Project Scheduling with Changeover Times: A Branch-and-Bound Approach

Christoph Schwindt

Institut für Wirtschaftstheorie und Operations Research
University of Karlsruhe
D-76128 Karlsruhe, Germany
e-mail: schwindt@wior.uni-karlsruhe.de

Abstract. In this paper we consider the scheduling of projects with sequence-dependent changeover times between activities. Those changeover times arise in multisite projects, where activities sharing common renewable resources are performed at distributed locations. In the context of production scheduling, the changeover times correspond to setup or cleaning times of production facilities. We study the problem of optimizing a time-based or financial project objective like the project duration or net present value subject to temporal and resource constraints. The temporal constraints are given by prescribed minimum and maximum time lags between the starts of activities. The resource constraints refer to the scarcity of renewable and storage resources. We present a branch-and-bound algorithm, which is based on iteratively replacing the resource constraints by precedence relationships between activities that compete for the same resources. Those precedence relationships can be formulated as minimum or maximum time lags. For renewable resources with sequence-dependent changeover times, appropriate sets of competing activities can be obtained by solving a maximum-cut problem in a schedule-induced precedence graph. We show how for the latter problem, an equivalent standard maximum-flow problem can be constructed.

Key words: Operations planning and scheduling, project scheduling, changeover times, branch-and-bound, maximum-flow problem

1 Introduction

A *project* is a one-time undertaking decomposing into interacting *activities*, which require time and resources for their execution. Let $V = \{0, 1, \ldots, n, n+1\}$ be the set of all activities. 0 and $n+1$ denote the dummy project beginning and project termination activities. Each activity $i \in V$ is associated with a *duration* $p_i \in \mathbb{Z}_{\geq 0}$, where $p_0 = p_{n+1} = 0$. Let S_i be the start time of activity i to be determined when scheduling the project. A vector $(S_i)_{i \in V}$ with $S_0 = 0$ and $S_i \geq 0$ for all $i \in V$ is called a (project) *schedule*. We assume that during their execution, activities must not be interrupted, i.e., $S_i + p_i$ is the completion time of activity i. Hence, a schedule S uniquely defines the timetable of the project.

For certain activities $i, j \in V$, a minimum time lag $\delta_{ij} \in \mathbb{Z}$ between the start of activity i and the start of activity j may be prescribed. The minimum time lags define *temporal constraints*

$$S_j - S_i \geq \delta_{ij} \quad ((i,j) \in E) \tag{1}$$

where $E \subset V \times V$. Negative minimum time lags δ_{ij} can be interpreted as positive maximum time lags $-\delta_{ij}$ between the starts of activities j and i. In particular, we assume that there exists a maximum time lag of $\overline{d}$ time units (the project deadline) between the project beginning 0 and the project termination $n+1$, i.e., $\delta_{(n+1)0} = -\overline{d}$. A schedule S satisfying the temporal constraints is termed *time-feasible*. The set of all time-feasible schedules is denoted by $\mathcal{S}_T$.

For performing the project, a set $\mathcal{R}^\rho$ of *renewable resources* k is used. The *capacity* $R_k \in \mathbb{N}$ of resource $k \in \mathcal{R}^\rho$ specifies the number of resource units available. Each activity $i \in V$ occupies $r_{ik} \in \mathbb{Z}_{\geq 0}$ units of resource k during its execution in time interval $[S_i, S_i + p_i[$. Let $V_k := \{i \in V \mid r_{ik} > 0\}$ be the set of all activities using resource k and put $\overline{V}_k := V_k \cup \{0, n+1\}$. Between the execution of two consecutive activities $i, j \in \overline{V}_k$ on one and the same unit of resource k, a *changeover time* $\vartheta^k_{ij} \in \mathbb{Z}_{\geq 0}$ is needed during which the unit is not available for processing, where for notational convenience $\vartheta^k_{0i} := \vartheta^k_{i(n+1)} := 0$ for all $i \in V_k$ and $\vartheta^k_{ii} := \infty$ for all $i \in \overline{V}_k$. We assume that the changeover times satisfy the *weak triangle inequalities*

$$\vartheta^k_{hi} + p_i + \vartheta^k_{ij} \geq \vartheta^k_{hj} \quad (h, i, j \in V_k) \tag{2}$$

