5

Supplements

When coping with real-life resource allocation problems, some of the assumptions of our three basic project scheduling problems may be too restrictive. This chapter is dedicated to expansions of the basic models which permit us to cover some features that are frequently encountered in practice.

In Section 5.1 we deal with *break calendars*, which specify time intervals during which some renewable resources cannot be used (such as weekends or night shifts, where skilled staff is not available). In that case, it is often necessary to relax the requirement that activities must not be interrupted when being in progress. Instead, we assume that the execution of certain activities can be suspended during breaks, whereas other activities still must not be interrupted. We explain how to perform temporal scheduling computations in presence of break calendars and outline how the enumeration scheme for regular objective functions discussed in Section 3.1 can be generalized to this problem setting.

When performing projects whose activities are distributed over different locations sharing common resources like manpower, heavy machinery, or equipment, *changeover times* for tear down, transportation, and reinstallation of resource units have to be taken into account. During the changeover, those resource units are not available for processing activities. Due to the transportation of resource units, the changeover times are generally sequence-dependent, which means that the time needed for changing over a resource unit between the execution of two consecutive activities depends on both activities. In Section 5.2 we show how to adapt the relaxation-based approaches to the occurrence of sequence-dependent changeover times.

In many applications of project management, the assignment of resources to the project activities is not (completely) predetermined by technology. We may then perform certain activities in *alternative execution modes*, which differ in durations, time lags, and resource requirements. The execution modes of an activity reflect tradeoffs between the time and resource demands. For example, the duration of an activity may be shortened by increasing the number of allotted resource units (time-resource tradeoff) or some resources used may be replaced by other resources (resource-resource tradeoff). If in that case the selection of an appropriate execution mode for each activity in the project planning phase is deferred from the time and resource estimations to the resource allocation step, we obtain a multi-mode resource allocation problem. In Section 5.3 we are concerned with relaxation-based procedures for solving multi-mode resource allocation problems with finitely many execution modes.

As we have seen in Section 1.3, the concept of (discrete) cumulative resources offers a straightforward way of modelling constraints arising from discrete material flows in assembly environments. Sometimes, however, inventories of intermediate products are not depleted and replenished batchwise at the occurrence of certain events but rather continuously over the execution time of consuming and producing real activities. Such continuous material flows are, for example, typical of mass production in the process industries. Material flows may also be semicontinuous, which means that facilities may be operated in batch or continuous production modes. In Section 5.4 we develop the concept of *continuous cumulative resources* and we propose a relaxation-based approach to solving resource allocation problems with the latter type of resources and convex objective functions. Resource conflicts are stepwise resolved by introducing linear constraints which ensure that at the start or completion of some activity, the inventory level is between the safety stock and the storage capacity. For each activity we branch over the alternatives whether or not the activity contributes to settling the resource conflict in question.

In the following Sections 5.1 to 5.4 we closely follow the presentation in the book of Neumann et al. (2003*b*), Sects. 2.11, 2.14, 2.15, and 2.12.2.

5.1 Break Calendars

In many real-life projects, certain renewable resources are not available during breaks like weekends or scheduled maintenance times. Scheduling the activities subject to break calendars is termed *calendarization*. For what follows, we assume that some real activities may be interrupted during a break, whereas others must not be interrupted due to technical reasons. Hence, the set of all real activities V^a decomposes into the set V^a_{bi} of all (break-)interruptible activities and the set V^a_{ni} of all non-interruptible activities. The processing of interruptible activities $i \in V^a_{bi}$ can only be stopped at the beginning of a break and has to be resumed at the end of the break. This assumption distinguishes calendarization from preemptive project scheduling problems, where activities may be interrupted at any point in time (see, e.g., Demeulemeester and Herroelen 1996). Furthermore, for each interruptible activity $i \in V^a_{bi}$, a *minimum execution time* $e_i \in \mathbb{N}$ is prescribed during which i has to be in progress without being suspended, e.g., $e_i = 1$. To simplify notation, we set $e_i := p_i$ for non-interruptible activities $i \in V^a_{ni}$ and assume that for activities $i \in V^a_{bi}$, the time between any two successive breaks is not less than e_i.

In this section we first describe procedures presented by Franck et al. (2001*a*) for the temporal scheduling of projects subject to break calendars for activities and prescribed time lags. We then briefly sketch how the relaxation-based approach for regular objective functions discussed in Section 3.1 can be adapted to the presence of break calendars. Preliminary versions of the temporal scheduling methods have been devised by Zhan (1992) and Franck (1999), Sect. 3.3. An alternative approach can be found in Trautmann (2001*b*). Here, the calendar-dependent precedence relationships between activities are taken into account by distinguishing between start-to-start, start-to-completion, completion-to-start, and completion-to-completion time lags.

A *break calendar* can be regarded as a right-continuous step function $b : \mathbb{R} \to \{0, 1\}$ where $b(t) = 0$ if time $t < 0$ or if t falls into a break, and $b(t) = 1$, otherwise. $\int_t^{t'} b(\tau)d\tau$ is the *total working time* in interval $[t, t'[$. In practice, different renewable resources $k \in \mathcal{R}^\rho$ may have different calendars. We then obtain the corresponding *activity calendars* b_i for activities $i \in V^a$ by setting $b_i(t) := 0$ exactly if i requires some resource $k \in \mathcal{R}^\rho$ which is not available at time t. If $b_i(t) = 0$, we have to suspend the execution of activity $i \in V^a_{bi}$ being in progress at time t. For activities $i \in V^a_{ni}$, the time interval between the start and completion of i must not contain any time t where $b_i(t) = 0$.

The constraints arising from minimum execution times e_i can be stated as follows:

$$b_i(\tau) = 1 \quad (i \in V^a,\ S_i \leq \tau < S_i + e_i) \tag{5.1}$$

If $i \in V^a_{ni}$, (5.1) means that the execution of i must not be interrupted by a break.

Let $C_i \geq S_i + p_i$ again denote the completion time of activity $i \in V^a$. In interval $[S_i, C_i[$, activity i is in progress at time t precisely if $b_i(t) = 1$. Thus, given start time S_i, the completion time $C_i(S_i)$ of i is uniquely determined by

$$C_i(S_i) = \min\{t \geq S_i + p_i \mid \int_{S_i}^t b_i(\tau)d\tau = p_i\}$$

Clearly, minimum and maximum time lags may depend on calendars, too. For example, a precedence constraint between activities i and j refers to the completion time C_i and thus to the calendar b_i of activity i. Therefore, we introduce a *time lag calendar* b_{ij} for each arc $(i, j) \in E$ of project network N. Point in time t is taken into account when computing the total working time between the starts of activities i and j exactly if $b_{ij}(t) = 1$. That is, $\int_{S_i}^{S_j} b_{ij}(\tau)d\tau$ equals the total working time in interval $[S_i, S_j[$ if $S_i \leq S_j$ and equals the negative total working time in interval $[S_j, S_i[$, otherwise.

