Math. Program., Ser. A 103, 515–539 (2005)

Digital Object Identifier (DOI) 10.1007/s10107-004-0551-6

Tamás Kis

A branch-and-cut algorithm for scheduling of projects with variable-intensity activities

Received: October 17, 2002 / Accepted: July 15, 2004
Published online: February 9, 2005 – © Springer-Verlag 2005

Abstract. In this paper we study a resource constrained project scheduling problem in which the resource usage of each activity may vary over time proportionally to its varying intensity. We formalize the problem by means of a mixed integer-linear program, prove that feasible solution existence is NP-complete in the strong sense and propose a branch-and-cut algorithm for finding optimal solutions. To this end, we provide a complete description of the polytope of feasible intensity assignments to two variable-intensity activities connected by a precedence constraint along with a fast separation algorithm. A computational evaluation confirms the effectiveness of our method on various benchmark instances.

Keywords: Project scheduling – Network flows, Polyhedral combinatorics – Branch-and-cut

1. Introduction

Most results on the resource-constrained project scheduling problem (RCPSP) assume *fixed* activity durations and a *constant rate* of resource usage while performing every activity. Extensions to RCPSP relaxing at least one of these assumptions comprise *preemption of activity execution*, the *discrete time/resource trade-off*, the *time/cost trade-off* and the *multi-mode* resource-constrained project scheduling problems (see e.g Herroelen et al. [13], Brucker et al. [3] and Demeulemeester and Herroelen [6]). We will study a further extension, called *RCPSVP*, in which the intensity of each activity may vary over time and the resource-usage is proportional to the intensity.

An instance of the problem is given by a finite set N of *activities*, a finite set R of *continuously divisible renewable resources* and a directed acyclic graph $D = (N, A)$ representing *precedence constraints among the activities*. The time horizon is divided into T periods, which will be indexed as $t = 1, \dots, T$. For each activity $i \in N$ a release time r^i and a deadline d^i specify an interval of time periods $\{r^i, \dots, d^i\}$ in which the activity must entirely be executed, where $1 \le r^i \le d^i \le T$. In each time period $t \in \{r^i, \dots, d^i\}$ at most an $a^i \le 1$ fraction of activity i may be completed. Activity i requires a total of q^i_k units of resource k, for each $k \in R$. If the (chosen) intensity of activity i is x^i_t in time period t, where $0 \le x^i_t \le a^i$ and $\sum_t x^i_t = 1$ hold, then it requires $q^i_k \cdot x^i_t$ units of resource k in that period. Each resource $k \in R$ has an *internal capacity* of b^k_t units that is available free of charge and it has an additional *external capacity* of $\bar{b}^k_t$ units at the expense of c^k_t for each external unit used.

T. Kis: Computer and Automation Research Institute, Hungarian Academy of Sciences, 1111 Budapest, Kende utca 13-17, Hungary. e-mail: tamas.kis@sztaki.hu

The scheduling problem consists of determining for each activity i an intensity x_t^i in each time period $t \in \{r^i, \dots, d^i\}$ such that $0 \le x_t^i \le a^i$, $\sum_{t=r^i}^{d^i} x_t^i = 1$, the precedence constraints among the activities are fulfilled, the resource demand does not exceed the resource availability (internal + external) in any time period, and the total cost of using external capacity is minimized.

Example 1. To illustrate the above concepts consider a problem instance with 2 renewable resources and 3 activities. All activities have the same time window $r^i = 1, \dots, d^i = 3$, while the maximum intensities are $a^1 = a^2 = 1/2$, and $a^3 = 1$. Activities 1 and 2 require $q_1^1 = 1$ and $q_1^2 = 2$ units of resource 1, respectively, and activity 3 requires $q_2^3 = 1$ units of resource 2. The internal capacity of both resources is 1, and there is no external capacity available. As for precedences, activity 1 must precede activity 3.

The unique solution to this instance is given by the intensity assignments $x^1 = (1/2, 1/2, 0)$, $x^2 = (1/4, 1/4, 1/2)$ and $x^3 = (0, 0, 1)$, respectively, where the t-th component of x^i represents the intensity of activity i in time period t. Fig. 1 depicts the resource usage of the activities with respect to x. □

Project scheduling with variable intensity activities has been studied by several authors. Weglarz [26] proposes a continuous time model for allocating a single, doubly constrained resource to a set of activities over time, the objective being to minimize project duration. The single resource may be allocated to activities in arbitrary amounts within given intervals. The *performance-speed* or *intensity* of each activity is determined by a continuous, non-decreasing function of the amount of resource allocated to it at any moment. Weglarz provides several analytical results and discusses a few numerical examples. Leachman et al. [19] study a discrete time version of the model of Weglarz in which all the different types of resources required by an activity are applied proportionally to the (varying) intensity of the activity. The authors propose a heuristic algorithm for minimizing the makespan. For modeling and solving the special case where all lower bounds on activity intensities are 0, Tavares [24], [25] suggests a non-linear program, containing products of decision variables, which is solved by a standard non-linear optimization software. Finally, Hans [11] discusses in Chapter 6 of his Ph.D. thesis exactly the same model which is the topic of this paper. His approach is branch-and-price, which is dual to our branch-and-cut approach. Hans obtains an initial upper bound by various constructive heuristics based on LP rounding techniques and by iterative improvement. In the column generation phase he uses a fast pricing algorithm and also rounding heuristics to obtain feasible solutions.

In this paper we propose a new mixed integer-linear program formulation of RCPSVP, where the novelty lies in the modeling of the precedence constraints (Section 2). It is also

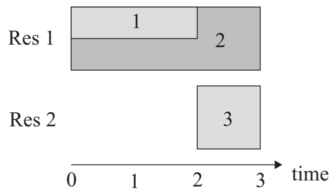

Fig. 1. Resource usage of activities.

shown how other objective functions fit in our framework. The pro's and con's of the alternative formulations are discussed in Section 3. In Section 4 we prove that RCPSVP is NP-hard in the strong sense. In Section 5, we analyze the polytope K^{ij} of all feasible intensity assignments to a pair of variable-intensity activities i and j with $(i, j) \in A$. This polytope is decomposed into two smaller dimensional polytopes which, in fact, are instances of the same polytope K. Valid inequalities for K along with a separation algorithm is provided next. In Section 6 we show that the inequalities found previously completely describe the convex hull of K and also that of K^{ij} and establish conditions under which the inequalities represent facets of these polytopes. In Section 7 we describe a branch-and-cut algorithm based on our polyhedral results and summarize various computational experiments with it. In particular, we compare our method to that of Hans for RCPSVP and also a variant of our method to the approach of Tavares for minimizing the makespan.

2. New formulations

Since resources must be allocated to activities over discrete time periods, it is natural to use a time-indexed formulation with the following variables:

x_t^i	=	intensity of activity i in time period t,
z_t^i	=	mask of activity i in time period t. $z_t^i = 1$ if activity i may be executed in time period t and 0 otherwise,
y_t^k	=	external capacity of resource k used in time period t.

Let $p^i = \lceil 1/a^i \rceil$ denote the minimum time to complete activity i. Now suppose $(i, j) \in A$, i.e., activity i must complete before activity j may start. Since activity i cannot complete before $r^i + p^i$, w.l.o.g. we may assume that $r^i + p^i \leq r^j$. Similarly, activity i must complete not later than $d^j - p^j$, that is, we may assume that $d^i \leq d^j - p^j$. After these preliminaries, the *weak formulation* for RCPSVP is as follows:

$$\text{Minimize} \sum_{k \in R} \sum_{t \in \{1,\dots,T\}} c_t^k \cdot y_t^k$$

subject to

$$\sum_{t=r^i}^{d^i} x_t^i = 1, \quad i \in N, \tag{1a}$$

$$x_t^i \leq a^i \cdot z_t^i, \quad i \in N,\ t \in \{r^i + p^i, \dots, d^i\} \tag{1b}$$

$$x_t^j \leq a^j \cdot (1 - z_t^i), \quad (i, j) \in A,\ t \in \{r^j, \dots, d^i\} \tag{1c}$$

$$z_t^i \geq z_{t+1}^i, \quad i \in N,\ t \in \{r^i + p^i, \dots, d^i - 1\}, \tag{1d}$$

$$\sum_{i: q_k^i > 0,\ \{r^i, \dots, d^i\} \ni t} q_k^i \cdot x_t^i \leq b_t^k + y_t^k, \quad k \in R,\ t \in \{1, \dots, T\}, \tag{1e}$$

$$0 \leq y_t^k \leq \overline{b}_t^k, \quad k \in R,\ t \in \{1, \dots, T\} \tag{1f}$$

$$0 \leq x_t^i \leq a^i, \quad i \in N,\ t \in \{r^i, \dots, d^i\} \tag{1g}$$

$$z_t^i \in \{0, 1\}, \quad i \in N,\ t \in \{r^i + p^i, \dots, d^i\} \tag{1h}$$

The objective is to minimize the weighted sum of external capacity used. By (1a), every activity must entirely be executed between its release time and deadline. The precedences between the activities are forced by (1b)–(1d). Namely, activity i can be executed at time t only if z_t^i is 1 (cf. (1b)). However, when $z_t^i = 1$, no activity j with $(i, j) \in A$ may be executed, due to (1c). Finally, (1d) ensures that there is a time point $t_0 \in \{r^i + p^i, \dots, d^i + 1\}$ such that $z_t^i = 1$ for all $t \in \{r^i + p^i, \dots, t_0 - 1\}$ and $z_t^i = 0$ for all $t \in \{t_0, \dots, d^i\}$, hence, activity i must complete before activity j may start. Since activity i cannot complete before $r^i + p^i$, we have no z_t^i variables for $t \in \{r^i, \dots, r^i + p^i - 1\}$. The external capacity y_t^k of resource k used in time period t is determined by (1e) and it cannot be more than $\overline{b}_t^k$, due to (1f). Finally, the domains of the variables x_t^i and z_t^i are given by (1g) and (1h), respectively.

If activity i has no successors in D then z_t^i can be set to 1 for all $t \in \{r^i + p^i, \dots, d^i\}$, and similarly y_t^k can be set to 0 if no activity may require resource k at time t, i.e., $t \notin \{r^i, \dots, d^i\}$ for all $i \in N$ with $q_k^i > 0$. A further reduction is possible for $(i, j) \in A$. Since i precedes j, the difference $z_t^j - z_t^i$ must be 0 or 1 in any feasible solution. Hence, the inequalities $x_t^j \le a^j \cdot (z_t^j - z_t^i)$, $t \in \{r^j + p^j, \dots, d^i\}$, are satisfied by all feasible solutions and are stronger than (1b) for activity j and (1c) for (i, j) together. Therefore, we may replace (1b) and (1c) by the following set of inequalities:

$$x_t^j \le \begin{cases} a^j \cdot (1 - z_t^i), & t \in \{r^j, \dots, \min\{r^j + p^j - 1, d^i\}\}, \\ a^j \cdot (z_t^j - z_t^i), & t \in \{r^j + p^j, \dots, d^i\}, \\ a^j \cdot z_t^j, & t \in \{\max\{d^i + 1, r^j + p^j\}, \dots, d^j\}. \end{cases} \qquad (i, j) \in A \qquad \text{(1bc)}$$

Notice that some of the intervals may be empty depending on the particular d^i and r^j values. We call the resulting program the *strong formulation*.