Conditions (2) mean that we cannot save changeover time by processing extra activities. We call a schedule S *changeover-feasible* if for every resource $k \in \mathcal{R}^\rho$, the activities from set V_k can be assigned to r_{ik} units of resource k each in such a way that for any two activities sharing common units, the time interval between the completion of the first and the start of the second activity is sufficiently large for carrying out the changeover (see Neumann 2003). Given schedule S, strict order

$$O_k(S) := \{(i, j) \in \overline{V}_k \times \overline{V}_k \mid S_j \geq S_i + p_i + \vartheta^k_{ij}\}$$

in ground set $\overline{V}_k$ provides the set of all pairs (i, j) for which S allows for a changeover from i to j on resource k. Accordingly, i and j may share some units of resource k precisely if $\{i, j\}$ is a *chain* in strict order $O_k(S)$, i.e., either $(i, j) \in O_k(S)$ or $(j, i) \in O_k(S)$. A set $U \subseteq V$ which does not include any chain $\{i, j\}$ is referred to as an *antichain* in $O_k(S)$. We call an antichain U in $O_k(S)$ with maximum weight $r^\rho_k(S) := \sum_{i \in U} r_{ik}$ an *active set* for resource k given schedule S. Let $\mathcal{A}_k(S)$ be such an active set. By definition, no two activities $i, j \in \mathcal{A}_k(S)$ can occupy common units of resource k, and thus $r^\rho_k(S)$ coincides with the number of units of resource k that are required for processing all activities $i \in V_k$ (including the changeovers).

Now let Φ^k_{ij} denote the number of units of resource k which are to be changed over from i to j given schedule S. Then $\Phi^k := (\Phi^k_{ij})_{(i,j) \in O_k(S)}$ can be interpreted as a $(0, n+1)$-flow in the precedence graph $G_k(S)$ of $O_k(S)$ with node set $\overline{V}_k$ and arc set $O_k(S)$, and

$$r^\rho_k(S) = \min_{\Phi^k \geq 0} \Big\{ \sum_{i \in V_k} \Phi^k_{0i} \;\Big|\; \sum_{(j,i) \in O_k(S)} \Phi^k_{ji} = \sum_{(i,j) \in O_k(S)} \Phi^k_{ij} = r_{ik} \text{ for all } i \in V_k \Big\}$$

In other words, $r^\rho_k(S)$ equals the value of a minimum $(0, n+1)$-flow Φ^k in $G_k(S)$ saturating the node capacities r_{ik} (see Trautmann 2003). Finding such a minimum flow can be done in $\mathcal{O}(n^3)$-time by applying well-known network flow techniques (see, e.g., Bang-Jensen and Gutin 2000, Sect. 3.9). The (renewable-)*resource constraints* can now be formulated as

$$r^\rho_k(S) \leq R_k \quad (k \in \mathcal{R}^\rho) \tag{3}$$

In practice, aside from the finite capacities of renewable resources, often the limited availability of material or funds and the scarcity of storage capacity or assembly space have to be taken

into account when scheduling the project. Those requirements can be modelled by using the concept of *cumulative* or *storage resources* introduced by Neumann and Schwindt (2002), which have also been treated by Beck (2002) and Laborie (2003). Storage resources can be regarded as stock-keeping facilities whose inventories are depleted and replenished over time by the activities of the project. Let $\mathcal{R}^\sigma$ denote the set of all storage resources under consideration. For each resource $k \in \mathcal{R}^\sigma$, a minimum inventory level (*safety stock*) $\underline{R}_k \in \mathbb{Z}_{\geq 0}$ and a maximum inventory level (*storage capacity*) $\overline{R}_k \in \mathbb{Z}_{\geq 0}$ are given. Activities $i \in V$ have (storage) demands r_{ik} for the resources $k \in \mathcal{R}^\sigma$, where $r_{ik} > 0$ means that the inventory of resource k is replenished by r_{ik} units at the completion of activity i and $r_{ik} < 0$ stands for a depletion of $-r_{ik}$ units at the start of activity i. r_{0k} is the initial stock of resource k.