The actual minimum difference Δ_{ij} between start times S_i and S_j that is prescribed by arc $(i, j) \in E$ depends on start time S_i and calendar b_{ij}:

$$\Delta_{ij}(S_i) = \min\{t \geq 0 \mid \int_{S_i}^t b_{ij}(\tau)\, d\tau \geq \delta_{ij}\} - S_i \quad ((i, j) \in E)$$

$S_i + \Delta_{ij}(S_i)$ is the earliest point in time $t \geq 0$ for which the total working time in interval $[S_i, t[$ or $[t, S_i[$, respectively, is greater than or equal to $|\delta_{ij}|$.

Since $b_{ij}(t) \in \{0, 1\}$ for all $t \geq 0$, it holds that $|\Delta_{ij}(S_i)| \geq |\delta_{ij}|$, and $\Delta_{ij}(S_i)$ and δ_{ij} have the same sign.

For temporal scheduling, the temporal constraints $S_j - S_i \geq \delta_{ij}$ for all $(i, j) \in E$ have to be replaced by

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

which due to $b_{ij}(t) \geq 0$ for all $t \in \mathbb{R}$ can also be written as

$$\int_{S_i}^{S_j} b_{ij}(\tau) d\tau \geq \delta_{ij} \quad ((i, j) \in E) \tag{5.2}$$

The interpretation of inequality (5.2) is as follows. If $\delta_{ij} \geq 0$, then the total working time $\int_{S_i}^{S_j} b_{ij}(\tau) d\tau$ between the starts of activity i at time S_i and the start of activity j at time S_j must be at least δ_{ij}. If $\delta_{ij} < 0$, then $\int_{S_i}^{S_j} b_{ij}(\tau) d\tau \geq \delta_{ij}$ means that the total working time $\int_{S_j}^{S_i} b_{ij}(\tau) d\tau = -\int_{S_i}^{S_j} b_{ij}(\tau) d\tau$ between S_j and S_i must not exceed $-\delta_{ij}$. Notice that for minimum time lags $d_{ij}^{min} = \delta_{ij} \geq 0$, constraint (5.2) is at least as tight as the ordinary temporal constraint $S_j - S_i \geq \delta_{ij}$, whereas maximum time lags $d_{ji}^{max} = -\delta_{ij} > 0$ are relaxed by considering breaks. Given start time S_i for activity i, the minimum start time S_j of activity j satisfying (5.2) is

$$t^* := \min\{t \geq 0 \mid \int_{S_i}^{t} b_{ij}(\tau)\, d\tau \geq \delta_{ij}\}$$

Constraints (5.1) and (5.2) are referred to as *calendar constraints*. A schedule S satisfying the calendar constraints is called *calendar-feasible*.

We now explain how to integrate the calendar constraints into the computation of earliest schedule ES by modifying the label-correcting method given by Algorithm 1.1. Algorithm 3.2 for the minimization of regular objective functions subject to temporal and disjunctive precedence constraints can be adapted similarly. The problem of finding the earliest calendar-feasible schedule ES can be formulated as follows:

$$\left.\begin{array}{ll} \text{Minimize} & \sum\limits_{i \in V} S_i \\ \text{subject to} & (5.1) \text{ and } (5.2) \\ & S_i \geq 0 \quad (i \in V) \end{array}\right\} \tag{5.3}$$

We start the label-correcting algorithm with $ES = (0, -\infty, \ldots, -\infty)$ and successively delay activities until all calendar constraints are satisfied. At the beginning, queue Q only contains the project beginning event 0. At each iteration, we dequeue an activity $i \in V$ from Q. If i is a real activity, we check whether start time ES_i complies with calendar b_i by computing the earliest point in time $t^* \geq ES_i$ for which there is no break in interval $[t^*, t^* + e_i[$ (cf. constraints (5.1)). In case of $ES_i < t^*$, the start of activity i must be delayed until time t^*. Next, we check inequalities (5.2) for all arcs $(i, j) \in E$ with initial node i. To this end, we compute the earliest start time

$t^* := \min\{t \geq ES_j \mid \int_{ES_i}^{t} b_{ij}(\tau)d\tau \geq \delta_{ij}\}$ of activity j given start time ES_i for activity i. If $ES_j < t^*$, schedule ES does not satisfy the corresponding prescribed time lag, and thus we increase ES_j up to t^*. In that case or if $b_j(\tau) = 0$ for some $t^* \leq \tau < t^* + e_j$, we enqueue j to Q if $j \notin Q$. Algorithm 5.1 summarizes this procedure.

Algorithm 5.1. Earliest calendar-feasible schedule

Input: MPM project network $N = (V, E, \delta)$, partition $\{V_{bi}^a, V_{ni}^a\}$ of set V^a, activity calendars b_i for $i \in V^a$, time lag calendars b_{ij} for $(i, j) \in E$.
Output: Earliest schedule ES.

```
set $ES_0 := 0$, $Q := \{0\}$, and $ES_i := -\infty$ for all $i \in V \setminus Q$;
while $Q \neq \emptyset$ do
   dequeue $i$ from $Q$;
   if $i \in V^a$ then
      determine $t^* := \min\{t \geq ES_i \mid b_i(\tau) = 1$ for all $t \leq \tau < t + e_i\}$;
      if $t^* > \overline{d}$ then terminate; (* there is no time-feasible schedule *)
      else if $ES_i < t^*$ then set $ES_i := t^*$;
   for all $(i, j) \in E$ do
      determine $t^* := \min\{t \geq ES_j \mid \int_{ES_i}^{t} b_{ij}(\tau)d\tau \geq \delta_{ij}\}$;
      if $ES_j < t^*$ then
         set $ES_j := t^*$;
         if $j \notin Q$ then enqueue $j$ to $Q$;
      if $j \in V^a \setminus Q$ and $b_j(\tau) = 0$ for some $t^* \leq \tau < t^* + e_j$ then enqueue $j$ to $Q$;
return earliest schedule $ES$;
```

Let β denote the number of breaks in all activity and time lag calendars. If some activity i is inspected more than $n(\beta+1)$ times, then there is no schedule satisfying the calendar constraints, and the algorithm can be stopped. Franck et al. (2001*a*) have shown that if the calendars are given as sorted lists of start and end times of breaks, Algorithm 5.1 can be implemented to run in $\mathcal{O}(mn\beta)$ time.

The latest schedule LS can be computed by using a similar label-correcting procedure again starting at node 0 and proceeding from terminal nodes j to initial nodes i of arcs $(i, j) \in E$. In difference to Algorithm 5.1, t^* is set to be the *lastest* time for which condition (5.1) or (5.2), respectively, is fulfilled (for details we refer to Neumann et al. 2003*b*, Sect. 2.11).