Besides minimizing the cost of using external resources, our model is suitable for other criteria as well. In the following three problems the external resource capacities are neglected. The *project duration* or *makespan* can be minimized by finding the smallest T (using dichotomic search between some lower and upper bounds), such that when setting all activity deadlines to T, the linear program (1) has a feasible solution. When a due-date $\tilde{d}^i$ is specified for each activity i, we may minimize the *maximum tardiness* by finding the smallest $T_{\max}$ such that system (1) admits a feasible solution with activity deadlines $d^i = \tilde{d}^i + T_{\max}$. Again, the minimum value of $T_{\max}$ can be found by dichotomic search. In addition, when weights w^i are also given, we can minimize the *weighted tardiness* $\sum_i w^i(\max\{0, C^i - \tilde{d}^i\})$, where C^i is the completion time of activity i. Namely, define new weights w_t^i as follows: $w_t^i = 0$ if $t \le \tilde{d}^i$ and $w_t^i = w^i$ for all $t \in \{\tilde{d}^i + 1, \dots, T\}$. Then, the minimum value of $\sum w_t^i \cdot z_t^i$ is the minimum weighted tardiness with respect to time horizon T.

The polyhedral results and the branch-and-cut approach presented in this paper can be used to solve the RCPSVP with respect to any of the above optimization criteria.

3. Comparison of alternative formulations

In this section we discuss the advantages and disadvantages of the alternative formulations for RCPSVP, the emphasis being on integer-linear programming approaches. As all approaches model the resource constraints in essentially the same way, we focus on the modeling of precedence constraints.

Suppose activity i must precede activity j, i.e., $(i, j) \in A$. Tavares [25] models this situation by a set of constraints equivalent to the following:

$$\left(\sum_{t=t_0}^{d^i} x_t^i\right) x_{t_0}^j = 0, \quad \forall\, t_0 \in \{r^j, \dots, d^j\}.$$

Tavares shows that the project duration can be minimized by defining new auxiliary variables z_t with $0 \le z_t \le 1$, $t \in \{1, \dots, T\}$, and adding the constraints

$$\left(\sum_{i \in N} \sum_{t=t_0}^{T} x_t^i\right) z_{t_0} = 0, \quad \forall\, t_0 \in \{1, \dots, T\}.$$

to the model. Notice that if $z_{t_0} > 0$, then no activity can be performed after t_0. Therefore, the above constraints along with the objective $\min(T - \sum_t z_t)$ express the makespan minimization problem. Since the constraints are clearly non-linear in the variables x and z, Tavares used a non-linear optimization software for solving the problem w.r.t. the makespan objective.

In contrast, Hans [11] models the precedence constraints using a set of binary vectors $\{\beta^h \in \{0, 1\}^{|N| \times T} \mid h \in \Pi\}$ consisting of the supports of all feasible intensity assignments to the activities. Although Hans considered distinct projects among which there were no precedence constraints, to simplify notation we assume that all activities belong to the same project. Notice that a binary vector $\beta \in \{0, 1\}^{|N| \times T}$ is the support of a feasible intensity assignment if and only if $\sum_{t=1}^{T} \beta_{i,t} \ge p^i$, $\min\{t \mid \beta_{i,t} = 1\} \ge r^i$, $\max\{t \mid \beta_{i,t} = 1\} \le d^i$, and if $(i, j) \in A$, then $\max\{t \mid \beta_{i,t} = 1\} < \min\{t \mid \beta_{j,t} = 1\}$. Clearly, precisely one vector β^h must be chosen. To this end, Hans has introduced new binary variables z_h, $h \in \Pi$, together with the following constraints:

$$\sum_{h \in \Pi} z_h = 1,$$

$$z_h \in \{0, 1\}, \quad h \in \Pi,$$

$$0 \le x_t^i \le a^i \left(\sum_{h \in \Pi} \beta_{i,t}^h z_h\right), \quad i \in N,\ t \in \{r^i, \dots, T\}$$

The first two constraints ensure that exactly one vector β^h is chosen. The third one specifies that x_t^i is either 0, or is between 0 and a^i, depending on whether $\beta_{i,t}^h$ is 0 or 1. In addition, Hans' formulation also contains constraints equivalent to (1a) and (1e), and instead of (1f) it has $y_t^k \ge 0$, and $\sum_k y_t^k \le \overline{b}^k$, for all t. As the size of Π can be enormous, column generation is the only viable approach to handle this formulation.

The primary advantage is that any objective function which depends linearly on the cost associated with the vector β^h, whatever this cost be, can be optimized by the same solver, provided that the pricing problem can be solved efficiently. The main drawback is that a huge number of columns must be handled, and enlarging all activity deadlines by only one time period may multiply the size of Π which may increase considerably the running time of the pricing algorithm.

In contrast, if all activity deadlines are increased by one time period in our formulation, the number of variables grows only by $O(|N|+|R|)$ and the increase in the number of constraints is in $O(|N|+|R|+|A|)$. The drawback is that good cutting planes must be supplied to the solver. Nevertheless, for the objective function studied in this paper our method is competitive with that of Hans, see Section 7.1.

4. Computational complexity

In this section we will prove that RCPSVP is NP-complete in the strong sense. To this end, we will show that RCPSVP contains the preemptive flowshop scheduling problem (PFSP) as a special case. As the latter problem has been shown NP-complete in the strong sense by Gonzalez and Sahni [9], our claim follows. For fundamental definitions and results of scheduling theory see e.g. Graham et al. [10] and Blazewicz et al. [2].

Recall that in a preemptive flowshop a finite set of *jobs* must be processed by a finite set of *processors*. A processor can process at most one job at a time. The *processing time* of job j on processor k is given by a positive integer number $\pi_{j,k}$ specifying that job j must be processed for exactly $\pi_{j,k}$ time units on processor k. If $\pi_{j,k} = 0$ then job j is not processed by processor k. The processing of a job may be interrupted at any (integral) time point and resumed later. Each job j must visit the processors in the same order, that is, if $\pi_{j,k} > 0$, $\pi_{j,\ell} > 0$ and k precedes ℓ in the order of processors, then job j must be processed for a total of $\pi_{j,k}$ time units on processor k before its processing may be started on processor ℓ. All jobs can be started at time 1 and the question is wether all jobs can be completed by a given time point C.

A solution σ to the PFSP specifies the time points in which a job is being processed by a processor. That is, $\sigma(j,k,t) = 1$ if job j is processed by processor k at time point t and 0 otherwise. A solution is *feasible* if and only if it satisfies the following conditions:

(i) $\sum_t \sigma(j,k,t) = \pi_{j,k}$ for all jobs j and processors k,
(ii) $\sum_j \sigma(j,k,t) \in \{0,1\}$ for all processors k and time points t,
(iii) $\max\{t \mid \sigma(j,k,t) = 1\} < \min\{t' \mid \sigma(j,\ell,t') = 1\}$ for all jobs j and pairs of processors k, ℓ such that k precedes ℓ in the order of processors, and $\pi_{j,k} > 0$ and $\pi_{j,\ell} > 0$ hold.

Below we describe a transformation f from PFSP to RCPSVP. Let I be an instance of PFSP, in the corresponding instance $f(I)$ of RCPSVP the set of resources R coincides with the set of processors. Each activity represents the processing of a job by a processor. The directed graph D specifies the order in which activities representing the processing of the same job by different processors must visit these processors (resources). If activity i represents the processing of job j on processor k, then the maximum intensity of activity i is $a^i = 1/\pi_{j,k}$, and it requires $q^i_k = \pi_{j,k}$ units of resource (processor) k and no other resources. All activities can be started in time period 1 and they must all be completed at latest in time period C. The internal capacity of each resource is 1 in each time period and its external capacity is always 0. We have the following:

Lemma 1. *An instance I of PFSP admits a feasible solution if and only if the corresponding instance $f(I)$ of RCPSVP admits a feasible solution.*

For a proof see Appendix I. Before proving strong NP-completeness of RCPSVP notice that the maximum numbers $\text{Max}[I]$ and $\text{Max}'[f(I)]$ in an instance I of PFSP and in the corresponding instance $f(I)$ of RCPSVP, respectively, are the same, i.e., both have the value $\max\{\max \pi_{j,k}, C\}$.

Corollary 1. *RCPSVP is NP-complete in the strong sense.*

Proof. Membership in NP is straightforward. To show strong NP-completeness it suffices to verify that f is a *pseudo-polynomial transformation* (Garey and Johnson [8], p. 101) from PFSP to RCPSVP. Namely, an instance I of PFSP admits a feasible solution if and only if the corresponding instance $f(I)$ of RCPSVP has a feasible solution, by Lemma 1. Moreover, f can be computed in time polynomial in the length of I and $\text{Max}[I]$. The length of $f(I)$ is not smaller than the length of I and $\text{Max}'[f(I)]$ is bounded by a two-variable polynomial in the length of I and $\text{Max}[I]$, since they are the same. □

As a consequence, RCPSVP cannot be solved in pseudo-polynomial time, unless P = NP. Hence, we will propose a branch-and-cut algorithm using strong valid inequalities which is the topic of the next section.

5. Polyhedral results for RCPSVP

If we omit the resource capacity constraints (1e) from the weak (or from the strong) formulation of RCPSVP, then the feasible solutions of the remaining system are all intensity assignments to activities respecting the release times, deadlines, maximum intensities and precedence constraints. We may consider this system as a collection of subsystems each describing the feasible intensity assignments to a *pair of activities* connected by a precedence constraint. In this and the next section we will obtain a complete description of the polytope associated with such a subsystem.