By $V_k^- := \{i \in V \mid r_{ik} < 0\}$ and $V_k^+ := \{i \in V \mid r_{ik} > 0\}$ we denote the sets of activities depleting and replenishing, respectively, the inventory of resource k. Then the *active set* $\mathcal{A}_k(S,t) := \{i \in V_k^- \mid S_i \leq t\} \cup \{i \in V_k^+ \mid S_i + p_i \leq t\}$ for resource k at time t given schedule S is the set of all activities whose joint resource demands provide the inventory level

$$r_k^\sigma(S,t) := \sum_{i \in \mathcal{A}_k(S,t)} r_{ik}$$

of resource k at time t. The *inventory constraints*

$$\underline{R}_k \leq r_k^\sigma(S,t) \leq \overline{R}_k \quad (k \in \mathcal{R}^\sigma,\ 0 \leq t \leq \overline{d}) \tag{4}$$

ensure that during the execution of the project, the inventory levels remain between the safety stocks and storage capacities. A schedule S satisfying the inventory constraints is called *inventory-feasible*. A *feasible schedule* is time-, changeover-, and inventory-feasible.

The problem (P) to be solved consists of finding a feasible schedule S minimizing some given time-based or financial objective function $f : \mathcal{S}_T \to \mathbb{R}$:

$$\left.\begin{array}{lll} \text{Minimize} & f(S) & \\ \text{subject to} & S_j - S_i \geq \delta_{ij} & ((i,j) \in E) \\ & S_0 = 0,\ S_i \geq 0 & (i \in V) \\ & r_k^\rho(S) \leq R_k & (k \in \mathcal{R}^\rho) \\ & \underline{R}_k \leq r_k^\sigma(S,t) \leq \overline{R}_k & (0 \leq t \leq \overline{d},\ k \in \mathcal{R}^\sigma) \end{array}\right\} \quad \text{(P)}$$

A solution to problem (P) is termed an *optimal schedule*. For what follows, we suppose that first, f is lower semicontinuous and second, f is regular or convexifiable. The first condition guarantees that an optimal schedule exists, provided that there is a feasible schedule. The second condition implies that the *resource relaxation* of problem (P), i.e., the problem of minimizing f on the set $\mathcal{S}_T$ of time-feasible schedules, can be solved efficiently. Examples of regular objective functions are the project duration, the maximum lateness, or the weighted tardiness. Convexifiable objective functions which have been investigated in literature are the weighted earliness-tardiness and the (negative) net present value of the project (see Schwindt 2002, Sect. 2.3, and Neumann 2003).

2 Changeover- and inventory-feasibility of schedules

The basic idea for solving project scheduling problem (P) is to delete the resource and inventory constraints and to transform the (generally infeasible) schedule arising from the resource relaxation into a feasible schedule. The transformation is achieved by first, refining the initial resource relaxation through precedence relationships between activities from active sets that induce the violation of a resource or inventory constraint and second, solving the relaxation again. Both steps are alternated until a feasible schedule has been found. By enumerating the alternative sets of precedence relationships resolving the resource or inventory conflicts, one obtains the enumeration scheme of our branch-and-bound procedure.

Activity sets $F \subseteq V$ causing conflicts on renewable resources $k \in \mathcal{R}^\rho$ are called *k-forbidden sets*. More precisely, we say that set F is k-forbidden if

$$\sum_{k \in F} r_{ik} > R_k$$

A set $F \subseteq V$ may cause a conflict on a storage resource $k \in \mathcal{R}^\sigma$ due to two reasons. Either the joint resource demands by activities from set F fall below the safety stock or they exceed the storage capacity of the resource. Sets F with

$$\sum_{i \in F} r_{ik} < \underline{R}_k \quad (\text{or } \sum_{i \in F} r_{ik} > \overline{R}_k)$$

are called *k-shortage sets* (or *k-surplus sets*, respectively).