The enumeration scheme for resource allocation problems with regular objective functions (see Algorithm 3.1) can be used without almost any modification for the case of calendar constraints as well. Schedule $S = \min(\cup_{\rho \in P} \mathcal{S}_T(\rho))$ is then computed by the adaptation of Algorithm 3.2 to the case of break calendars, where the calendars b_{ij} for pairs $(i, j) \in \rho$ coincide with calendars b_i if $i \in V^a$ and are given by $b_{ij}(t) = 1$ for all $0 \leq t \leq \overline{d}$ if $i \in V^e$. $\mathcal{S}_T(\rho)$ now denotes the set of all schedules satisfying the calendar constraints (5.1) and (5.2) for relation network $N(\rho)$. Similarly to the case without calendars, it can be shown that set $S_T(\rho)$, though generally being disconnected, possesses a unique minimal point (cf. Franck 1999, Sect. 3.2) and that this property still carries over to the union $\cup_{\rho \in P} S_T(\rho)$ of sets $S_T(\rho)$.

5.2 Sequence-Dependent Changeover Times

This section is concerned with sequence-dependent changeover times arising when several (sub-)projects using common renewable resources are performed simultaneously at different sites (*multi-site scheduling*, see e.g., Sauer et al. 1998). When a unit of resource $k \in \mathcal{R}^\rho$ passes from the execution of an activity i at a location a to an activity j to be carried out at a different location b, the unit has to be torn down after the completion of i, transported from a to b, and put into service for processing j. Thus, the changeover time of resource k between the execution of activities i and j generally depends on resource k and on both activities i and j.

There is an extensive literature dealing with sequence-dependent changeovers in shop-floor environments, where changeover times are caused by replacing tools or cleaning. The great majority of the papers considers the problem of minimizing the total cost associated with changeovers (for a literature review we refer to Aldowaisan et al. 1999). Brucker and Thiele (1996) have devised a branch-and-bound algorithm for a general-shop problem where the makespan is to be minimized subject to precedence constraints and sequence-dependent changeover times between operations. Kolisch (1995), Ch. 8, has shown how to model changeover times between activities of a project by introducing alternative execution modes for the activities (see Section 5.3). The changeover times between two activities are assumed to be equal to a sequence-independent setup time or equal to zero. Moreover, the capacity R_k of each resource $k \in \mathcal{R}^\rho$ equals one. Trautmann (2001*a*), Sect. 3.3, has devised a branch-and-bound algorithm for minimizing the project duration in case of arbitrary resource capacities R_k, single-unit resource requirements $r_{ik} \in \{0, 1\}$, and general sequence-dependent changeover times.

In the sequel, we drop the assumption of single-unit resource requirements and consider any regular or convexifiable objective function f. Let $V_k^a := \{i \in V^a \mid r_{ik} > 0\}$ be the set of all activities using resource $k \in \mathcal{R}^\rho$. With $\vartheta_{ij}^k \in \mathbb{Z}_{\geq 0}$ we denote the changeover time from activity an $i \in V_k^a$ to an activity $j \in V_k^a$ on resource k, where $\vartheta_{ii}^k = 0$ for all $i \in V_k^a$. We suppose that the *weak triangle inequality*

$$\vartheta_{hi}^k + p_i + \vartheta_{ij}^k \geq \vartheta_{hj}^k$$

is satisfied for all $k \in \mathcal{R}^\rho$ and all $h, i, j \in V_k^a$. This assumption is generally met in practice because otherwise it would be possible to save changeover time by processing additional activities. For notational convenience we additionally assume that there are neither changeovers from the project beginning event 0 to activities $i \in V^a$ (setups) nor changeovers from activities $i \in V^a$ to the project termination event $n+1$ (teardowns). The latter condition can always be fulfilled by introducing the minimum time lags $d_{0i}^{min} = \max_{k \in \mathcal{R}^\rho : i \in V_k^a} \vartheta_{0i}^k$ and $d_{i,n+1}^{min} = \max_{k \in \mathcal{R}^\rho : i \in V_k^a} \vartheta_{i,n+1}^k$ and then putting $\vartheta_{0i}^k := \vartheta_{i,n+1}^k = 0$ for all $k \in \mathcal{R}^\rho$ and all $i \in V_k^a$.

The resource-constrained project scheduling problem (P) with sequence-dependent changeover times can be formulated as follows. We strive at minimizing objective function f such that all temporal and cumulative-resource constraints are observed and at any point in time, the demands for renewable resources by activities and changeovers do not exceed the respective resource capacities. More precisely, let for given resource $k \in \mathcal{R}^\rho$, $X_k : V_k^a \to \mathbb{P}(\mathbb{N})$ be a mapping providing for each activity $i \in V_k^a$ the set of units of resource k processing activity i, i.e.,

$$|X_k(i)| = r_{ik} \quad (i \in V^a) \tag{5.4}$$

We call a schedule S *changeover-feasible* if for each resource $k \in \mathcal{R}^\rho$, mapping X_k can be chosen such that

$$\left.\begin{array}{r} S_j \geq S_i + p_i + \vartheta_{ij}^k \\ \text{or } S_i \geq S_j + p_j + \vartheta_{ji}^k \end{array}\right\} \quad (i, j \in V_k^a : i \neq j,\ X_k(i) \cap X_k(j) \neq \emptyset) \tag{5.5}$$

and

$$X_k(i) \subseteq \{1, \ldots, R_k\} \quad (i \in V_k^a) \tag{5.6}$$

(5.5) says that if there is a unit of resource k processing both activities i and j, then activities i and j (including the possible changeover in between) must not overlap. (5.6) limits the availability of resource k to R_k units. Since all changeover times are nonnegative, a changeover-feasible schedule always observes the renewable-resource constraints (1.7).

In the following, we develop an equivalent characterization of the changeover-feasibility of schedules, which will serve as a basis for the solution method discussed later on and which draws from a model used by Nägler and Schönherr (1989) for solving time-resource and time-cost tradeoff problems. The underlying concepts go back to a model for aircraft scheduling presented in Lawler (1976), Sect. 4.9. A similar tanker scheduling problem has already been studied in an early paper by Dantzig and Fulkerson (1954). Let S be some schedule and let $k \in \mathcal{R}^\rho$ be a renewable resource. The analogue to schedule-induced strict order $\theta(S)$ introduced in Subsection 2.1.1 is the relation

$$\theta^k(S) := \{(i, j) \in V_k^a \times V_k^a \mid S_j \geq S_i + p_i + \vartheta_{ij}^k\}$$

Owing to the weak triangle inequality and because $p_i > 0$ for all $i \in V^a$, relation $\theta^k(S)$ is transitive and asymmetric and thus represents a strict order in set V_k^a. In contrast to the case without changeover times, however, $\theta^k(S)$ does not represent an interval order in general. We illustrate the latter statement by an example.

