OR Spektrum (2001) 23: 325–334

OR Spektrum
© Springer-Verlag 2001

# Project scheduling with calendars*

**Birger Franck, Klaus Neumann, and Christoph Schwindt**

Institut für Wirtschaftstheorie und Operations Research, University of Karlsruhe, 76128 Karlsruhe, Germany (e–mail: {franck,neumann,schwindt}@wior.uni-karlsruhe.de)

Received: July 26, 2000 / Accepted: May 15, 2001

**Abstract.** For many applications of project scheduling to real-life problems, it is necessary to take into account calendars specifying time intervals during which some resources such as manpower or machines are not available. Whereas the execution of certain activities like packaging may be suspended during breaks, other activities cannot be interrupted due to technical reasons. Minimum and maximum time lags between activities may depend on calendars, too. In this paper, we address the problem of scheduling the activities of a project subject to calendar constraints. We devise efficient algorithms for computing earliest and latest start and completion times of activities. Moreover, we sketch how to use these algorithms for developing priority-rule methods coping with renewable-resource constraints and calendars.

**Key words:** Project scheduling-calendars – Minimum and maximum time lags – Priority-rule methods

## 1 Introduction

Project scheduling is concerned with the assignment of execution time intervals to the activities of the project (for a detailed introduction to project scheduling, we refer to Neumann and Schwindt, 1997, or to Franck et al., 2001, appearing in this issue). Let $V = \{0, 1, \ldots, n, n+1\}$ be the set of all activities and let $p_i \in \mathbb{Z}_{\geq 0}$ be the duration of activity $i \in V$, where 0 and $n+1$ with $p_0 = p_{n+1} = 0$ denote dummy activities which represent the project beginning and the project termination, respectively. Between the start times $S_i, S_j \geq 0$ of two different activities $i, j \in V$

* The authors would like to acknowledge the helpful comments of two anonymous referees. This research has been supported in part by the Deutsche Forschungsgemeinschaft (Grant Ne 137/4).

*Correspondence to:* C. Schwindt

there may be a prescribed *minimum time lag* $d_{ij}^{min} \in \mathbb{Z}_{\geq 0}$ (a prescribed *maximum time lag* $d_{ij}^{max} \in \mathbb{Z}_{\geq 0}$) saying that activity $j$ can be started $d_{ij}^{min}$ units of time after the start of activity $i$ at the earliest (that activity $j$ must be started $d_{ij}^{max}$ units of time after the start of activity $i$ at the latest). Activities and time lags can be represented by an activity-on-node project network $N$ with node set $V$ and arc set $E$. For each minimum time lag $d_{ij}^{min}$, we introduce an arc $\langle i, j \rangle$ from node $i$ to node $j$ weighted by $\delta_{ij} = d_{ij}^{min}$. Maximum time lags $d_{ij}^{max}$ correspond to backward arcs $\langle j, i \rangle$ from node $j$ to node $i$ with weight $\delta_{ji} = -d_{ij}^{max}$. By $Pred(i)$ and $Succ(i)$ we denote the set of all (direct) predecessors and (direct) successors, respectively, of node $i$ in network $N$. The *temporal constraints* given by minimum and maximum time lags can be written as

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

When scheduling real-life projects, make-to-order production, or process flows in chemical industries (for the latter two applications of resource-constrained project scheduling, we refer to Neumann and Schwindt, 1997, and Neumann et al., 2001, respectively), we have to take into account breaks like weekends or holidays where manpower or machines are not available (cf. Schwindt and Trautmann, 2000). Scheduling the activities subject to such *break calendars* is termed *calendarization* (see Zhan, 1992). Some activities may be interrupted during a break, whereas others must not be interrupted due to technical reasons. Therefore, we have to distinguish between (break-)interruptible activities $i \in V^{bi} \subset V$ and non-interruptible activities $j \in V^{ni} = V \setminus V^{bi}$ where $i \in V^{ni}$ if $p_i = 0$. For each interruptible activity $i \in V^{bi}$, a *minimum execution time* $\varepsilon_i \in \mathbb{N}$ is prescribed during which $i$ has to be in progress without interruption (where the completion of $i$ is *not* regarded as interruption). In practice, $\varepsilon_i$ is generally chosen such that the processing time $p_i$ is an integral multiple of $\varepsilon_i$, e.g. $\varepsilon_i = 1$. For activities $i \in V^{ni}$, we set $\varepsilon_i := p_i$, that is, $i$ has to be processed during $p_i$ units of time without suspending the execution and thus cannot be interrupted at all.