The following theorem, which has been proved by Bartusch et al. (1988) for the special case of vanishing changeover times, shows how to remove violations of the resource constraints by defining minimum time lags $\delta_{ij} = p_i + \vartheta_{ij}^k$ between activities i, j from forbidden sets.

Theorem 1 (Bartusch et al. 1988) *Schedule S is changeover-feasible if and only if for each k-forbidden set F there exist two activities $i, j \in F$ such that $S_j \geq S_i + p_i + \vartheta_{ij}^k$.*

As we have seen in Section 1, the changeover-feasibility of a schedule can be efficiently checked by solving a minimum-flow problem. Now assume that we are given some schedule S such that $r_k^\rho(S) > R_k$ for some $k \in \mathcal{R}^\rho$. Then the question arises how to find a k-forbidden set F that does not satisfy the condition of Theorem 1. Clearly, the latter set necessarily forms an antichain in strict order $O_k(S)$, for otherwise we would have two activities $i, j \in F$ with $S_j \geq S_i + p_i + \vartheta_{ij}^k$. From our analysis in Section 1 it further follows that active set $\mathcal{A}_k(S)$ is k-forbidden. Since $\mathcal{A}_k(S)$ is an antichain in $O_k(S)$, the k-forbidden set sought can be chosen to be equal to $\mathcal{A}_k(S)$.

In what follows, we show how to compute set $\mathcal{A}_k(S)$ for given schedule S. We start by transforming the node capacities r_{ik} of flow network $G_k(S)$ from Section 1 into corresponding arc capacities by splitting up every node $i \in V_k$ into two nodes i' and i'', which are linked by an arc (i', i'') with lower and upper capacities $l_{i'i''} = u_{i'i''} = r_{ik}$. The remaining arcs in $G_k(S)$ are associated with zero lower and infinite upper capacities. Let $\overline{G}_k(S)$ denote the resulting (modified) flow network. In Möhring (1985) it is shown that *maximum $(0, n+1)$-cuts C* in such a network only contain forward arcs. As a consequence, any path from 0 to $n+1$ in $\overline{G}_k(S)$ is only cut once, which implies that $U := \{i \in V_k \mid (i', i'') \in C\}$ is an antichain in $O_k(S)$. Since

$l_{0i'} = l_{i''(n+1)} = 0$ for all $i \in V_k$ and $l_{i''j'} = 0$ for all $(i,j) \in O_k(S)$, the capacity of cut C equals $\sum_{(i',i'')\in C} l_{i'i''} = \sum_{i\in U} r_{ik}$. On the other hand, from the strong duality relationship between the minimum-flow and maximum-cut problems it follows that the capacity of C coincides with the minimum flow value $r_k^\rho(S)$ and thus $U = \mathcal{A}_k(S)$.

We proceed by explaining the way in which a maximum $(0, n+1)$-cut C can be obtained by one application of a standard maximum-flow procedure like the preflow-push algorithm by Cherkassky and Goldberg (1997), which in addition to the maximum flow also provides a minimum cut in the flow network. At first, we notice that a *feasible* flow $\dot{\Phi}^k$ of value $\sum_{i\in V_k} r_{ik}$ in $\overline{G}_k(S)$ can easily be constructed by setting $\dot{\Phi}^k_{0i'} := \dot{\Phi}^k_{i'i''} := \dot{\Phi}^k_{i''(n+1)} := r_{ik}$ and $\dot{\Phi}^k_{i''j'} := 0$ for all $i, j \in V_k$. Since we want to minimize the value of the $(0, n+1)$-flow, we subsequently send a maximum amount of $\dot{\Phi}^k_{ij}$ back from $n+1$ to 0, which can be done by computing a maximum $(n+1, 0)$-flow $\ddot{\Phi}^k$ in the *residual network* of $\overline{G}_k(S)$ and $\dot{\Phi}^k$. The minimum $(0, n+1)$-flow in $\overline{G}_k(S)$ then equals $\Phi^k = (\Phi^k_{ij})_{(i,j)\in O_k(S)}$ where $\Phi^k_{ij} = \dot{\Phi}^k_{ij} + \ddot{\Phi}^k_{ij} - \ddot{\Phi}^k_{ji}$ for all $(i,j) \in O_k(S)$. The minimum $(n+1, 0)$-cut in the residual network, which is yielded along with maximum flow $\ddot{\Phi}^k$, then coincides with the maximum $(0, n+1)$-cut C in $\overline{G}_k(S)$. In sum, given a schedule S, both the changeover-feasibility of S can be checked and if S is not changeover-feasible, a k-forbidden set F with $S_j < S_i + p_i + \vartheta^k_{ij}$ for all $i, j \in F$ can be determined by solving $|\mathcal{R}^\rho|$ maximum-flow problems.