Example 5.1. Consider the schedule S depicted in Figure 5.1a and assume that the changeover times are $\vartheta_{12} = \vartheta_{34} = 0$ and $\vartheta_{14} = \vartheta_{32} = 1$. The strict order induced by schedule S is $\theta(S) = \{(1,2),(3,4)\}$, whose precedence graph $G(\theta(S)) = 2\mathbf{P}_2$ is shown in Figure 5.1b. Since a strict order θ is an interval order if and only if its precedence graph does not contain the parallel composition $2\mathbf{P}_2$ of two arcs as induced subgraph (see, e.g., Möhring 1984 or Trotter 1992, Sect. 3.8), $\theta(S)$ is not an interval order.

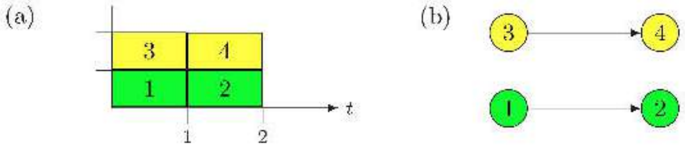

Fig. 5.1. Schedule-induced strict orders are no longer interval orders: (**a**) Gantt chart for schedule S; (**b**) precedence graph $G(\theta(S))$

Let for given schedule S and resource $k \in \mathcal{R}^\rho$, X_k be a mapping satisfying conditions (5.4) and (5.5) and let $r_k(S) := |\cup_{i \in V_k^a} X_k(i)|$ denote the number of resource units used. Clearly, S is changeover-feasible exactly if $r_k(S) \le R_k$ for all $k \in \mathcal{R}^\rho$. We consider an antichain U in schedule-induced strict order $\theta^k(S)$. It follows from the definition of $\theta^k(S)$ that $[S_i, S_i + p_i + \vartheta_{ij}^k[\cap [S_j, S_j + p_j + \vartheta_{ji}^k[\ne \emptyset$ for any two activities $i, j \in U$. (5.5) then implies that $X_k(i) \cap X_k(j) = \emptyset$ for any $i, j \in U$. This means that $|\cup_{i \in U} X_k(i)| = \sum_{i \in U} |X_k(i)| = \sum_{i \in U} r_{ik}$. On the other hand, it is obvious that for any subset $U' \subseteq V_k^a$, the number $|\cup_{i \in U'} X_k(i)|$ of resource units occupied by activities from U' is less than or equal to the joint requirements $\sum_{i \in U'} r_{ik}$ for resource k. Consequently, $r_k(S)$ equals the weight $\sum_{i \in U_k} r_{ik}$ of a maximum-weight antichain U_k in $\theta^k(S)$. Since all activities from set U_k pairwise overlap in time, U_k can be regarded as an *active set* $\mathcal{A}_k(S)$ for S. Schedule S is changeover-feasible precisely if none of the active sets $\mathcal{A}_k(S)$ with $k \in \mathcal{R}^\rho$ is forbidden.

Now recall that such a maximum-weight antichain U_k is a maximum-weight stable set in the precedence graph $G(\theta^k(S))$ equipped with node weights r_{ik} ($i \in V_k^a$). Since $G(\theta^k(S))$ is transitive, stable set U_k can be determined in $\mathcal{O}(n^3)$ time by computing a minimum (s,t)-flow u^k of value $\phi^k(u^k) = r_k(S)$ in the flow network $\overline{G}_k(\theta^k(S))$ with node set $V_k^a \cup \{s,t\}$ and arc set $\theta^k(S) \cup (\{s\} \times V_k^a) \cup (V_k^a \times \{t\})$, where nodes $i \in V_k^a$ are associated with lower capacities r_{ik} (cf. Subsection 2.1.1). Example 5.1 shows that strict order $\theta^k(S)$ generally does not represent an interval order, for which a maximum-weight stable set in the precedence graph can be found in linear time by computing a maximum-weight clique in the associated interval graph, cf. Golumbic (2004), Sects. 4.7 and 8.2.

The lower node capacities r_{ik} can be transformed into equivalent arc capacities by splitting up every node $i \in V_k^a$ into two nodes i' and i'' linked by arc (i', i'') with lower capacity $l_{i'i''} = r_{ik}$ and infinite upper capacity. The network flow methods then do not only provide a minimum (s,t)-flow u^k in $\overline{G}_k(\theta^k(S))$ but also a maximum (s,t)-cut $[U'_k, U''_k]$, whose capacity equals the minimum flow value $\phi(u^k)$ (see, e.g., Ahuja et al. 1993, Sect. 6.5). In addition, it can easily be shown that any maximum (s,t)-cut in $\overline{G}_k(\theta^k(S))$ is a uniformly directed cut containing only forward arcs. Thus, $\mathcal{A}_k(S) = \{i \in V_k^a \mid (i', i'') \in [U'_k, U''_k]\}$. As has already been noticed by Möhring (1985), Sect. 1.5, the computation of a maximum (s,t)-cut may also be performed in the transitive reduction of $\overline{G}_k(\theta^k(S))$ (i.e., in the network which arises from $\overline{G}_k(\theta^k(S))$ by replacing the arc set with its covering relation). In that case, any maximum (s,t)-cut $[U'_k, U''_k]$ contains only arcs (i', i'') obtained by splitting up some node $i \in V_k^a$, i.e., $\mathcal{A}_k(S) = \{i \in V_k^a \mid i' \in U'_k\}$.

To adapt the enumeration schemes for regular and convexifiable objective functions from Algorithms 3.1 and 3.3, respectively, to the occurrence of sequence-dependent changeover times, we make the following modifications. First, we replace the active sets $\mathcal{A}(S,t)$ at times t by active sets $\mathcal{A}_k(S)$. If for some $k \in \mathcal{R}^\rho$, $\mathcal{A}_k(S)$ is a forbidden set, we compute the set $\mathcal{B}$ of all minimal delaying alternatives B for $F = \mathcal{A}_k(S)$. In case of a regular objective function f, for given $B \in \mathcal{B}$ we then introduce the disjunctive precedence constraint

$$\min_{j \in B} S_j \geq \min_{i \in A}(S_i + p_i + \vartheta_{ij}^k)$$

between sets $A = F \backslash B$ and B including the changeover times ϑ_{ij}^k on k between $i \in A$ and $j \in B$. If f is convexifiable and $\{i\} \times B$ is some minimal delaying mode with $B \in \mathcal{B}$ and $i \in A = F \setminus B$, we add ordinary precedence constraints

$$S_j \geq S_i + p_i + \vartheta_{ij}^k \quad (j \in B)$$

between activity i and all activities $j \in B$, again including the changeover times ϑ_{ij}^k.