5.1. Feasible intensity assignments to a pair of activities

Let i, j be a pair of activities with $(i, j) \in A$. If $d^i < r^j$, then this precedence constraint is meaningless, so we may assume that $r^j \le d^i$. Recall also that $r^i + p^i \le r^j$ and $d^i \le d^j - p^j$. We define the polytope K^{ij} as the convex hull of all feasible intensity assignments to i and j such that i completes before j starts. That is, K^{ij} is the convex hull of all points $(x^i, x^j, z^i) \in \mathbb{R}^{s^i} \times \mathbb{R}^{s^j} \times \{0,1\}^{s^i - p^i}$, where $s^i = d^i - r^i + 1$ and $s^j = d^j - r^j + 1$, satisfying the following constraints:

$$\sum_{t=r^i}^{d^i} x_t^i = 1, \tag{2a}$$

$$0 \le x_t^i \le a^i, \quad t \in \{r^i, \ldots, r^i + p^i - 1\} \tag{2b}$$

$$0 \le x_t^i \le a^i \cdot z_t^i, \quad t \in \{r^i + p^i, \ldots, d^i\} \tag{2c}$$

$$z_t^i \ge z_{t+1}^i, \quad t \in \{r^i + p^i, \ldots, r^j - 1\} \tag{2d}$$

$$z_t^i \ge z_{t+1}^i, \quad t \in \{r^j, \ldots, d^i - 1\} \tag{2e}$$

$$\sum_{t=r^j}^{d^j} x_t^j = 1, \tag{2f}$$

$$0 \leq x_t^j \leq a^j \cdot (1 - z_t^i), \quad t \in \{r^j, \dots, d^i\} \tag{2g}$$

$$0 \leq x_t^j \leq a^j, \quad t \in \{d^i + 1, \dots, d^j\} \tag{2h}$$

In order to find a linear representation of K^{ij} consider the polytopes K^{i*} and K^{*j} derived from K^{ij} as follows:

$$K^{i*} = \text{conv}\left\{(x^i, z^i) \in \mathbb{R}^{s^i} \times \{0,1\}^{s^i - p^i} \mid (x^i, z^i) \text{ satisfies } (2a) - (2e)\right\},$$

$$K^{*j} = \text{conv}\left\{(x^j, \tilde{z}^i) \in \mathbb{R}^{s^j} \times \{0,1\}^{d^i - r^j + 1} \mid (x^j, \tilde{z}^i) \text{ satisfies } (2e) - (2h)\right\}.$$

The main result of this section is the following:

Lemma 2. *Let (x^i, x^j, z^i) be any point in $\mathbb{R}^{s^i} \times \mathbb{R}^{s^j} \times \mathbb{R}^{s^i - p^i}$. Then $(x^i, x^j, z^i) \in K^{ij}$ if and only if $(x^i, z^i) \in K^{i*}$ and $(x^j, \tilde{z}^i) \in K^{*j}$, where $\tilde{z}_t^i = z_t^i$ for all $t \in \{r^j, \dots, d^i\}$.*

Before the proof we derive some common properties of K^{i*} and K^{*j}. In fact, K^{i*} and K^{*j} are instances of the polytope K defined next. Let $1 \leq m < n$ be integer numbers, and $0 < a \leq 1$ a real number such that $(n - m)a \geq 1$. Then K is the convex hull of all points $(x, z) \in \mathbb{R}^n \times \{0,1\}^m$ satisfying the following linear constraints[1]:

$$\sum_{t=1}^{n} x_t = 1 \tag{3a}$$

$$x_t \leq a \cdot (1 - z_t), \quad t \in \{1, \dots, m\} \tag{3b}$$

$$x_t \leq a, \quad t \in \{m + 1, \dots, n\} \tag{3c}$$

$$z_t \geq z_{t+1}, \quad t \in \{1, \dots, m - 1\} \tag{3d}$$

$$x_t \geq 0, \quad t \in \{1 \dots, n\}. \tag{3e}$$

We obtain a polytope equivalent to K^{i*} by the substitutions $m = s^i - p^i$, $n = s^i$, $a = a^i$, $z_t = 1 - z^i_{d^i - t+1}$, $t \in \{1, \dots, m\}$, and $x_t = 1 - x^i_{d^i - t+1}$, $t \in \{1, \dots, n\}$. We get a polytope equivalent to K^{*j} by setting $m = d^i - r^j + 1$, $n = s^j$, $a = a^j$, $z_t = \tilde{z}^i_{t+r^j-1}$, $t \in \{1, \dots, m\}$, and $x_t = x^j_{t+r^j-1}$, $t \in \{1, \dots, n\}$.

A necessary and sufficient condition for a vector $(x, z) \in \mathbb{R}^n \times \mathbb{R}^m$ to be in K is provided next. First of all, if $(\hat{x}, \hat{z})$ is a vertex of K, then $\hat{z}$ is one of the following vectors $z^\ell \in \{0,1\}^m$:

$$z_t^\ell = \begin{cases} 1 \text{ if } t \in \{1, \dots, \ell - 1\}, \\ 0 \text{ if } t \in \{\ell, \dots, m\}, \end{cases} \qquad \ell \in \{1, \dots, m + 1\}.$$

Moreover, if $\hat{z} = z^\ell$ for some vertex $(\hat{x}, \hat{z})$ and some ℓ, then $\hat{x}_t = 0$ for all $t \in \{1, \dots, \ell - 1\}$, $0 \leq \hat{x}_t \leq a$ for all $t \in \{\ell, \dots, n\}$ and $\sum_{t=1}^n \hat{x}_t = 1$. After these preparations, we can prove the following:

[1] We might have chosen another definition, e.g. have $a \cdot z_t$ on the r.h.s. of (3b) and $\leq$ instead of $\geq$ in (3d), but then all subsequent formulas would be more complicated.

Lemma 3. *Let (x, z) be any point in $\mathbb{R}^n \times \mathbb{R}^m$. Then $(x, z) \in K$ if and only if the numbers $\lambda_1 = 1 - z_1$, $\lambda_\ell = z_{\ell-1} - z_\ell$ ($\ell \in \{2, \dots, m\}$) and $\lambda_{m+1} = z_m$ are all non-negative and there exist vectors $x^\ell \in \mathbb{R}^n$, $\ell \in \{1, \dots, m+1\}$, such that $\sum_{\ell=1}^{m+1} \lambda_\ell x^\ell = x$ and every (x^ℓ, z^ℓ) belongs to K.*

Proof. (Necessity) If (x, z) is a point in K and the points $(\hat{x}^h, \hat{z}^h)$, $h \in \Theta$, constitute the set of vertices of K, then there exist reals $\omega_h \geq 0$ such that $(x, z) = \sum_h \omega_h(\hat{x}^h, \hat{z}^h)$, $\sum_h \omega_h = 1$.

Let $\lambda_\ell = \sum_{h:\hat{z}^h = z^\ell} \omega_h$, $\ell \in \{1, \dots, m+1\}$. Then $\lambda_\ell \geq 0$ and $\sum_\ell \lambda_\ell = 1$. If $\lambda_\ell > 0$, let $x^\ell = \sum_{h:\hat{z}^h = z^\ell}(\omega_h/\lambda_\ell)\hat{x}^h$. Otherwise, if $\lambda_\ell = 0$, choose an arbitrary x^ℓ such that $(x^\ell, z^\ell) \in K$. Then every (x^ℓ, z^ℓ) belongs to K and $\sum_\ell \lambda_\ell(x^\ell, z^\ell) = (x, z)$. Using this and the fact that $z_m^\ell = 1$ if and only if $\ell = m+1$, $\lambda_{m+1} = z_m$ follows. An inductive argument proves that the rest of the λ_ℓ must also equal the specified values.

(Sufficiency) Observe that $\sum_{\ell=1}^{m+1} \lambda_\ell = 1$ and also $\sum_{\ell=1}^{m+1} \lambda_\ell z^\ell = z$. Consequently, if there exist vectors x^ℓ satisfying the conditions of the lemma, then (x, z) is a convex combination of the points (x^ℓ, z^ℓ). Since the vectors (x^ℓ, z^ℓ) all belong to K, $(x, z) \in K$. □

Proof of Lemma 2. By the definitions, if $(x^i, x^j, z^i) \in K^{ij}$, then $(x^i, z^i) \in K^{i*}$ and $(x^j, \tilde{z}^i) \in K^{*j}$, where $\tilde{z}_t^i = z_t^i$ for all $t \in \{r^j, \dots, d^i\}$.

Conversely, suppose $(x^i, z^i) \in K^{i*}$ and $(x^j, \tilde{z}^i) \in K^{*j}$, where $\tilde{z}_t^i = z_t^i$ for all $t \in \{r^j, \dots, d^i\}$. In order to apply the previous lemma to K^{i*} and K^{*j}, define the vectors $z^{i,\ell} \in \{0, 1\}^{s^i - p^i}$ and $z^{j,\ell} \in \{0, 1\}^{d^i - r^j + 1}$ as follows.

$$z_t^{i,\ell} = \begin{cases} 1 \text{ if } t \in \{r^i + p^i, \dots, \ell - 1\}, \\ 0 \text{ if } t \in \{\ell, \dots, d^i\} \end{cases} \qquad \ell \in \{r^i + p^i, \dots, d^i + 1\}.$$

For each $\ell \in \{r^j, \dots, d^i + 1\}$, let $z_t^{j,\ell} = z_t^{i,\ell}$ for all $t \in \{r^j, \dots, d^i\}$.

Applying Lemma 3 to K^{i*} yields vectors $x^{i,\ell}$ and coefficients λ_ℓ, $\ell \in \{r^i + p^i, \dots, d^i + 1\}$ such that $\sum_{\ell=r^i+p^i}^{d^i+1} \lambda_\ell(x^{i,\ell}, z^{i,\ell}) = (x^i, z^i)$, $(x^{i,\ell}, z^{i,\ell}) \in K^{i*}$ for each ℓ, $\lambda_{r^i+p^i} = 1 - z_{r^i+p^i}$, $\lambda_\ell = z_{\ell-1}^i - z_\ell^i$, $\ell \in \{r^i + p^i + 1, \dots, d^i\}$, and $\lambda_{d^i+1} = z_{d^i}^i$.

For K^{*j} we get vectors $x^{j,\ell}$ and coefficients α_ℓ, $\ell \in \{r^j, \dots, d^i + 1\}$ such that $\sum_{\ell=r^j}^{d^i+1} \alpha_\ell(x^{j,\ell}, z^{j,\ell}) = (x^j, \tilde{z}^i)$, $(x^{j,\ell}, z^{j,\ell}) \in K^{*j}$ for each ℓ, $\alpha_{r^j} = 1 - \tilde{z}_{r^j}^i$, $\alpha_\ell = \tilde{z}_{\ell-1}^i - \tilde{z}_\ell^i$, $\ell \in \{r^j + 1, \dots, d^i\}$, and $\alpha_{d^i+1} = \tilde{z}_{d^i}^i$.

Since $z_t^i = \tilde{z}_t^i$, $t \in \{r^j, \dots, d^i\}$ by assumption, it follows that $\alpha_\ell = \lambda_\ell$, $\ell \in \{r^j + 1, \dots, d^i + 1\}$, and $\alpha_{r^j} = 1 - \sum_{\ell=r^j+1}^{d^i+1} \lambda_\ell = \sum_{\ell=r^i+p^i}^{r^j} \lambda_\ell$.

Letting $x^{j,\ell} = x^{j,r^j}$ for each $\ell \in \{r^i + p^i, \dots, r^j - 1\}$, we have $(x^i, x^j, z^i) = \sum_{\ell=r^i+p^i}^{d^i+1} \lambda_\ell(x^{i,\ell}, x^{j,\ell}, z^{i,\ell})$. Since each $(x^{i,\ell}, x^{j,\ell}, z^{i,\ell})$ belongs to K^{ij} by the definition of the polytopes K^{i*} and K^{*j}, $(x^i, x^j, z^i) \in K^{ij}$ as well. □

Corollary 2. *If $K^{i*} = \{(x^i, z^i) \mid A^{i*}x^i + B^{i*}z^i \leq b^{i*}\}$ and $K^{*j} = \{(x^j, \tilde{z}^i) \mid A^{*j}x^j + B^{*j}\tilde{z}^i \leq b^{*j}\}$, then $K^{ij} = \{(x^i, x^j, z^i) \mid A^{i*}x^i + B^{i*}z^i \leq b^{i*}, A^{*j}x^j + [0, B^{*j}]z^i \leq b^{*j}\}$, where 0 is a null matrix with $r^j - r^i - p^i$ columns and appropriate number of rows.*

We will provide a minimal linear representation of K in Section 6, which can trivially be transformed to one for K^{i*} and for K^{*j}, respectively. We will prove that putting together these two representations yields a minimal linear representation of K^{ij}.