We now turn to the inventory-feasibility of schedules. Neumann and Schwindt (2002) have shown the following counterpart of Theorem 1 for storage resources.

Theorem 2 (Neumann and Schwindt 2002) *Schedule S is inventory-feasible if and only if*

1. *for each k-shortage set F with $k \in \mathcal{R}^\sigma$ there exist two activities $i \in V_k^+ \setminus F$ and $j \in F \cap V_k^-$ such that $S_j \geq S_i + p_i$, and*
2. *for each k-surplus set F with $k \in \mathcal{R}^\sigma$ there exist two activities $i \in V_k^- \setminus F$ and $j \in F \cap V_k^+$ such that $S_j \geq S_i - p_j$.*

Accordingly, an inventory shortage for some storage resource k at a time t can be sorted out by defining minimum time lags $\delta_{ij} = p_i$ between replenishing activities i outside active set $\mathcal{A}_k(S,t)$ and depleting activities j from set $\mathcal{A}_k(S,t)$. Symmetrically, an inventory excess for k at time t can be resolved by introducing minimum time lags $\delta_{ij} = -p_j$ between depleting activities i outside $\mathcal{A}_k(S,t)$ and replenishing activities from $\mathcal{A}_k(S,t)$. The latter (negative) minimum time lags represent maximum time lags of p_j time units between activities j and i. In difference to the case of renewable resources, the active sets and corresponding inventory levels can readily be obtained by sorting the start and completion times of activities $i \in V$ in increasing order.

3 Partitioning the feasible region

Given a resource or inventory conflict induced by some schedule, Theorems 1 and 2 tell us that the conflict can be settled by introducing specific time lags between activities. In general, for one and the same conflict alternative sets of time lags may be considered. For example, the resource conflict induced by a two-element k-forbidden set $\{i, j\}$ requires the introduction of

either $\delta_{ij} = p_i + \vartheta_{ij}^k$ or $\delta_{ji} = p_j + \vartheta_{ji}^k$. In what follows we are concerned with the question how alternative sets of time lags can be constructed in an appropriate way. For simplicity, we restrict ourselves to resource conflicts. The case of inventory conflicts can be dealt with analogously.

Let S be a schedule such that for some renewable resource $k \in \mathcal{R}^\rho$, active set $\mathcal{A}_k(S)$ is k-forbidden. We may enumerate sets of time lags $\delta_{ij} = p_i + \vartheta_{ij}^k$ resolving the conflict by considering all $\subseteq$-minimal collections P of pairs (i, j) such that any k-forbidden subset $F \subseteq \mathcal{A}_k(S)$ contains two activities i, j with $(i, j) \in P$. Now let $\mathcal{P}$ be the set of collections P obtained in that way and denote by $\mathcal{S}_T(P)$ the set of all time-feasible schedules satisfying the temporal constraints $S_j - S_i \geq \delta_{ij}$ for all $(i, j) \in P$. Then $\mathcal{P}$ generally contains distinct collections P and P' with $\mathcal{S}_T(P) \cap \mathcal{S}_T(P') \neq \emptyset$, which means that one and the same schedule S may be considered more than once during the enumeration process. This redundancy can only be avoided if the decomposition of the feasible region of problem (P) provided by the branch-and-bound procedure is a *partition* into nonintersecting schedule sets.