5.3 Alternative Execution Modes for Activities

In practice an activity can often be carried out in one out of finitely many alternative execution modes with different processing times, time lags, and resource requirements. The multiple modes give rise to several types of tradeoffs permitting a more efficient use of resources. Sometimes the tradeoffs include the consumption of *nonrenewable resources* like the project budget. As for renewable resources, the availability of nonrenewable resources is limited. The availability of nonrenewable resources, however, does not refer to individual points in time but to the entire planning period. Each time an activity is carried out, the residual availability of a nonrenewable resource is decreased by the corresponding resource demand. Thus, nonrenewable resources can be viewed as special cumulative resources (cf. Section 1.3) that are depleted but never replenished. This implies that for nonrenewable resources, resource-feasibility solely depends on the selection of activity modes and not on the schedule. That is the reason why nonrenewable resources can be omitted when dealing with single-mode project scheduling problems.

Since the early 1980s, the (discrete) multi-mode project duration problem with precedence constraints among the activities instead of general temporal constraints has been treated by several authors. The case of resource-resource tradeoffs has already been considered by Elmaghraby (1977), Sect. 3.4.2. Exact algorithms have been reviewed and their performance has been tested by Hartmann and Drexl (1998). At present, the most efficient method for solving this problem is the branch-and-bound algorithm of Sprecher and Drexl (1998). Hartmann (1999*b*), Sect. 7.3, has compared several heuristic approaches. An experimental performance analysis presented in the latter reference reveals that among the tested heuristics, the best procedure is a genetic algorithm published in Hartmann (2001). A special case of the multi-mode project duration problem has been studied by Demeulemeester et al. (2000), who have developed a branch-and-bound algorithm for the discrete time-resource tradeoff problem. For each real activity, a workload for a single renewable resource is specified. The alternative execution modes arise from all undominated integral duration-requirement combinations the product of which is at least equal to the given workload.

For the case of general temporal constraints, four different algorithms have been proposed in literature. The tabu search procedure by De Reyck and Herroelen (1999) performs a local search in the set of possible mode assignments to activities. For given execution modes, the resulting single-mode problem is then solved by the branch-and-bound algorithm of De Reyck and Herroelen (1998*a*). Franck (1999), Sect. 7.2, has adapted a priority-rule method by Kolisch (1995), Sect. 6.2, to the case of general temporal constraints. At each iteration, the activity to be scheduled is chosen on the basis of a first priority rule. A second priority rule provides the execution mode for the selected activity. A streamlined multi-pass version of this procedure can be found in Heilmann (2001). Dorndorf (2002), Ch. 6, has described an extension of the branch-and-bound algorithm by Dorndorf et al. (2000*c*) for the single-mode project duration problem (cf. Subsection 3.1.4) to the multi-mode case, where mode assignment and activity scheduling are iterated alternately. Brucker and Knust (2003) have presented an adaptation of their lower bound for the single-mode problem (see Subsection 3.1.3) to the presence of multiple execution modes. The corresponding linear program is again solved by column-generation techniques.

In this section we discuss the enumeration scheme of a branch-and-bound procedure proposed by Heilmann (2003) for the multi-mode project duration problem, where the selection of activity modes and the allocation of resources are performed in parallel. The basic principle of this relaxation-based enumeration scheme can be used for solving multi-mode resource-constrained project scheduling problems with arbitrary regular or convexifiable objective functions. Roughly speaking, the idea is to consider single-mode problems arising from *mode relaxations* where only the unavoidable resource requirements, core durations, and core time lags occurring in all selectable execution modes are taken into account. The mode relaxations are stepwise refined by assigning execution modes to activities and thus reducing the sets of selectable modes. For what follows, we assume that only the requirements for renewable and nonrenewable resources depend on the mode selection. The case where execution modes also differ in requirements for cumulative resources can be treated similarly (see Trautmann 2001*a*, Sect. 3.1).

A discrete multi-mode resource allocation problem decomposes into two subproblems: the discrete *mode assignment problem* and the (single-mode) *resource allocation problem*. Let $\mathcal{M}_i$ denote the set of alternative execution modes for activity $i \in V$, where $|\mathcal{M}_i| = 1$ if $i \in V^e$. We call a binary vector $\underline{x} = (\underline{x}_{im_i})_{i\in V, m_i\in\mathcal{M}_i}$ with $\sum_{m_i\in\mathcal{M}_i} \underline{x}_{im_i} \leq 1$ a (partial) assignment of modes $m_i \in \mathcal{M}_i$ to activities $i \in V$ (an *assignment*, for short), where $\underline{x}_{im_i} = 1$ if activity i is carried out in mode m_i and $x_{im_i} = 0$, otherwise. An assignment $\underline{x}' > \underline{x}$ is called an extension of $\underline{x}$. An assignment x satisfying the *mode assignment constraints*

$$\sum_{m_i\in\mathcal{M}_i} x_{im_i} = 1 \quad (i \in V) \tag{5.7}$$

is termed a *full assignment*. Solving the mode assignment problem consists in finding a full assignment x such that x complies with the temporal and nonrenewable-resource constraints. Each assignment $\underline{x}$ defines a corresponding single-mode resource allocation problem.

Now let

$$\mathcal{M}_i(\underline{x}) := \begin{cases} \mathcal{M}_i, \text{ if } \sum_{m_i\in\mathcal{M}_i} \underline{x}_{im_i} = 0 \\ \{m_i\} \text{ with } \underline{x}_{im_i} = 1, \text{ otherwise} \end{cases}$$

be the set of modes that can be selected for activity i in full-assignment extensions $x \geq \underline{x}$ and let $\mathcal{R}^\nu$ be the set of nonrenewable resources with availabilities $R_k \in \mathbb{N}$. By $r_{ikm_i} \in \mathbb{Z}_{\geq 0}$ we denote the requirement for resource $k \in \mathcal{R}^\rho \cup \mathcal{R}^\nu$ if real activity $i \in V^a$ is executed in mode $m_i \in \mathcal{M}_i$. Then

$$r_{ik}(\underline{x}) := \min_{m_i\in\mathcal{M}_i(\underline{x})} r_{ikm_i}$$

is the (unavoidable) requirement of activity $i \in V^a$ for resource $k \in \mathcal{R}^\rho \cup \mathcal{R}^\nu$ given assignment $\underline{x}$. Assignment $\underline{x}$ is called resource-feasible if $\underline{x}$ satisfies the nonrenewable-resource constraints

$$\sum_{i\in V^a} r_{ik}(\underline{x}) \leq R_k \quad (k \in \mathcal{R}^\nu) \tag{5.8}$$

Alternative assignments $\underline{x}$ are associated with different single-mode project networks $N(\underline{x})$. Without loss of generality, we assume that the node set V and the arc set E of $N(\underline{x})$ are the same for all assignments $\underline{x}$. For each arc