We finish this section by relating K to the polytope $\mathcal{P}_=$ defined in [21] to model fixed charge network or variable upper-bound flow problems. Namely, $\mathcal{P}_= = \text{conv}\{(x,z) \in \mathbb{R}^n \times \{0,1\}^n \mid \sum_{t=1}^n x_t = g, 0 \le x_t \le a_t z_t, t = 1,\dots,n\}$, where g and the a_ts are constants. The system defining the polytope models a fragment of a network consisting of a node u and arcs entering u. The total flow through the arcs has to meet a specified demand g. The upper bounds on the arcs are variable as they are determined by the binary variables z_t. Now, in the network modeled by K there are variable upper bounds on some but not on all of the arcs. Namely, the upper bound on arc flow x_t is constant a for all $t \in \{m+1,\dots,n\}$, while it is variable on all arc flows x_t with $t \in \{1,\dots,m\}$. Despite of the apparent similarities to the variable upper bound flow model, our model is very special for the upper bounds on arcs are not independent, due to constraints $z_t \ge z_{t+1}$. Namely, if z_{t_0} is set to 1, then for all $t \in \{1,\dots,t_0\}$ the upper bound on x_t is fixed at a and if $z_{t_0} = 0$ then the upper bound is 0 on all arc flows x_t with $t \in \{t_0,\dots,m\}$. We will exploit this property when deriving facets of K.

5.2. *Valid inequalities and a separation algorithm for K*

Clearly, the equation (3a) and the inequalities (3b)–(3e) are all valid for K. Moreover, the inequality

$$z_m \ge 0 \tag{3f}$$

is also valid for K. We derive a new class of valid inequalities below.

Denote $p = \lceil 1/a \rceil$ and $a_{\text{rem}} = 1 - (p-1)a$. Since $(n-m)a \ge 1$ by definition, $m + p \le n$. The short proof of the following statement has been kindly suggested by a referee.

Lemma 4. *Let $\emptyset \ne S_1 \subseteq \{1,\dots,m\}$ and $S_2 \subseteq \{m+1,\dots,n\}$ be such that $|S_1|+|S_2| = p$ and let t_1 be the smallest element of S_1. The (S_1, S_2) inequality*

$$a_{\text{rem}} z_{t_1} + a \sum_{t \in S_1 - \{t_1\}} z_t \le \sum_{t \in \{t_1,\dots,n\} - (S_1 \cup S_2)} x_t \tag{3g}$$

is valid for K.

Proof. As K is a convex polytope, it suffices to show that any vertex of K satisfies (3g). Since the inequalities are valid for any vertex $(\hat{x}, \hat{z})$ with $\hat{z}_{t_1} = 0$, assume $\hat{z}_{t_1} = 1$. Therefore, $\hat{x}_{t_1} = 0$ by (3b), and the total processing in $\{t_1,\dots,n\} - (S_1 \cup S_2)$, that is, the right hand side of (3g), is at least 1 minus the maximum processing in $S_1 \cup S_2 - \{t_1\}$. The latter is $a|S_2| + a\sum_{t \in S_1 - \{t_1\}}(1 - \hat{z}_t)$. Plugging all this together, the statement follows. □

The usefulness of the (S_1, S_2) inequalities is illustrated by the following:

Example 2. Suppose $m = 4, n = 7$ and $a = 2/5$. Then $p = 3$ and $a_{\text{rem}} = 1/5$. The vector (x, z), where $x = (1/10, 0, 0, 3/10, 2/5, 1/5, 0) \in \mathbb{R}^7$ and $z = (3/4, 1/2, 1/2, 1/4) \in \mathbb{R}^4$, satisfies (3a)–(3f), but violates (3g) for $S_1 = \{1, 4\}$ and $S_2 = \{5\}$. Namely, we have

$$\begin{aligned} a_{\text{rem}} z_1 + a z_4 &= \frac{1}{5} \cdot \frac{3}{4} + \frac{2}{5} \cdot \frac{1}{4} = \frac{1}{4} > \frac{1}{5} = 0 + 0 + \frac{1}{5} + 0 \\ &= x_2 + x_3 + x_6 + x_7 = \sum_{t \in \{1,\dots,7\} - (S_1 \cup S_2)} x_t. \end{aligned}$$

□

We close this section by a separation algorithm for the (S_1, S_2) inequalities. For each $t_1 \in \{1, \dots, m\}$, the procedure tries to find a violated (S_1, S_2) inequality with t_1 being the smallest element of S_1. Namely, rewrite (3g) as follows:

$$\sum_{t \in S_1 - \{t_1\}} (a \cdot z_t + x_t) + \sum_{t \in S_2} x_t \le \left(\sum_{t \in \{t_1+1,\dots,n\}} x_t \right) - a_{\text{rem}} z_{t_1}.$$

Now consider the following optimization problem:

$$\max \sum_{t \in S_1 - \{t_1\}} (a \cdot z_t + x_t) + \sum_{t \in S_2} x_t, \tag{4}$$

where the max is over all set pairs (S_1, S_2) such that $t_1 \in S_1 \subseteq \{t_1, \dots, m\}$, $S_2 \subset \{m + 1, \dots, n\}$ and $|S_1| + |S_2| = p$. The maximum is greater than the constant $\sum_{t \in \{t_1+1,\dots,n\}} x_t - a_{\text{rem}} z_{t_1}$ if and only if at least one (S_1, S_2) inequality with t_1 being the smallest element of S_1 is violated by (x, z).

In order to solve problem (4), define a set of items $I(t_1) = \{t_1 + 1, \dots, n\}$ with item weights $w(t) = a \cdot z_t + x_t$ if $t_1 + 1 \le t \le m$, and x_t if $m + 1 \le t \le n$. Then the $p - 1$ largest-weight items constitute an optimal solution to (4). In fact, this problem can be seen as finding a maximum weight basis in the uniform matroid over $I(t_1)$ in which every $p - 1$ items is a basis. Repeating this procedure for each $t_1 \in \{1, \dots, m\}$ we can find a violated (S_1, S_2) inequality or conclude that none exists.

For efficient implementation notice that the item weights do not depend on t_1 and thus it is enough to sort the elements of the set $I(1)$ in decreasing order of their weights. Then, after increasing t_1 by 1, eliminate from the ordered list those t for which $t \le t_1$ holds. The time complexity of the entire separation procedure is $O(n \log n)$.

6. Minimal linear representations for K and K^{ij}

In this section we will prove that the inequalities (3a)–(3g) constitute a linear representation of K. Moreover, we will establish conditions under which these inequalities represent facets of K. Finally, we will extend these results to K^{ij}.

6.1. A linear representation of K

Let P be the polytope consisting of all points $(x, z) \in \mathbb{R}^n \times \mathbb{R}^m$ satisfying the system (3a)–(3g). Since the equation (3a) and the inequalities (3b)–(3g) are all valid for K, $K \subseteq P$. Our main goal is to prove the following:

Theorem 1. $P \subseteq K$.

Proof. Assuming the contrary, fix some $(x, z) \in P \setminus K$. Since $(x, z) \notin K$, it is not a convex combination of points in K. We will show that then (3g) is violated for a pair of sets (S_1, S_2) which contradicts the assumption $(x, z) \in P$.

To this end, define the capacitated network $G(x, z) = (V, E)$ with node set V consisting of a source s, a sink q, a node v_ℓ for each $\ell \in \{1, \ldots, m+1\}$ and a node w_t for each $t \in \{1, \ldots, n\}$. The source s is connected to every v_ℓ by one arc (s, v_ℓ) with capacity $c(s, v_\ell) = \lambda_\ell$, where the λ_ℓ are defined in Lemma 3. Moreover, there is one arc from each w_t to sink q with capacity $c(w_t, q) = x_t$. Finally, for each $\ell \in \{1, \ldots, m+1\}$ and $t \in \{\ell, \ldots, n\}$ there is an arc (v_ℓ, w_t) with capacity $c(v_\ell, w_t) = a \cdot \lambda_\ell$. This construction is illustrated in Fig. 2. Since $(x, z) \in P$, all arc capacities in $G(x, z)$ are non-negative.

Let $c(\delta(S)) = \sum_{(u,v) \in \delta(S)} c(u, v)$ denote the capacity of an $s - q$ cut $\delta(S) = \{(u, v) \in E \mid u \in S, v \in \overline{S}\}$, where $s \in S \subseteq V \setminus \{q\}$ and $\overline{S} = V \setminus S$. The minimum capacity of an $s - q$ cut in $G(x, z)$ is at most 1, as $c(\delta(\{s\})) = \sum_\ell \lambda_\ell = 1$.

Claim 1. The minimum capacity of an $s - q$ cut in $G(x, z)$ is strictly smaller than 1.

Proof. Assuming the contrary, it follows that there exists a compatible flow f in $G(x, z)$ of value 1, due to the MAX-FLOW MIN-CUT Theorem of Ford and Fulkerson [7]. For each $\ell \in \{1, \ldots, m+1\}$ define a vector $x^\ell \in \mathbb{R}^n$ as follows. If $\lambda_\ell > 0$, then let $x^\ell_t = f(v_\ell, w_t)/\lambda_\ell$ for $t \in \{\ell, \ldots, n\}$, and $x^\ell_t = 0$ for $t \in \{1, \ldots, \ell - 1\}$. On the other hand, when $\lambda_\ell = 0$, choose any x^ℓ such that $(x^\ell, z^\ell) \in K$. Hence, every (x^ℓ, z^ℓ) belongs to K and $\sum_\ell \lambda_\ell x^\ell_t \leq x_t$ for each t. But $\sum_t \sum_\ell \lambda_\ell x^\ell_t = 1$, since f is feasible, and $\sum_t x_t = 1$ as $(x, z) \in P$, whence, $\sum_\ell \lambda_\ell x^\ell = x$. Then Lemma 3 ensures that $(x, z) \in K$, a contradiction. □

Let $N(v_\ell) = \{w_t \in V \mid (v_\ell, w_t) \in E\}$ denote the set of nodes reachable from v_ℓ by one arc. If S is a subset of nodes, then let $N_S(v_\ell) = N(v_\ell) \cap S$.

Property 1. *Let S be an $s - q$ cut of $G(x, z)$. If $\ell < \kappa$ then $N_{\overline{S}}(v_\kappa) \subseteq N_{\overline{S}}(v_\ell)$.*

Claim 2. There exists a minimum capacity $s - q$ cut $\delta(S)$ in $G(x, z)$ with the following structure:

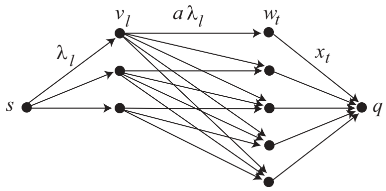

Fig. 2. The construction of $G(x, z)$.