Such a partition can be obtained as follows. Let $\pi_1 = (i_1, j_1), \ldots, \pi_\nu = (i_\nu, j_\nu)$ be a numbering of all pairs (i, j) with $i, j \in \mathcal{A}_k(S),\ i \neq j$. We then generate ν disjoint subsets $\mathcal{Q}_\mu$ $(\mu = 1, \ldots, \nu)$ of the current search space $\mathcal{Q}$ where

$$\mathcal{Q}_\mu = [\mathcal{Q} \cap \mathcal{S}_T(\{\pi_\mu\})] \setminus [\cup_{\lambda=1}^{\mu-1} \mathcal{S}_T(\{\pi_\lambda\})]$$

The construction of subsets $\mathcal{Q}_\mu$ can be achieved by introducing, at the enumeration node belonging to pair $\pi_\mu = (i, j)$, the temporal constraint $S_j - S_i \geq p_i + \vartheta_{ij}^k$ and defining the reverse constraint $S_j - S_i < p_i + \vartheta_{ij}^k$ for all enumeration nodes belonging to pairs $\pi_{\mu+1}, \ldots, \pi_\nu$. By exploiting the fact that the feasible region of (P) is the union of *integral* polytopes, the reverse constraint can be tightened to temporal constraint $S_j - S_i \leq p_i + \vartheta_{ij}^k - 1$, which corresponds to a maximum time lag of $-\delta_{ji} = p_i + \vartheta_{ij}^k - 1$ time units between activities i and j. By using the tightened constraints, we obtain that all sets $\mathcal{Q}_\mu$ are closed.

4 Branch-and-bound procedure

The enumeration scheme of our branch-and-bound procedure for solving problem (P) is now as follows (cf. Algorithm 1). L is a list of search spaces and $\mathcal{C}$ denotes the set of candidate schedules to be generated. At first, we add the set of all time-feasible schedules $\mathcal{S}_T$ to list L and put $\mathcal{C} := \emptyset$. At each iteration we take some search space $\mathcal{Q}$ from list L and determine a minimizer S of f on $\mathcal{Q}$ (recall that since f is regular or convexifiable, the latter problem can be solved efficiently). We then compute active sets $\mathcal{A}_k(S)$ for all renewable resources k. If some of those sets is k-forbidden, we generate ν subsets $\mathcal{Q}_\mu$ of search space $\mathcal{Q}$ as described in Section 3 and add those subsets to list L. Otherwise, we proceed by checking the inventory constraints. To this end, we scan S for a start or completion time t of some activity $i \in V$ such that active set $\mathcal{A}_k(S, t)$ is a k-shortage or k-surplus set for some storage resource k. If no such point in time t is found, schedule S is (inventory-)feasible and is thus added to the set $\mathcal{C}$ of candidate schedules. Otherwise, the search space is again decomposed into disjoint subsets $\mathcal{Q}_\mu$, which are subsequently added to list L. We then take the next search space $\mathcal{Q}$ from list L and proceed in the same way until no more search spaces $\mathcal{Q}$ remain on list L. Eventually, we return the set $\mathcal{C}$ of candidate schedules found.

It follows from Theorems 1 and 2 that at the end of Algorithm 1, the set $\mathcal{C}$ of candidate schedules contains an optimal schedule if problem (P) is solvable. Furthermore, it is easily seen that the depth of the enumeration tree is $\mathcal{O}(n^2 \max\{|\mathcal{R}^\rho|, |\mathcal{R}^\sigma|\})$, since for each resource less than n^2 pairs (i, j) can be generated on a path from the root node to a leaf of the enumeration tree.

