan reduction for increasing OS values and the negative effect for increasing RC values. Hence, a network with higher density (measured by the amount of precedence relations and the resulting higher OS value) benefits more from activity pre-emption and fast tracking than lower OS value networks which already allows a high degree of flexibility. On the contrary, higher resource tightness prevents further makespan improvements by pre-empting or fast-tracking project activities. Last, it is worth mentioning that the option to fast track has a major effect on the project makespan, even with high values for the setup times. As an example, the PRCPSP-FT with setup times up to 75% of the original duration still leads to average improvements varying from 0.11% to 1.53% on average. This illustrates that the presence of relatively high setup times does not prevent project makespan reductions when allowing activity pre-emption and/or fast tracking. 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 makespan.

4. Conclusions

The previous research of the pre-emptive 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 makespan. However, a recently studied pre-emptive extension, known as the pre-emptive resource-constrained project scheduling problem with fast tracking (PRCPSP-FT, Vanhoucke & Debels, 2008), allows that these pre-emptive sub-activities can be executed in parallel, and leads to a major decrease in the total project makespan.

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 used a straightforward extension of the branch-and-bound 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 makespan, 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.

References

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. Pre-emptive resource-constrained project scheduling with setup times, Working Paper 06/391, Ghent University.

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 exact solution procedure for the pre-emptive 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., & 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.

Herroelen, W., Demeulemeester, E., & De Reyck, B. (1999). A classification scheme for project scheduling. In J. Weglarz (Ed.), *Project scheduling – Recent models. Algorithms and applications, International series in operations research and management science* (Vol. 14, pp. 77–106). Boston: Kluwer Academic Publishers.

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., 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, pp. 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.

Vanhoucke, M., & Debels, D. (2008). The impact of various activity assumptions on the lead-time and resource utilization of resource-constrained projects. *Computers and Industrial Engineering, 54*, 140–154.

Vanhoucke, M. (2006). Work continuity constraints in project scheduling. *Journal of Construction Engineering and Management, 132*, 14–25.

Vanhoucke, M. (2007). Work continuity optimization for the Westerscheldetunnel project in The Netherlands. *Tijdschrift voor Economie en Management, 52*, 435–449.