(i) there exists $t_1 \in \{1, \dots, m\}$ such that $v_\ell \in \overline{S}$ for all $\ell \in \{1, \dots, t_1\}$ and $v_\ell \in S$ for all $\ell \in \{t_1 + 1, \dots, m + 1\}$,
(ii) $w_t \in \overline{S}$ for all $t \in \{1, \dots, t_1\}$,
(iii) for every $v_\ell \in S$, $|N_{\overline{S}}(v_\ell)| \le p - 1$,
(iv) $|N_{\overline{S}}(v_{t_1+1})| = p - 1$.

Proof. Let $\delta(S)$ be a minimum capacity $s - q$ cut of $G(x, z)$. Clearly, $c(\delta(S)) < 1$ holds by assumption. First we claim that there exists $\ell \in \{1, \dots, m + 1\}$ such that $v_\ell \in S$. Indeed, otherwise the arcs (s, v_ℓ) all leave S, whence $c(\delta(S)) \ge \sum_{\ell=1}^{m+1} c(s, v_\ell) = \sum_{\ell=1}^{m+1} \lambda_\ell = 1$, contradicting $c(\delta(S)) < 1$.

Proof of part (iii). Suppose there exists $v_\ell \in S$ with $|N_{\overline{S}}(v_\ell)| \ge p$. If $\lambda_\ell = 0$, we can replace S by $S - \{v_\ell\}$ without changing the capacity of the cut. So assume $\lambda_\ell > 0$. From flow theory (see e.g., Ahuja et al. [1]) we know that any maximum value $s - q$ flow f saturates all arcs leaving S, since $\delta(S)$ is of minimum capacity. In particular, $f(v_\ell, w_t) = c(v_\ell, w_t) = a \cdot \lambda_\ell$ holds for all arcs $(v_\ell, w_t) \in E$ with $w_t \in N_{\overline{S}}(v_\ell)$. Using conservation of flow we can derive that

$$\lambda_\ell \ge f(s, v_\ell) = \sum_{(v_\ell, w_t) \in E} f(v_\ell, w_t) \ge |N_{\overline{S}}(v_\ell)| \cdot a \cdot \lambda_\ell.$$

Since $p \cdot a \ge 1$ by the definition of p and $|N_{\overline{S}}(v_\ell)| \ge p$ by assumption, it must be the case that $|N_{\overline{S}}(v_\ell)| = p$ and $p \cdot a = 1$, otherwise there is a contradiction. Hence, we may replace S by $S - \{v_\ell\}$ without changing the capacity of the cut.

Proof of part (i). Since S contains at least one of the nodes v_ℓ, there exists a unique $t_1 \in \{0, \dots, m\}$ such that $v_\ell \notin S$ for all $\ell \in \{1, \dots, t_1\}$ and $v_{t_1+1} \in S$. Suppose there exists $\kappa > t_1 + 1$ such that $v_\kappa \notin S$. If $\lambda_\kappa = 0$, we can add v_κ to S yielding a cut with the same capacity. Assume $\lambda_\kappa > 0$. Since $N_{\overline{S}}(v_\kappa) \subseteq N_{\overline{S}}(v_{t_1+1})$ by Property 1 and $|N_{\overline{S}}(v_{t_1+1})| \le p - 1$ by part (iii), we also have $|N_{\overline{S}}(v_\kappa)| \le p - 1$. Since the capacity of the cut determined by $S \cup \{v_\kappa\}$ is

$$\begin{aligned} c(\delta(S \cup \{v_\kappa\})) &= c(\delta(S)) - c(s, v_\kappa) + \sum_{w_t \in N_{\overline{S}}(v_\kappa)} c(v_\kappa, w_t) \\ &\le c(\delta(S)) - \lambda_\kappa + (p - 1) \cdot a \cdot \lambda_\kappa < c(\delta(S)), \end{aligned} \tag{5}$$

$\delta(S)$ is not of minimum capacity, which is a contradiction. It remains to show that $t_1 \ge 1$. Suppose $t_1 = 0$. To simplify notation, we introduce dummy z_t symbols with value 0, for $t \in \{m + 1, \dots, n\}$. Then S contains the source node s, all nodes v_ℓ and some of the nodes w_t. Denoting by U the indices t such that $w_t \notin S$, we determine the capacity of $\delta(S)$ below:

$$\begin{aligned} c(\delta(S)) &= \sum_{t \in U} \sum_{\ell=1}^{\min(m+1,t)} a \cdot \lambda_\ell + \sum_{t \in \{1,\dots,n\} - U} x_t \\ &= \sum_{t \in U} (1 - z_t) a + \sum_{t \in \{1,\dots,n\} - U} x_t \ge \sum_{t \in \{1,\dots,n\}} x_t, \end{aligned}$$

where we exploited that $\sum_{\ell=1}^{\min(m+1,t)} \lambda_\ell = 1 - z_t$ and that $(1 - z_t)a \geq x_t$, for (x, z) satisfies (3b) by assumption. But $\sum_{t=1}^{n} x_t = 1$, hence $c(\delta(S)) \geq 1$, a contradiction.

Proof of part (ii). Suppose there exists $w_t \in S$ with $t \in \{1, \dots , t_1\}$. Then $(w_t, q) \in E$ is an arc leaving S. Since w_t is not connected to any other node in S, the capacity of $\delta(S)$ can be decreased by x_t unless $x_t = 0$. Since $\delta(S)$ is of minimum capacity by assumption, it must be the case that $x_t = 0$. However, in this case w_t can be removed from S yielding a cut with the same capacity.

Proof of part (iv). Consider the node v_{t_1}. Since $w_{t_1} \notin S$ by part (ii), it follows that $N_{\overline{S}}(v_{t_1}) = N_{\overline{S}}(v_{t_1+1}) \cup \{w_{t_1}\}$. Consequently, if $|N_{\overline{S}}(v_{t_1+1})| < p-1$, then $|N_{\overline{S}}(v_{t_1})| < p$. If $\lambda_{t_1} = 0$, then $c(\delta(S \cup \{v_{t_1}\})) = c(\delta(S))$, whence we can replace S by $S \cup \{v_{t_1}\}$ and repeat the whole analysis. Assume $\lambda_{t_1} > 0$. Then adding v_{t_1} to S would decrease the capacity of the cut, a contradiction. □

To finish the proof of the theorem, define the sets $S_1 = \{t_1\} \cup \{t \in \{t_1+1, \dots , m\} \mid w_t \in \overline{S}\}$ and $S_2 = \{t \in \{m + 1, \dots , n\} \mid w_t \in \overline{S}\}$, where S is the node set found in Claim 2. Notice that $\emptyset \neq S_1 \subseteq \{1, \dots , m\}$, $S_2 \subseteq \{m + 1, \dots , n\}$ and $|S_1 \cup S_2| = p$, by part (iv) of Claim 2. To simplify the following presentation, define new symbols z_t with value 0 for all $t \in \{m + 1, \dots , n\}$. Since the capacity of $\delta(S)$ is strictly smaller than 1, we have

$$1 > \sum_{t=1}^{t_1} c(s, v_t) + \sum_{\ell=t_1+1}^{m} \sum_{t\in\{\ell,\dots,n\}\cap(S_1\cup S_2)} c(v_\ell, w_t) + \sum_{t\in\{t_1,\dots,n\}\setminus(S_1\cup S_2)} c(w_t, q).$$

Using the definition of arc capacities and that of the λ_ℓ, this can be rewritten as

$$\begin{aligned} &= 1 - z_{t_1} + a \sum_{t\in(S_1\cup S_2)\setminus\{t_1\}} \sum_{\ell=t_1+1}^{t} (z_{\ell-1} - z_\ell) + \sum_{t\in\{t_1,\dots,n\}\setminus(S_1\cup S_2)} x_t \\ &= 1 - z_{t_1} + a \sum_{t\in(S_1\cup S_2)\setminus\{t_1\}} (z_{t_1} - z_t) + \sum_{t\in\{t_1,\dots,n\}\setminus(S_1\cup S_2)} x_t. \end{aligned}$$

Since $z_t = 0$ for all $t \in \{m + 1, \dots , n\}$ and $|S_1 \cup S_2| = p$, this is equivalent to

$$= 1 - z_{t_1} + a(p - 1)z_{t_1} - a \sum_{t\in S_1\setminus\{t_1\}} z_t + \sum_{t\in\{t_1,\dots,n\}\setminus(S_1\cup S_2)} x_t.$$

Now, as $-z_{t_1} + a(p - 1)z_{t_1} = -a_{\text{rem}} z_{t_1}$, it follows that (3g) is violated for the sets (S_1, S_2), that is, $(x, z) \notin P$, a contradiction. □

6.2. *Facets of K*

In order to find a minimal linear representation of polytope K, it suffices to consider only the system (3a)–(3g), by Theorem 1. To show that a valid inequality $\alpha x + \beta z \leq \gamma$ from (3b)–(3g) represents a facet of K we will use the standard proof technique consisting in exhibiting a point (x, z) in K such that $\alpha x + \beta z = \gamma$ and (x, z) satisfies all other inequalities with strict inequality, see e.g., Schrijver [23].

Lemma 5. *The inequalities (3b), (3d), (3f) always represent facets of K. The inequalities (3c) represent facets iff $a < 1$. For $1 \leq t \leq n$ inequality (3e)$_t$ represents a facet iff $a > 1/(n-1)$ and if $t \geq m+1$, $p < n-m$. Finally, for each $\emptyset \neq S_1 \subseteq \{1, \dots, m\}$, $S_2 \subset \{m+1, \dots, n\}$ with $|S_1|+|S_2| = p$ the corresponding inequality in (3g) represents a facet unless $t_1 = 1$ and $a = 1/p$.*

Proof. Firstly, we show that there exists a point in K satisfying all inequalities (3b)–(3g) with strict inequality. One may verify that if $\varepsilon > 0$ is a sufficiently small positive number, the vector (x, z) given by

$$x_t = 1/n,\ t \in \{1, \dots, n\} \text{ and } z_t = \varepsilon^t,\ t \in \{1, \dots, m\} \tag{6}$$

is in K satisfying every inequality in (3b)–(3g) with strict inequality.

The construction of an appropriate point in K for (3b), (3d) and (3f) being straightforward, we turn directly to the inequalities (3c). If $a = 1$ then $x_t = 1$ implies $x_{t'} = 0$ for all $t' \in \{1, \dots, n\} - \{t\}$, proving the necessity of the condition. Conversely, if $a < 1$, the vector (x, z) with $x_t = a$, $x_{t'} = (1-a)/(n-1)$, $t' \in \{1, \dots, n\} - \{t\}$, and z as in (6) satisfies $x_t = a$ with equality, while all other inequalities are strict if ε is sufficiently small.

Concerning (3e), if $a = 1/(n-1)$ then $x_t = 0$ implies $x_{t'} = a$ for all $t' \in \{1, \dots, n\} - \{t\}$, proving the necessity of the condition. Moreover, if $t \geq m+1$ and $m + p = n$, then $x_t \geq 0$ is implied by the (S_1, S_2) inequality with $t_1 = m$, $S_1 = \{m\}$, $S_2 = \{m+1, \dots, n\} - \{t\}$. Now suppose $a > 1/(n-1)$. The vector (x, z) with $x_t = 0$, $x_{t'} = 1/(n-1)$, $t' \in \{1, \dots, n\} - \{t\}$, and z as in (6) satisfies $x_t = 0$ with equality while all other inequalities are strict if ε is sufficiently small and when $t \geq m+1, m+p < n$.