Algorithm 1 Enumeration scheme of branch-and-bound procedure

initialize list of search spaces $L := \{\mathcal{S}_T\}$ and set of candidate schedules $\mathcal{C} := \emptyset$;
repeat
  delete some search space $\mathcal{Q}$ from list L;
  if $\mathcal{Q} \neq \emptyset$ **then**
    determine minimizer S of f on $\mathcal{Q}$;
    for all $k \in \mathcal{R}^\rho$ **do** determine $\mathcal{A}_k(S)$ by solving maximum-flow problem;
    if $F := \mathcal{A}_k(S)$ is a k-forbidden set for some $k \in \mathcal{R}^\rho$ **then**
      generate all pairs $(i_1, j_1), \ldots, (i_\nu, j_\nu) \in F \times F \setminus \{(i, i) \mid i \in F\}$;
      for $\mu = 1, \ldots, \nu$ **do** put $\mathcal{Q}_\mu := \mathcal{Q}$;
      for $\mu = 1, \ldots, \nu$ **do**
        put $\mathcal{Q}_\mu := \mathcal{Q}_\mu \cap \{S \in \mathcal{S}_T \mid S_{j_\mu} \geq S_{i_\mu} + p_{i_\mu} + \vartheta^k_{i_\mu j_\mu}\}$ and add $\mathcal{Q}_\mu$ to list L;
        for $\lambda = \mu + 1, \ldots, \nu$ **do**
          put $\mathcal{Q}_\lambda := \mathcal{Q}_\lambda \cap \{S \in \mathcal{S}_T \mid S_{j_\mu} \leq S_{i_\mu} + p_{i_\mu} + \vartheta^k_{i_\mu j_\mu} - 1\}$;
    else ($*$ S is changeover-feasible $*$)
      if there is a time $t \in \cup_{i \in V}\{S_i, S_i + p_i\}$ such that $F := \mathcal{A}_k(S, t)$ is k-shortage or k-surplus set for some $k \in \mathcal{R}^\sigma$ **then**
        if F is a k-shortage set **then**
          generate all pairs $(i_1, j_1), \ldots, (i_\nu, j_\nu) \in (V^+_k \setminus F) \times (F \cap V^-_k)$;
          for $\mu = 1, \ldots, \nu$ **do** put $\mathcal{Q}_\mu := \mathcal{Q}$;
          for $\mu = 1, \ldots, \nu$ **do**
            put $\mathcal{Q}_\mu := \mathcal{Q}_\mu \cap \{S \in \mathcal{S}_T \mid S_{j_\mu} \geq S_{i_\mu} + p_{i_\mu}\}$ and add $\mathcal{Q}_\mu$ to list L;
            for $\lambda = \mu + 1, \ldots, \nu$ **do**
              put $\mathcal{Q}_\lambda := \mathcal{Q}_\lambda \cap \{S \in \mathcal{S}_T \mid S_{j_\mu} \leq S_{i_\mu} + p_{i_\mu} - 1\}$;
        else ($*$ F is a k-surplus set $*$)
          generate all pairs $(i_1, j_1), \ldots, (i_\nu, j_\nu) \in (V^-_k \setminus F) \times (F \cap V^+_k)$;
          for $\mu = 1, \ldots, \nu$ **do** put $\mathcal{Q}_\mu := \mathcal{Q}$;
          for $\mu = 1, \ldots, \nu$ **do**
            put $\mathcal{Q}_\mu := \mathcal{Q}_\mu \cap \{S \in \mathcal{S}_T \mid S_{j_\mu} \geq S_{i_\mu} - p_{j_\mu}\}$ and add $\mathcal{Q}_\mu$ to list L;
            for $\lambda = \mu + 1, \ldots, \nu$ **do**
              put $\mathcal{Q}_\lambda := \mathcal{Q}_\lambda \cap \{S \in \mathcal{S}_T \mid S_{j_\mu} \leq S_{i_\mu} - p_{j_\mu} - 1\}$;
      else ($*$ S is inventory-feasible $*$)
        put $\mathcal{C} := \mathcal{C} \cup \{S\}$;
until $L = \emptyset$;
return $\mathcal{C}$;

Finally, we summarize the results from an experimental performance analysis of a branch-and-bound method that is based on the enumeration scheme of Algorithm 1. The test set consists of 360 projects with five renewable resource, five storage resources, and 10, 20, 50, or 100 activities each. Those projects have been obtained by combining two test sets used by Franck et al. (2001)

and Neumann and Schwindt (2002) for project scheduling with renewable or storage resources, respectively. The changeover times have been drawn at random such that the expected value of changeover time ϑ_{ij}^k equals $0.25p_i$ ($k \in \mathcal{R}^\rho,\ i,j \in V_k$). The objective function f has been chosen to be the project duration, i.e., $f(S) = S_{n+1}$. The branch-and-bound algorithm has been coded in ANSI C. For solving the maximum-flow problems, we have used Cherkassky and Goldberg's implementation of the preflow-push algorithm. The tests have been performed on a PII personal computer with 333 MHz clock pulse, 128 MB RAM, and Windows NT 4.0 as operating system.