$(i, j) \in E$, the associated time lag may depend on the execution modes of both activities i and j. Hence, the weight of an arc $(i, j) \in E$ in the multi-mode project network N is a matrix $\delta_{ij} = (\delta_{im_ijm_j})_{m_i \in \mathcal{M}_i, m_j \in \mathcal{M}_j}$, where the elements $\delta_{im_ijm_j} \in \mathbb{Z}$ denote the scalar arc weights that refer to the execution of activities i and j in modes $m_i \in \mathcal{M}_i$ and $m_j \in \mathcal{M}_j$. For assignment $\underline{x}$,

$$\delta_{ij}(\underline{x}) := \min_{m_i \in \mathcal{M}_i(\underline{x})} \min_{m_j \in \mathcal{M}_j(\underline{x})} \delta_{im_ijm_j}$$

is the resulting (core) weight of arc (i, j) in network $N(\underline{x})$. An assignment $\underline{x}$ is called time-feasible if $N(\underline{x})$ does not contain any cycle of positive length. A time- and resource-feasible assignment is referred to as a feasible assignment. A schedule S is said to be time-feasible with respect to assignment $\underline{x}$ if S satisfies the temporal constraints

$$S_j - S_i \geq \delta_{ij}(\underline{x}) \quad ((i, j) \in E) \tag{5.9}$$

The set of schedules which are time-feasible with respect to assignment $\underline{x}$ are denoted by $\mathcal{S}_T(\underline{x})$

Define $p_{im_i} \in \mathbb{N}$ to be the processing time if real activity $i \in V^a$ is executed in mode $m_i \in \mathcal{M}_i$. The (core) duration of activity $i \in V$ given assignment $\underline{x}$ is

$$p_i(\underline{x}) := \min_{m_i \in \mathcal{M}_i(\underline{x})} p_{im_i}$$

For schedule S, the set of real activities being in progress at time t then equals $\mathcal{A}(S, \underline{x}, t) := \{i \in V^a \mid S_i \leq t < S_i + p_i(\underline{x})\}$ and $r_k(S, \underline{x}, t) := \sum_{i \in \mathcal{A}(S,\underline{x},t)} r_{ik}(\underline{x})$ is the demand for resource $k \in \mathcal{R}^\rho$ at time t. A schedule S which satisfies the renewable-resource constraints

$$r_k(S, \underline{x}, t) \leq R_k \quad (k \in \mathcal{R}^\rho, \ 0 \leq t \leq \bar{d}) \tag{5.10}$$

as well as the cumulative-resource constraints (1.20) is called resource-feasible with respect to assignment $\underline{x}$. By $\mathcal{S}_R(\underline{x})$ we denote the set of all schedules satisfying (5.10). Recall that the resource-feasibility of an assignment $\underline{x}$ requires that the nonrenewable-resource constraints (5.8) are fulfilled. A schedule that is time- and resource-feasible with respect to assignment $\underline{x}$ is termed feasible with respect to $\underline{x}$. $\mathcal{S}(\underline{x}) = \mathcal{S}_T(\underline{x}) \cap \mathcal{S}_R(\underline{x}) \cap \mathcal{S}_C$ is the set of all feasible schedules with respect to $\underline{x}$. The multi-mode resource-constrained project scheduling problem can now be stated as follows:

$$\left.\begin{array}{lll} \text{Minimize} & f(S) & \\ \text{subject to} & \displaystyle\sum_{m_i \in \mathcal{M}_i} x_{im_i} = 1 & (i \in V) \\ & x_{im_i} \in \{0, 1\} & (i \in V, \ m_i \in \mathcal{M}_i) \\ & S \in \mathcal{S}_T(x) \cap \mathcal{S}_R(x) \cap \mathcal{S}_C & \end{array}\right\} \text{(MP)}$$

A feasible solution to problem (MP) consists in a schedule-assignment pair (S, x), where x is a feasible full assignment (i.e., a solution to the mode-assignment problem) and S is a feasible schedule with respect to x (i.e., a feasible solution to the respective single-mode project scheduling problem). An optimal solution is a feasible solution (S, x) with minimum objective function value $f(S)$.

From Theorem 1.12 it immediately follows that finding a feasible solution (S, x) is NP-hard. In addition, Kolisch (1995), Sect. 2.3, and Schwindt (1998*b*) have shown by transformations from KNAPSACK and PRECEDENCE-CONSTRAINED KNAPSACK, respectively, that the problems of testing whether there is a resource-feasible or a time-feasible full mode assignment x are already NP-complete. Consequently, the resource relaxation of a multi-mode resource allocation problem is NP-hard. Hence, to obtain a problem that can be solved efficiently, the mode assignment constraints (5.7) have to be relaxed as well. The *mode relaxation* for an assignment $\underline{x}$ then reads

$$\left.\begin{array}{ll} \text{Minimize} & f(S) \\ \text{subject to} & \mathcal{S}_T(\underline{x}) \cap \mathcal{S}_R(\underline{x}) \cap \mathcal{S}_C \end{array}\right\} \quad (\mathrm{P}(\underline{x}))$$

Obviously, the single-mode resource-constrained project scheduling problem $(\mathrm{P}(\underline{x}))$ is a relaxation of all mode relaxations $(\mathrm{P}(\underline{x}'))$ belonging to extensions $\underline{x}'$ of $\underline{x}$, i.e.,

$$\mathcal{S}(\underline{x}') \subseteq \mathcal{S}(\underline{x}) \quad (\underline{x}' \geq \underline{x})$$

This observation is the starting point for a relaxation-based enumeration scheme for solving multi-mode problem (MP). Let ρ be some relation in node set V, and let $\mathcal{S}_T(\rho, \underline{x}) := \{S \in \mathcal{S}_T(\underline{x}) \mid S_j \geq S_i + p_i(\underline{x}) \text{ for all } (i,j) \in \rho\}$ be the relation polytope belonging to ρ and assignment $\underline{x}$. The algorithm starts with the empty assignment $\underline{x} = 0$. For the corresponding single-mode problem $(\mathrm{P}(\underline{x}))$, schedules are enumerated as minimal points of appropriate (unions of) relation polytopes $\mathcal{S}_T(\rho, \underline{x})$, see Algorithms 3.1 and 3.3. Each time a schedule S feasible with respect to $\underline{x}$ has been obtained, the execution mode of some activity i with $\sum_{m_i \in \mathcal{M}_i} \underline{x}_{im_i} = 0$ is fixed such that the resulting assignment $\underline{x}'$ is still feasible (if there is no mode $m_i \in \mathcal{M}_i$ such that $\underline{x}'$ is feasible, we perform backtracking). Then, the time-feasibility of S with respect to the new assignment $\underline{x}'$ is restored. Due to $\mathcal{S}(\underline{x}') \subseteq \mathcal{S}(\underline{x})$, S may be not resource-feasible with respect to $\underline{x}'$. In that case, the enumeration of schedules is resumed by extending the current relation ρ until a schedule S' which is feasible with respect to $\underline{x}'$ has been found. These steps are reiterated until a feasible full assignment x has been reached or there is no feasible extension of the current assignment $\underline{x}'$.