Finally, consider an inequality in (3g). If $t_1 = 1 \in S_1$ and $a = 1/p$, then this inequality is implied by the inequalities $x_t \leq a(1 - z_t)$, $t \in S_1$, and $x_t \leq a$, $t \in S_2$. Namely, since $a = 1/p$, $a_{\text{rem}} = a$ follows. Therefore, we have

$$\begin{aligned}\sum_{t \in S_1} a z_t &\leq \sum_{t \in S_1} a z_t + \sum_{t \in S_2} (a - x_t) \leq \sum_{t \in S_1} (a - x_t) + \sum_{t \in S_2} (a - x_t) \\ &= ap - \sum_{t \in S_1 \cup S_2} x_t = 1 - \sum_{t \in S_1 \cup S_2} x_t = \sum_{\{1, \dots, n\} - (S_1 \cup S_2)} x_t.\end{aligned}$$

On the other hand, when $t_1 \geq 2$ or $ap > 1$, we will define a point (x, z) in K such that

$$a_{\text{rem}} z_{t_1} + a \sum_{t \in S_1 - \{t_1\}} z_t = \sum_{t \in \{t_1, \dots, n\} - (S_1 \cup S_2)} x_t \tag{7}$$

while all other inequalities will be strict. Let ε and ϕ be small positive numbers to be chosen later. Denote $U = \{t_1, \dots, n\} - (S_1 \cup S_2)$ and $k = |U| = n - t_1 + 1 - p$. Notice that $1 \notin U$ even if $t_1 = 1$. Therefore, $k \leq n - 1$. Finally, $k \geq 1$, since $t_1 \leq m$, $|S_1 \cup S_2| = p$ and $m + p \leq n$. Observe that $a(n-k) = a(t_1 - 1 + p) > 1$, since $t_1 \geq 2$ or $ap > 1$ by assumption.

Define z as in (6), while let x be given by

$$x_t = \begin{cases} \phi, & t \in U \\ (1 - k\phi)/(n-k), & t \in \{1, \dots, n\} - U. \end{cases}$$

Now, by substituting the definitions for x and z in (7) we obtain the following relation between ε and ϕ:

$$a_{\text{rem}}\varepsilon^{t_1} + a \sum_{t \in S_1 - \{t_1\}} \varepsilon^t = k\phi. \tag{8}$$

Observe that the left hand side is polynomial in terms of ε and that all coefficients are positive. Consequently, the left hand side is a monotone increasing and continuous function of ε. Hence, for any $\phi > 0$ there exists a unique $\varepsilon > 0$ satisfying eq. (8) and vice versa.

Clearly, (x, z) satisfies equation (3a) and the inequalities (3d)–(3f) with strict inequality. Concerning (3c), for each $t \in U$, $x_t < a$ if ϕ is small. If $t \in \{1, \ldots, n\} - U$, then $x_t = (1 - k\phi)/(n - k) < 1/(n - k) = 1/(t_1 - 1 + p) \leq 1/p \leq a$. Checking (3b) is a bit tricky. For each $t \in U$ strict inequality holds if ε and ϕ are sufficiently small. It remains to show that for each $t \notin U$, $x_t = (1 - k\phi)/(n - k) < a(1 - \varepsilon^t) = a(1 - z_t)$. This can be rewritten as $a(n - k)\varepsilon^t < a(n - k) - 1 + k\phi$. Since $a(n - k) > 1$, any $\varepsilon < 1 - 1/(a(n - k))$ will do for arbitrary ϕ.

Now consider (3g) for any $(S_1', S_2') \neq (S_1, S_2)$. Let $U' = \{t_1', \ldots, n\} - (S_1' \cup S_2')$. Since $|S_1'| + |S_2'| = p = |S_1| + |S_2|$, at least one of the sets $U - U'$ and $U' - U$ is not empty. First suppose $U' - U$ is not empty. The left hand side of (3g) on (S_1', S_2') is at most

$$a_{\text{rem}}\varepsilon + a \sum_{t=2}^{\min(p,m)} \varepsilon^t. \tag{9}$$

We claim that the right hand side is greater than (9) if ε is sufficiently small. Namely, choose $t' \in U' - U$ arbitrarily. As $t' \notin U$, $x_{t'} = (1 - k\phi)/(n - k)$. Since $(1 - k\phi)/(n - k) \geq 1/n$ when $0 \leq \phi \leq 1/n$, it follows that $x_{t'} \geq 1/n$ for ϕ sufficiently small. Hence, the right hand side of (3g) on (S_1', S_2') is at least $1/n$. So, choose $\varepsilon > 0$ independently from the particular (S_1', S_2') so that the quantity (9) is strictly smaller than $1/n$. The chosen ε determines ϕ by (8). If $\phi > 1/n$, decrease ϕ and proportionally also ε.

If $U' - U$ is empty, then $U - U'$ cannot be empty. These conditions along with $|S_1| + |S_2| = p = |S_1'| + |S_2'|$ imply that $t_1 < t_1'$. Since $\varepsilon^{t_1'}/\varepsilon^{t_1}$ can be made arbitrarily close to 0 by decreasing ε, the left hand side of (3g) on (S_1', S_2') can be made smaller than ϕ while maintaining eq. (8). Since the right hand side of (3g) is at least ϕ, we have shown that the inequality is strict.

Finally, our demonstration also shows that every facet inducing inequality in (3b)–(3g) represents a distinct facet of K. $\square$

6.3. A minimal linear representation of K^{ij}

To obtain a minimal linear representation of K^{ij} we first determine one for K^{i*} and K^{*j}, respectively, as follows. That is, define the polytope K with respect to K^{i*} as in Section 5.1. Take the minimal linear representation of K and convert it back to one for K^{i*}, let $A^{i*}x^i + B^{i*}z^i \leq b^{i*}$ denote this system. We can get a minimal linear representation $A^{*j}x^j + B^{*j}z^i \leq b^{*j}$ for K^{*j} by a similar procedure. We will show that almost every inequality in the combined system $A^{i*}x^i + B^{i*}z^i \leq b^{i*}$, $A^{*j}x^j + [0, B^{*j}]z^i \leq b^{*j}$ represents a facet of K^{ij}.

First observe that (3f) becomes $1 - z^i_{r^i+p^i} \geq 0$ in polytope K^{i*} and it gets $z^i_{d^i} \geq 0$ in polytope K^{*j}. Clearly $z^i_{d^i} \geq 0$ is implied by the facet-representing inequalities $x^i_{d^i} \geq 0$ and $x^i_{d^i} \leq a^i z^i_{d^i}$ for K^{i*}. Therefore, $z^i_{d^i} \geq 0$ is superfluous in the linear representation of K^{ij}. Now consider $z^i_{r^i+p^i} \leq 1$. If $r^i + p^i = r^j$, then this inequality is implied by the facet-representing inequalities $x^j_{d^j} \geq 0$ and $x^j_{r^j} \leq a^j(1 - z^i_{r^j})$ for K^{*j}. As a by-product, $z^i_{r^j} \leq 1$ never represents a facet of K^{ij}. We have the following:

Theorem 2. *The system $A^{i*}x^i + B^{i*}z^i \leq b^{i*}$, $A^{*j}x^j + [0, B^{*j}]z^i \leq b^{*j}$ without $z^i_{d^i} \geq 0$ and without $z^i_{r^i+p^i} \leq 1$ if $r^i + p^i = r^j$ is a minimal linear representation of K^{ij}.*

Proof. By Corollary 2, the linear system in the statement represents K^{ij}. Now consider e.g., a facet representing inequality $\alpha x^i + \beta z^i \leq \beta_0$ for K^{i*}. First suppose $\alpha x^i + \beta z^i \leq \beta_0$ is not one among (2e). There exists a point $(x^i, z^i) \in K^{i*}$ satisfying this inequality with equality while all other inequalities in the minimal linear representation of K^{i*} are strict. In particular, $z^i_t > z^i_{t+1}$ for $t \in \{r^j, \ldots, d^i - 1\}$ by assumption and w.l.o.g. $z^i_{d^i} > 0$, since this inequality does not represent a facet of K^{i*}. By the same token, we may also assume that $z^i_{r^j} < 1$. We have to show that there exists a point $(x^j, \tilde{z}^i) \in K^{*j}$, where $\tilde{z}^i_t = z^i_t$ for each $t \in \{r^j, \ldots, d^i\}$, such that all inequalities in the minimal linear representation of K^{*j} are strict. To this end, we construct a point in the polytope K which can be transformed to a point in K^{*j} with the desired properties. Namely, let $m = d^i - r^j + 1, n = s^j, \lambda_1 = 1 - z^i_{r^j}, \lambda_\ell = z^i_{\ell+r^j-2} - z^i_{\ell+r^j-1}, \ell \in \{2, \ldots, m\}$, and $\lambda_{m+1} = z^i_{d^i}$. We clearly have $\sum_{\ell=1}^{m+1} \lambda_\ell = 1$, and $\lambda_\ell > 0$ for each ℓ. Define the vectors $x^\ell \in \mathbb{R}^n$ as follows:

$$x^\ell_t = \begin{cases} 0 & \text{if } 1 \leq t \leq \ell - 1 \\ 1/(n - \ell + 1) & \text{if } \ell \leq t \leq n \end{cases} \qquad \ell \in \{1, \ldots, m+1\}.$$

Hence, $(x^\ell, z^\ell) \in K$ for each ℓ, where the z^ℓ are defined in Section 5.1. Moreover, for each inequality in the system (3b)–(3g) there exists at least one vector (x^ℓ, z^ℓ) on which that inequality is strict, as one may verify. Therefore, $(x, z) = \sum_\ell \lambda_\ell(x^\ell, z^\ell)$ is a point in K such that all inequalities on it are strict, since each $\lambda_\ell > 0$.

The cases when $\alpha x^i + \beta z^i \leq \beta_0$ is one among (2e) (in which case it represents a facet of both K^{i*} and K^{*j}) or $\alpha x^j + \beta \tilde{z}^i \leq \beta_0$ represents a facet of K^{*j} can be handled similarly. □

7. Implementation and computational evaluation

We implemented two branch-and-cut algorithms, B^+ and B^-, for solving the mixed integer-linear program (1). We assume familiarity with this technique, for an introduction see e.g. Padberg and Rinaldi [22] and Jünger et al. [16]. The LP relaxation of RCPSVP is obtained from the strong formulation by relaxing the constraint (1h) to $0 \leq z^i_t \leq 1$, $\forall\ i, t$. The algorithm B^+ uses the following classes of valid inequalities to cut off a solution (x, y, z) of the LP relaxation with fractional z:

(i) To each pair of activities $(i, j) \in A$ corresponds a polytope K^{ij}. The algorithm checks whether $(x^i, x^j, z^i) \in K^{ij}$ by finding the most violated (S_1, S_2) inequalities for K^{i*} and also for K^{*j}.