For the different numbers n of activities, Table 1 shows the percentages p_{opt}, p_{ins}, p_{nopt}, and p_{unk} of instances for which within an imposed running time limit of 100 seconds optimality of the schedule found has been shown, insolvability has been shown, a feasible but not necessarily optimal schedule has been found, or the solvability status has remained unknown. Δ_{LB} denotes the mean optimality gap for the best schedule found with respect to a lower bound LB on the minimum project duration. In case where the algorithm has provided an optimal schedule, LB coincides with the minimum project duration. Otherwise, LB equals the project duration which arises from solving the resource relaxation at the root node.

	$n = 10$	$n = 20$	$n = 50$	$n = 100$
p_{opt}	54.44 %	62.22 %	25.56 %	8.89 %
p_{ins}	45.56 %	31.11 %	24.44 %	15.56 %
p_{nopt}	0.00 %	6.67 %	38.89 %	48.89 %
p_{unk}	0.00 %	0.00 %	11.11 %	26.67 %
Δ_{LB}	0.00 %	4.03 %	9.95 %	7.25 %

Table 1: Computational results

The results displayed in Table 1 show that the algorithm performs well for the small instances with no more than 20 activities (for 174 of those 180 small projects, the enumeration could be completed within the prescribed time limit). The fraction of instances solved to optimality markedly decreases with increasing n. On the other hand, the values for the optimality gap indicate that the algorithm is able to provide schedules with good accuracy for projects including up to 100 activities. Nevertheless, for about one-fourth of the projects with 100 activities, the enumeration is stopped before having found a feasible schedule. This behavior suggests the use of a truncated branch-and-bound algorithm of type filtered beam search or limited discrepancy search when coping with large-size problem instances.

References

[1] Bang-Jensen, J. and G. Gutin (2000). *Digraphs: Theory, Algorithms and Applications*. Springer, Berlin.

[2] Bartusch, M., R.H. Möhring, and F.-J. Radermacher (1988). Scheduling project networks with resource constraints and time windows. *Annals of Operations Research* 16, 201–240.

[3] Beck, J.C. (2002). Heuristics for scheduling with inventory: Dynamic focus via constraint criticality. *Journal of Scheduling* 5, 43–69.

[4] Cherkassky, B.V. and A.V. Goldberg (1997). On implementing the push-relabel method for the maximum flow problem. *Algorithmica* 19, 390–410.

[5] Franck, B., K. Neumann, and C. Schwindt (2001). Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. *OR Spektrum* 23, 297–324.

[6] Laborie, P. (2003). Algorithms for propagating resource constraints in AI planning and scheduling: Existing approaches and new results. *Artificial Intelligence* 143, 151–188.

[7] Möhring, R.H. (1985). Algorithmic aspects of comparability graphs and interval graphs. In I. Rival (ed.): *Graphs and Orders*. D. Reidel Publishing Company, Dordrecht, pp. 41–101.

[8] Neumann, K. (2003). Project scheduling with changeover times: Modelling and applications. *Proceedings of IEPM 2003*.

[9] Neumann, K. and C. Schwindt (2002). Project scheduling with inventory constraints. *Mathematical Methods of Operations Research* 56, 513–533.

[10] Schwindt, C. (2002). *Introduction to Resource Allocation Problems in Project Management*. Habilitation thesis, University of Karlsruhe.

[11] Trautmann, N. (2003). Project scheduling with changeover times: Schedule feasibility and network flows. *Proceedings of IEPM 2003*.