5.4 Continuous Cumulative Resources

In this section we deal with continuous cumulative resources whose inventory is depleted and replenished at constant rates by the activities of the project. This type of resources has been considered by Schwindt (2002) and Neumann et al. (2005) in the context of scheduling problems arising in the process industries. Recently, Sourd and Rogerie (2005) have presented constraint propagation techniques for computing lower and upper approximations to the loading profiles of continuous cumulative resources.

The concept of continuous cumulative resources also covers the renewable and (discrete) cumulative resources, which we have considered until now. For the case of convex objective functions f, we show how the expanded resource-constrained project scheduling problem can be solved by using a relaxation-based approach. The basic principle is again to substitute the resource constraints into a finite disjunction of linear inequalities, which can be viewed as parameterized precedence constraints between activities.

Let $\widetilde{\mathcal{R}}^\gamma$ be the set of continuous cumulative resources with safety stocks $\underline{R}_k \in \mathbb{Z} \cup \{-\infty\}$ and storage capacities $\overline{R}_k \in \mathbb{Z} \cup \{\infty\}$, where $\overline{R}_k \geq \underline{R}_k$. Performing an activity $i \in V$ increases the inventory in resource $k \in \widetilde{\mathcal{R}}^\gamma$ by $r_{ik} \in \mathbb{Z}$ units. Analogously to the case of discrete cumulative resources, we suppose that $\underline{R}_k \leq \sum_{i \in V} r_{ik} \leq \overline{R}_k$ for all $k \in \widetilde{\mathcal{R}}^\gamma$, which ensures that the terminal inventories are within the prescribed bounds. If $r_{ik} < 0$, we again speak of a depletion of resource k, and if $r_{ik} > 0$, we say that resource k is replenished. Depletion and replenishments arise at constant rates $\hat{r}_{ik} = r_{ik}/p_i$. This means that events $i \in V^e$ deplete and replenish at infinite rates, which corresponds to the setting for discrete cumulative resources. Since renewable-resource constraints can be expressed by temporal and discrete cumulative-resource constraints, the new model also includes both types of resource constraints that have been studied previously. By V_k^- and V_k^+ we respectively denote the sets of activities depleting or replenishing resource k. $V_k := V_k^- \cup V_k^+$ is the set of all depleting and replenishing activities for resource k. The resource constraints again say that at any point in time, the inventory level of each resource must be between the safety stock and the storage capacity.

Now let S be some schedule. By

$$x_i(S,t) := \begin{cases} 0, \text{ if } t < S_i \\ 1, \text{ if } t \geq S_i + p_i \\ (t - S_i)/p_i, \text{ otherwise} \end{cases}$$

we denote the portion of activity $i \in V$ that has been processed by time t. If $i \in V^e$, then $x_i(S,t) = 0$ if $S_i < t$, and $x_i(S,t) = 1$, otherwise. The inventory in resource $k \in \widetilde{\mathcal{R}}^\gamma$ at time t is

$$\widetilde{r}_k(S,t) := \sum_{i \in V} r_{ik} x_i(S,t)$$

The corresponding loading profile $\widetilde{r}_k(S,\cdot)$ is a right-continuous, piecewise affine function. The resource constraints can be stated as

$$\underline{R}_k \leq \widetilde{r}_k(S,t) \leq \overline{R}_k \quad (k \in \widetilde{\mathcal{R}}^\gamma,\ 0 \leq t \leq \overline{d}) \tag{5.11}$$

A schedule satisfying resource constraints (5.11) is called resource-feasible. Let $\widetilde{\mathcal{S}}_C$ denote the set of resource-feasible schedules. The set of all feasible schedules is $\mathcal{S} = \mathcal{S}_T \cap \widetilde{\mathcal{S}}_C$. The resource-constrained project scheduling problem to be dealt with reads as follows:

$$\left.\begin{array}{ll} \text{Minimize} & f(S) \\ \text{subject to} & S \in \mathcal{S}_T \cap \widetilde{\mathcal{S}}_C \end{array}\right\} \ (\widetilde{\mathrm{P}})$$

where f is some convex objective function. An optimal schedule is a schedule S solving problem $(\widetilde{\mathrm{P}})$.

Next, we explain the basic principle of the solution procedure. For simplicity of exposition we assume for the moment that $V_k \subseteq V^a$ for all $k \in \widetilde{\mathcal{R}}^\gamma$. Similarly to the relaxation-based algorithms from Chapter 3, we first delete the resource constraints and solve the resulting time-constrained project scheduling problem. Subsequently, resource conflicts are stepwise sorted out by refining the relaxation with new constraints. For notational convenience we suppose that all storage capacities are infinite. This can always be ensured by the following transformation (cf. Remark 1.21a). For each resource $k \in \widetilde{\mathcal{R}}^\gamma$, we set $\overline{R}_k := \infty$ and add a fictitious resource k' with requirements $r_{ik'} = -r_{ik}$ for all $i \in V_k$, safety stock $\underline{R}_{k'} = -\overline{R}_k$, and storage capacity $\overline{R}_{k'} = \infty$.

Let S be an optimal solution to the resource relaxation and assume that at time t, the inventory in some resource $k \in \widetilde{\mathcal{R}}^\gamma$ falls below the safety stock, i.e., $\widetilde{r}_k(S,t) < \underline{R}_k$. We partition V_k into two sets A and B with the following meaning. Set A contains all activities $j \in V_k^-$ to be completed by time t and all activities $j \in V_k^+$ to be started no earlier than at time t:

$$\left.\begin{array}{ll} S_j \le t - p_j & (j \in A \cap V_k^-) \\ S_j \ge t & (j \in A \cap V_k^+) \end{array}\right\} \tag{5.12}$$

The total depletion of the inventory in resource k at time t caused by activities $j \in A$ equals $-\sum_{j \in A \cap V_k^-} r_{jk}$. The activities j from set B must be scheduled in such a way that at time t, their net replenishment of resource k is greater than or equal to the shortfall $\underline{R}_k - \sum_{j \in A \cap V_k^-} r_{jk}$ caused by the activities from set A. This can be ensured as follows. For each activity $j \in B$, we introduce a continuous decision variable x_j with

$$0 \le x_j \le 1 \quad (j \in B) \tag{5.13}$$

providing the portion of activity j that will be processed by time t. The requirement that the inventory in resource k at time t must not fall below $\underline{R}_k$ then reads

$$\sum_{j \in B} r_{jk} x_j \ge \underline{R}_k - \sum_{j \in A \cap V_k^-} r_{jk} \tag{5.14}$$

The coupling between decision variables x_j and S_j is achieved by the temporal constraints (parameterized in x_j)

$$\left.\begin{array}{ll} S_j \geq t - p_j x_j & (j \in B \cap V_k^-) \\ S_j \leq t - p_j x_j & (j \in B \cap V_k^+) \end{array}\right\} \qquad (5.15)$$

Inequalities (5.15) ensure that for each schedule S satisfying (5.15) it holds that $x_j \geq x_j(S,t)$ if activity $j \in B$ depletes and $x_j \leq x_j(S,t)$ if activity $j \in B$ replenishes the stock of k. Adding constraints (5.12) to (5.15) to the relaxation removes the inventory shortage at time t.