(ii) Flow cover inequalities. These inequalities are defined in [21]. Although they are not needed to describe the polytopes K^{i*} and K^{*j}, the constraints (1e) along with (1bc), (1g) and (1h) give rise to variable upper bound flow problems.

(iii) Gomory fractional cuts, see e.g., Nemhauser and Wolsey [20].

Algorithm B^- is identical to B^+ except it does not separate class (i) inequalities. Both algorithms start from scratch, i.e., without setting any initial upper bound.

We coded algorithms B^+ and B^- in C++ using the ILOG CPLEX 7.5 branch-and-cut solver [15]. Besides modeling the problem and calling the solver, we coded only the separation algorithm described in Section 5.2 for class (i) and we let the solver find violated inequalities in classes (ii) and (iii). The LPs were solved by the built-in dual-simplex method and we also used the built-in heuristic to find feasible solutions. In B^+ the separation algorithm for class (i) inequalities was called in the root and then in every fifth node during the search.

7.1. Comparison to Branch-and-Price

In this section we compare and evaluate three algorithms: the branch-and-price algorithm of Hans [11], denoted by H, and the algorithms B^+ and B^-. Hans evaluated his method on the benchmark instances of De Boer [5]. Each instance consists of one project only whose precedence graph was generated by the randomized procedure of Kolisch et al. [17]. The maximal intensity a^i of each activity i is given by $1/p^i$, where p^i was chosen randomly between 1 and 5. Moreover, every activity may require up to 5 distinct resources. In each time period the internal capacity of each resource is finite, and its external capacity is infinite with cost uniformly 1. The instances are subdivided into classes characterized by common parameter values such as the number of activities in the project n, the number of resources r and the average slack $s = \sum_{i=1}^{n}(s^i - p^i)/n$. For each combination of the parameter values $n = 10, 20, 50$, $r = 3, 10, 20$ and $s = 2, 5, 10, 15, 20$, ten random instances were generated yielding a total of 450 instances.

The computing environment of Hans was a PC with a 600 MHz Pentium 3 processor, Windows 2000 operating system, Borland Delphi 5 programming language and CPLEX 7.0. Hans stopped the algorithm after 1800 seconds. We performed the tests on a PC with a 1.6 GHz Pentium 4 processor under Windows 2000, and terminated the search after 675 seconds. The computational results of algorithm H reported here are slightly better than those in [11], since we used the latest data (fall of 2003) from E. Hans [12]. In the following $ub(A)$ and $lb(A)$ denote the best upper and lower bound, respectively, obtained by algorithm A, where A is one of H, B^+ and B^-.

Table 1 shows how often the three algorithms found provably optimal solutions. For each class of instances the upper number indicates the number of times algorithm H proved optimality, while the other two numbers show the performance of B^+ and B^-, respectively. Notice that either variant of our branch-and-cut algorithm proved optimality in considerably more cases than algorithm H and we got better results with B^+.

Table 1. Number of times algorithms H, B^+ and B^- proved optimality in each class.

	$n = 10$			$n = 20$			$n = 50$		
s	$r = 3$	10	20	3	10	20	3	10	20
2	10	10	10	10	10	10	7	5	0
	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10
5	10	10	10	7	1	0	0	0	0
	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/10	10/8
10	10	4	5	0	0	0	0	0	0
	10/10	10/10	10/10	10/10	10/10	10/10	10/8	2/1	0/0
15	1	0	2	0	0	0	0	0	0
	10/10	10/10	10/10	10/10	7/7	10/5	2/0	0/0	0/0
20	1	1	1	0	0	0	0	0	0
	10/10	10/10	10/10	9/8	4/2	3/2	0/0	0/0	0/0

Table 2. The average of $ub(B^+)/ub(H)$ and, when different, the average of $ub(B^-)/ub(H)$.

	$n = 10$			$n = 20$			$n = 50$		
s	$r = 3$	10	20	3	10	20	3	10	20
2	1	1	1	1	1	1	1	1	1
5	1	1	1	0.99	0.99	0.99	0.95	0.96	0.99
10	1	1	0.99	0.86	0.97	0.98	0.73	0.94/0.96	0.97/0.99
15	0.88	0.98	0.98	0.63	0.94	0.97	0.63/0.74	0.92/1.08	1.03/1.11
20	0.89	0.98	0.98	0.71	0.91	0.97	0.56/0.69	0.92/1.1	0.96/1.21

Table 2 summarizes the average $ub(B^+)/ub(H)$ ratio over the ten instances in each class, and also, when different, the average $ub(B^-)/ub(H)$ ratio. In almost all classes B^+ gives at least as good results as algorithm H on average, the only exception being when $n = 50$, $r = 20$ and $s = 15$, but B^- is inferior to H in four classes consisting of hard instances. Moreover, both B^+ and B^- improved on the best upper bounds in several cases. Comparing this table to Table 1 reveals that algorithm H found the optimum in more cases than it was able to prove optimality.

The performance of our branch-and-cut algorithms can also be measured by the ratio of the lower bound to the upper bound on every instance. Table 3 depicts for each class the average $lb(B^+)/ub(B^+)$ ratio and also, when different, the average $lb(B^-)/ub(B^-)$ ratio. In either case the ratio decreases when both the number of activities and the average slack tend to be high. However, the gap between the lower and upper bounds is significantly bigger when the (S_1, S_2) inequalities are not used.

Finally, some details of the computations are provided in Table 4. For technical reasons we give this data for algorithm B^+ only. There are three groups corresponding to the three problem sizes in terms of the number of activities. Averages are taken over the 150 instances constituting a group.

Table 3. The average of $lb(B^+)/ub(B^+)$ and, when different, the average of $lb(B^-)/ub(B^-)$.

	$n = 10$			$n = 20$			$n = 50$		
s	$r = 3$	10	20	3	10	20	3	10	20
2	1	1	1	1	1	1	1	1	1
5	1	1	1	1	1	1	1	1	1
10	1	1	1	1	1	1	1/0.99	0.96/0.90	0.93/0.88
15	1	1	1	1	0.99	1	0.86/0.68	0.85/0.70	0.80/0.70
20	1	1	1	0.99	0.97/0.95	0.98/0.96	0.77/0.51	0.66/0.49	0.72/0.53

Table 4. Some details of the computations of algorithm B^+.

	$n = 10$	$n = 20$	$n = 50$
Avg. time horizon T	28	36	45.08
Avg. CPU time in seconds	4.3	125	372
Avg. no. of search tree nodes	176	1697	1096
Avg. no. of flow cover cuts added	111	189	239
Avg. no. of fractional cuts added	9	12	13
Avg. no. of (S_1, S_2) cuts added	21	76	193

In summary, our algorithms proved optimality in considerably more cases than branch-and-price and we also improved on the previously known upper bounds. Finally, it is advantageous to use (S_1, S_2) inequalities especially when solving hard instances with a large number of activities and large slack.

7.2. *Results on Tavares' instances*

Tavares [24] proposed a non-linear program for the makespan minimization problem (cf. Section 3). In his book, computational results are reported for two test cases (Case A and Case B) with the following main characteristics. There is only one resource with finite constant internal capacity and with no external capacity. For each activity i, $r^i = 0$, $a^i = 1/p^i$, where p^i is the given integer minimum duration of activity i, and $q_1^i = 10p^i$. The precedences between the activities are given by an acyclic directed graph.

In Case A there is a project network with 75 activities. However, the same network gives rise to a series of problem instances by varying the capacity of the single resource. The tested values are 120, 100, 80, 75, 70, 65 giving rise to six distinct problem instances. In Case B the project network consists of 150 activities and there is a problem instance for each of the resource capacity limits 240, 200, 180, 120. For more details of the instance generation we refer to [25].

A lower bound on the minimum makespan is the length of the longest path in the project network, denoted by lb_p, which is computed after fixing the activity durations to the given minimum values. Clearly, this bound does not depend on the resource capacity limit. For all Case A instances $lb_p = 27$, whereas for all Case B instances $lb_p = 21$. A *resource based lower bound*, lb_r, can be computed by dividing the total resource requirement of the activities by the capacity of the resource, which was communicated to us by J. Coelho [4]. Clearly, $lb_{\max} = \max\{lb_p, lb_r\}$ is a valid lower bound on the makespan.

We solved the makespan minimization problem by a dichotomic search procedure described in Appendix II with run-time limits $\tau_1 = 3600$ and $\tau_2 = 600$ seconds. Table 5 summarizes our computational results on Case A and Case B, respectively, and also compare them to those of Tavares. Column rcap indicates the resource capacity, ub(Tav) is the upper bound obtained by Tavares [24] (the starred values are optimal) and *Opt* is our upper bound which was always optimal. Some details of computations are given in terms of the CPU time (in seconds), the total number of Gomory fractional cuts added (#frac cut), and the total number of (S_1, S_2) cuts added, #(S_1, S_2). Since CPLEX never generated any flow cover cuts, we do not indicate their number. In every case algorithm B^+ found a feasible solution for $T = lb_{\max}$ which, therefore, was always optimal. In

Table 5. Results on Case A and Case B instances.

	rcap	ub(Tav)	Opt	CPU time	#frac cut	#(S_1, S_2)
Case A	120	27*	27	0.04	0	0
	100	27*	27	0.06	0	0
	80	27*	27	0.23	1	7
	75	29	27	0.25	27	11
	70	29	27	0.29	19	17
	65	30	28	0.57	8	15
Case B	240	21*	21	0.09	4	1
	200	21*	21	0.11	0	0
	180	24	21	0.06	0	0
	120	27	25	14.46	0	93

addition, the optimal solution was always found already in the root node of the branch-and-bound tree. For all but the last instance in Case A, J. Coelho obtained the same bounds as ours by using metaheuristics [4]. However, we improved on the upper bounds of Tavares in five out of ten cases in a short computation time.

7.3. Evaluation on PSPLIB

We also evaluated our makespan minimization procedure on the well-known PSPLIB instances that were originally designed for the non-preemptive resource constrained project scheduling problem [18]. In each PSPLIB instance there are 4 resources each having a finite constant internal capacity and no external capacity. Each activity i has a deterministic duration p_i, which we interpret as the inverse of the maximum intensity of activity i, that is, let $a^i = 1/p_i$. In each time period of execution, activity i requires $\rho_{i,k}$ units of resource k. Therefore, we set $q^i_k = \rho_{i,k} p_i$, for each resource k.

We tested our algorithm on 30-activity ($j30$) and 60-activity ($j60$) instances, respectively. On the $j30$ instances the run-time limit of the two phases were set as $\tau_1 = \tau_2 =$ 300 seconds, whereas on $j60$ instances we set $\tau_1 = 1800$, $\tau_2 = 600$ seconds. We divide the $j30$ as well as $j60$ instances into two subclasses: those on which the total CPU time (through all invocations of B^+) of the algorithm was less than 60 seconds and the rest. Thus we have four classes: $j30^-$, $j60^-$ (less than 60 seconds total CPU time), $j30^+$, $j60^+$ (more than 60 seconds total CPU time). Every $j30$ instace has been solved, i.e., a feasible solution has been found, but fourteen $j60^+$ instances have remained unsolved. Table 6 has four rows corresponding to these subclasses. The second column indicates the number of instances in the class, and, when different, the number of instances solved. Moreover, the third and fourth columns indicate the average as well as minimum ratio of the strongest lower bound to the best upper bound over all solved instances in the class.

Table 6. Results on PSPLIB instances

	#inst.	avg.	min	nodes	CPU	#flow	#frac	#(S_1, S_2)
		lb/ub	lb/ub		time	cut	cut	cut
$j30^-$	413	1.00	1.00	60.95	1.46	6.60	3.36	15.84
$j30^+$	67	0.95	0.72	2613	442	72	11	554
$j60^-$	380	1.00	1.00	2.7	2.07	2.00	1.35	12.92
$j60^+$	100/86	0.93	0.67	591	2118	30	43	903

We measured the total CPU time and gathered the total search tree nodes, and the total number of flow, fractional and (S_1, S_2) cuts added through all invocations of B^+ while solving each instance. The averages of these data over all solved instances in a class are given in the next five columns of the table. Notice that the majority of instances were solved to optimality in less than 3 seconds. On hard instances the computation time can be 1 hour or more, but was never more than 1.5 hours.

8. Conclusions and future work

In this paper we have proposed an efficient branch-and-cut algorithm for solving the strongly NP-hard RCPSVP problem. The success of the algorithm is due partly to the new LP-formulation of the precedence constraints, and partly to the investigation of the polyhedra of scheduling two variable-intensity activities connected by a precedence constraint. In order to solve harder problem instances than those on which the algorithm performs quite well, new polyhedral results are necessary which can be the topic of future research.

Acknowledgements. The author is grateful to the two anonymous referees whose comments helped to significantly improve the presentation and to eliminate some errors. We are thankful to Willy Herroelen for providing us with an extract from [6]. We are indebted to Erwin Hans and José Coelho who provided us with benchmark instances and shared with us their experience in solving the RCPSVP and the makespan minimization problem, respectively. The supports of the Hungarian NRDP grant "Digital Factories, Production Networks" No. 2/040/2001, and OTKA T046509 are gratefully acknowledged.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows. Prentice Hall, Englewood Cliffs, New Jersey, 1993
2. Blazewicz, J., Ecker, K.H., Pesch, E., Schmidt, G., Weglarz, J.: Scheduling Computer and Manufacturing Processes. 2nd edition, Springer-Verlag, Berlin, Heidelberg, New-York, 2001
3. Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models and methods. Eur. Jour. Ops. Res. **112**, 3–41 (1999)
4. Coelho, J.: Personal communication. 2003
5. De Boer, R.: Resource-Constrained Multi-Project Management - A Hierarchical Decision Support System. Ph.D. thesis, Twente University Press, The Netherlands, 1998
6. Demeulemeester, E.L., Herroelen, W.S.: Project Scheduling: A Research Handbook. Kluwer Academic Publ., International series in operations research and management science **49**, 2002
7. Ford, L.R., Fulkerson, D.R.: Maximal flow through a network. Canadian J. Math. **8** 399–404 (1956)
8. Garey, M.R., Johnson, D.S.: Computers and Intractability: A guide to the Theory of NP-Completeness. W. H. Freeman and Co., San Francisco, 1979
9. Gonzalez, T., Sahni, S.: Flowshop and jobshop schedules: Complexity and approximation. Oper. Res. **26/1**, 36–52 (1978)
10. Graham, R.L., Lawler, E.L., Lenstra, J.K., Rinnooy Kan, A.H.G.: Optimization and approximation in deterministic sequencing and scheduling theory: a survey. Ann. Discrete Math. **5**, 287–326 (1979)
11. Hans, E.W.: Resource loading by branch-and-price techniques. Ph.D. thesis, Twente University Press, The Netherlands, 2001
12. Hans, E.W.: personal communication. 2003
13. Herroelen, W.S., De Reyck, B., Demeulemeester, E.L.: Resource-constrained project scheduling: a survey of recent developments. Comput. Ops. Res. **25/4**, 279–302 (1998)
14. Hoffman, A.J., Kruskal, J.B.: Integral boundary points of convex polyhedra. In: H.W. Kuhn, A.W. Tucker (eds.), Linear Inequalities and Related Systems, Princeton University Press, Princeton, 1956, pp. 233–246
15. ILOG: ILOG CPLEX 7.5, User's manual. ILOG S.A, Gentilly, France, 2001

16. Jünger, M., Reinelt, G., Thienel, S.: Practical problem solving with cutting plane algorithms in combinatorial optimization. DIMACS Ser. in Discr. Math. and Theor. Comput. Sci. **20**, 111–152 (1995)
17. Kolisch, R., Sprecher, A., Drexl, A.: Characterization and generation of a general class of resource-constrained project scheduling problems. Technical Report, 301, University of Kiel, Germany, 1992
18. Kolisch, R., Sprecher, A.: PSPLIB – a project scheduling library. Eur. Jour. Ops. Res. **96**, 205–216 (1997)
19. Leachman, R.C., Dincerler, A., Kim, S.: Resource constrained scheduling of projects with variable-intensity activities. IIE Trans. **22/1**, 31–40 (1990)
20. Nemhauser, G.L., Wolsey, L.A.: Integer and Combinatorial Optimization. John Wiley & Sons, New York, 1988
21. Padberg, M.W., Van Roy, T.J., Wolsey, L.A.: Valid linear inequalities for fixed charge problems. Oper. Res. **33/4**, 842–861 (1985)
22. Padberg, M.W., Rinaldi, G.: Optimization of a 532 city symmetric traveling salesman problem by branch and cut. Oper. Res. Lett. **6**, 1–7 (1987)
23. Schrijver, A.: Theory of linear and integer programming. John Wiley & Sons, Chicester, 1986
24. Tavares, L.V.: Advanced models for project management. Kluwer Academic Publishers, 1998, pp. 177–216
25. Tavares, L.V.: A review of the contribution of operational research to project management. Eur. Jour. Ops. Res. **136**, 1–18 (2002)
26. Weglarz, J.: Project scheduling with continuously-divisible doubly constrained resources. Management Science **27/9**, 1040–1053 (1981)

Appendix I

Proof of Lemma 1. Suppose first that the instance of PFSP has a solution σ. If activity i represents the processing of job j on processor k then set $x_t^i = \sigma(j,k,t)/\pi_{j,k}$. Moreover, set $z_t^i = 1$ if and only if there exists $t' \geq t$ with $x_{t'}^i > 0$, and 0 otherwise. Finally, set $y_t^k = 0$ for all resources (processors) k and time periods t. One can verify that the above (x, y, z) satisfies system (1).

Conversely, suppose system (1) admits a feasible solution (x', y', z'). We will prove that in this case there exists a possibly different feasible solution (x, y, z) of the system such that $x_t^i = a^i$ or 0, $y = y'$ and $z = z'$. From this claim it follows that the schedule σ, defined by $\sigma(j,k,t) = x_t^i/a^i$ for each j, k, t and i such that activity i represents the processing of job j by processor k, is a feasible solution for the PFSP instance. It is enough to verify property (ii), as the other two requirements against feasible schedules easily follow from the properties of feasible solutions of system (1). We have the following estimation on the number of jobs requiring a processor k in time period t:

$$\sum_j \sigma(j,k,t) = \sum_i x_t^i/a^i = \sum_{i:q_k^i>0, t\in\{r^i,\ldots,d^i\}} q_k^i \cdot x_t^i \leq 1.$$

Here, the second summation is over all activities i representing the processing of a job j by processor (resource) k. The second equality follows from the definition of a^i and q_k^i and the inequality is due to the fact that (x, y, z) satisfies (1e) and y must be all 0 since all external resource capacities are 0.

To prove our claim, observe that in any feasible solution of (1), z is a 0/1 vector. Let U^j be the subset of time points where activity j may be executed, i.e., $z_t^j = 1$ and $z_t^i = 0$ for all i such that $(i, j) \in A$ (cf. constraints (1b) and (1c)). Since all components of z are fixed to 0 or 1, we may rewrite the system (1) as follows:

$$\sum_{t=r^i}^{d^i} x_t^i = 1, \quad i \in N, \tag{10a}$$

$$\sum_{i:q_k^i>0} (1/a^i) \cdot x_t^i \le 1, \quad t \in \{1, \dots, T\} \tag{10b}$$

$$0 \le x_t^i \le a^i, \quad i \in N,\ t \in U^i \tag{10c}$$

$$x_t^i = 0, \quad i \in N,\ t \notin U^i. \tag{10d}$$

Multiply the constraints (10a),(10c) and (10d) by $1/a^i$ and replace $(1/a^i) \cdot x_t^i$ by a new variable $\tilde{x}_t^i$. The resulting system, depicted below, is a transportation problem.

$$\sum_{t=r^i}^{d^i} \tilde{x}_t^i = 1/a^i, \quad i \in N \tag{11a}$$

$$\sum_{i:q_k^i>0} \tilde{x}_t^i \le 1, \quad t \in \{1, \dots, T\} \tag{11b}$$

$$0 \le \tilde{x}_t^i \le 1, \quad i \in N,\ t \in U^i \tag{11c}$$

$$\tilde{x}_t^i = 0, \quad i \in N,\ t \notin U^i. \tag{11d}$$

Since the constraint matrix of (11) is totally unimodular and the right hand side is integral (since every a^i is the inverse of some $\pi_{j,k} \in \mathbb{Z}^+$), the Hoffman-Kruskal theorem on unimodular matrices [14] implies that there exists an integral solution $\overline{x}$. Since $\overline{x}_t^i$ is between 0 and 1, $\overline{x}$ is a 0-1 vector. Then, $x_t^i = a^i \cdot \overline{x}_t^i$ is a solution of (10) such that $x_t^i = 0$ or a^i, as claimed. □

Appendix II

To compute the minimum project duration we can embed algorithm B^+ into a dichotomic search procedure. By setting all activity deadlines to a given value T, algorithm B^+ can decide whether a feasible schedule of makespan at most T exists. As we will call B^+ several times, a run-time limit is set for each invocation. Therefore, branch-and-cut can terminate with either a feasible solution, or it may prove that no feasible solution exits, or it stops by reaching the time limit without finding a feasible solution or proving infeasibility.

The procedure first computes the lower bound $lb_{\max} = \max\{lb_p, lb_r\}$, noting that if there are two or more resources, then lb_r is the maximum of the resource based lower bound over all resources. Then it tries to determine an upper bound on the minimum project duration by calling algorithm B^+ with activity deadlines $T = lb_{\max}$, $(3/2)lb_{\max}$, $2lb_{\max}$, respectively, in this order, and with a run-time limit τ_1 for each call. If it fails in all three cases, then the algorithm terminates with no solution found. On the other hand, if already for $T = lb_{\max}$ a feasible solution is found, it stops, that solution is optimal, since $lb_{\max}$ is a lower bound on the minimum makespan. Otherwise, a dichotomic search is performed between $T - lb_{\max}/2$ and T and a run-time limit τ_2 is set for each call of B^+. The output of the algorithm are the least bound ub for which a feasible solution has been found, and the largest lb such that infeasibility of $lb - 1$ has been proven.