The inventory in resource k attains its minimum at a point in time when some replenishing activity i is started or when some depleting activity i is completed. That is why time t can always be chosen to be equal to S_i for some $i \in V_k^+$ or equal to $S_i + p_i$ for some $i \in V_k^-$, and thus we can replace t in (5.12) and (5.15) by S_i or $S_i + p_i$. We then write A^{ik} and B^{ik} instead of A and B as well as x_j^{ik} instead of x_j. Note that without loss of generality we can assume $i \in A^{ik}$ for all $k \in \widetilde{\mathcal{R}}^\gamma$ and all $i \in V_k$ because the corresponding inequality (5.12) is always satisfied. Passing from constants t to variables S_i ensures that only a finite number of constraints have to be introduced before the resource constraints (5.11) are satisfied.

From the above reasoning it follows that $\widetilde{\mathcal{S}}_C$ again represents the union of finitely many polyhedra. The set of all minimal points of $\widetilde{\mathcal{S}}_C$, however, is generally uncountable, which implies that the set $\mathcal{AS}$ of all active schedules is infinite (and hence so are all of its supersets depicted in Figure 2.4).

The solution procedure is now as follows. We solve the convex program

$$\left.\begin{array}{ll} \text{Minimize} & f(S) \\ \text{subject to} & S \in \mathcal{S}_T \\ & \text{(5.12) to (5.15) for partitions } \{A^{ik}, B^{ik}\} \text{ selected} \end{array}\right\} \qquad (5.16)$$

and add new constraints of type (5.12) to (5.15) to problem (5.16) until either the search space $\mathcal{P}$ becomes void or the resulting schedule S is feasible. Then, we return to an alternative partition $\{A^{ik}, B^{ik}\}$ and proceed until all alternatives have been investigated. Convex program (5.16) can be solved in polynomial time because its feasible region $\mathcal{P}$ represents a polytope. Of course, the objective function value of any optimal solution to (5.16) represents a lower bound on the objective function value $f(S)$ of any feasible schedule S satisfying the added constraints of type (5.12) to (5.15).

Next we discuss some implementation issues. Assume that the inventory in some resource $k \in \widetilde{\mathcal{R}}^\gamma$ falls below the safety stock at time $t = S_i$ $(i \in V_k^+)$ or $t = S_i + p_i$ $(i \in V_k^-)$. To enumerate the sets A^{ik} and B^{ik} we construct a binary tree as follows. Each level of the tree belongs to one activity $j \in V_k$. For each activity j we branch over the alternatives $j \in A^{ik}$ and $j \in B^{ik}$ and add the corresponding constraints (5.12) or (5.13), (5.15), as well as for both alternatives the relaxation

$$\sum_{j \in B^{ik}} r_{jk} x_j^{ik} \geq \underline{R}_k - \sum_{j \in A^{ik} \cap V_k^-} r_{jk} - \sum_{j \in V_k^+ \setminus (A^{ik} \cup B^{ik})} r_{jk}$$

of constraint (5.14) to the convex program (5.16). Each leaf of the tree corresponds to one distinct partition $\{A^{ik}, B^{ik}\}$. We can stop the enumeration for activity i as soon as the inventory shortage at time S_i or $S_i + p_i$ is settled, even if $A^{ik} \cup B^{ik} \subset V_k$. In the latter case, it may be necessary to resume the branching later on if the shortage reappears while dealing with other resource conflicts. Since for each resource $k \in \widetilde{\mathcal{R}}^\gamma$ and each activity $i \in V_k$, the construction of the corresponding sets A^{ik} and B^{ik} requires at most $|V_k|$ steps, the height of the branch-and-bound tree is of order $\mathcal{O}(|\widetilde{\mathcal{R}}^\gamma|n^2)$.

The computational effort can be reduced considerably by testing whether the search space $\mathcal{P}$ has become void before solving convex program (5.16). Let $\widetilde{d}_{ij}$ be the minimum time lag between activities i and j that is implied by the prescribed temporal constraints, inequalities (5.12), and inequalities (5.15) where x_j^{ik} is set to be equal to 1 if $j \in V_k^-$ and equal to 0, otherwise. Assume that for some activity $j \in V_k$ the addition to set A^{ik} or B^{ik} leads to a new temporal constraint $S_j - S_i \geq \delta_{ij}$. Then $\mathcal{P} = \emptyset$ if $\delta_{ij} + \widetilde{d}_{ji} > 0$. In that case, the alternative set B^{ik} or A^{ik}, respectively, can immediately be selected for activity j.

Now let (S, x) be an optimal solution to (5.16) such that schedule S is feasible. We then obtain a feasible schedule S' with $f(S') \leq f(S)$ by

(a) moving all activities $j \in V_k$ from A^{ik} to B^{ik} for which (5.12) is active,
(b) moving all activities $j \in V_k^-$ from B^{ik} to A^{ik} for which $x_j^{ik} = 1$, and
(c) moving all activities $j \in V_k^+$ from B^{ik} to A^{ik} for which $x_j^{ik} = 0$

and solving convex program (5.16) again. Based on this dominance rule, feasible solutions belonging to leaves of the enumeration tree can be improved and thus the current upper bounds can be decreased by performing the above transformations (a) to (c).

Eventually, we consider the general case including discrete depletions and replenishments at the occurrence times of events. For events $j \in B \cap V^e$ decision variable x_j can be fixed to 0 if $j \in V_k^-$ and to 1 if $j \in V_k^+$ because $p_j = 0$ (compare (5.15)). If activity i with $t = S_i$ or $t = S_i + p_i$ is chosen to be an event, then i must deplete the stock of resource k. Moreover, for an event $j \in B \cap V_k^-$ it may happen that $S_j = t$ though $x_j = 0$, i.e., $x_j < x_j(S, t)$. As a consequence, the shortage at time t may persist after having introduced constraints (5.12) to (5.15), in which case we perform backtracking. If problem $(\widetilde{\mathrm{P}})$ is solvable, the enumeration tree contains alternative partitions removing the shortage.