A *break calendar* (or *calendar*, for short) is a function $b : \mathbb{R}_{\geq 0} \rightarrow \{0, 1\}$ with the following interpretation: $b(t) = 1$ indicates that time $t$ belongs to a working period (we then speak of $t$ as a *working time*) whereas $b(t) = 0$ means that $t$ falls into a break (in that case, $t$ is called a *spare time*). Without loss of generality, we assume that a calendar $b$ is a step function, i.e., $b$ is piecewise constant, and that $b$ is continuous from the right at its jump points. Thus, $b$ is integrable, and for $0 \leq \alpha < \beta$, $\int_\alpha^\beta b(\tau)\, d\tau$ is the *total working time* in interval $[\alpha, \beta[$. Figure 1 shows an example of a calendar function $b$ with the corresponding total working time $\int_0^t b(\tau)\, d\tau$ up to time $t$. $\int_0^t b(\tau)\, d\tau$ is a continuous and piecewise linear function in $t$ whose corner points coincide with the start and end of breaks in calendar $b$. The total working time $\int_4^{15} b(\tau)\, d\tau$ in interval $[4, 15[$ equals $\int_0^{15} b(\tau)\, d\tau - \int_0^4 b(\tau)\, d\tau = 10 - 3 = 7$, which corresponds to the (total) area of the shaded boxes.

The processing of the real activities $i = 1, \ldots, n$ of the project requires *renewable resources* such as manpower or machines. Let $\mathcal{R}$ denote the set of renewable resources. For the present, we suppose that the capacity of the renewable resources $k \in \mathcal{R}$ is not limited. This assumption will be dropped in Section 4, where we deal

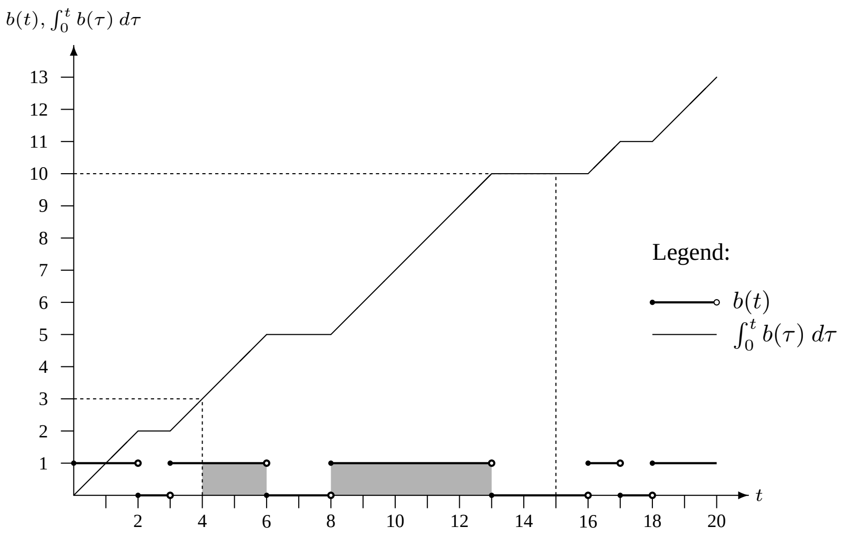

**Fig. 1.** Calendar $b$ and total working time $\int b(\tau) d\tau$

with the case of resource constraints arising from the scarcity of resources. In practice, different resources $k \in \mathcal{R}$ generally have different calendars. We obtain the corresponding *activity calendar* $b_i$ for activity $i \in V$ by setting $b_i(t)$ equal to zero if there is a resource $k \in \mathcal{R}$ used by $i$ which due to a break is not available at time $t$, and equal to one, otherwise. If $b_i(t) = 0$, we have to suspend the execution of activity $i \in V^{bi}$ being in progress right before point in time $t$. For what follows, we establish the convention that first an activity $i \in V^{bi}$ cannot be interrupted during working time, i.e., an interruption of $i$ at a time $t$ with $b_i(t) = 1$ is not allowed, and that second the execution of $i$ interrupted at some time $t$ has to be resumed exactly at the end of the current break, i.e. at time $t' := \min\{\tau > t \mid b_i(\tau) = 1\}$. This convention is generally accepted in practice, in particular, if the processing of activities requires the setup of certain resources such as machines or chemical reactors. Furthermore, we suppose that for activities $i \in V^{bi}$, the minimum length of a working time interval between two successive breaks in calendar $b_i$ is greater than or equal to minimum execution time $\varepsilon_i$. This ensures that once $i$ has been started at time $S_i$ such that $i$ is continuously in progress during $\varepsilon_i$ units of time, the working time between the completion and start of any two successive interruptions of $i$ is not less than $\varepsilon_i$.

## 2 Modelling

The requirement that no activity $i \in V$ can be interrupted before $i$ has been processed for (at least) $\varepsilon_i$ units of time, i.e., that there is no spare time in interval $[S_i, S_i + \varepsilon_i[$, can be stated as

$$b_i(\tau) = 1 \text{ for all } \tau \in [S_i, S_i + \varepsilon_i[ \quad (i \in V) \tag{2}$$

Let $B_i$ be the number of breaks in calendar $b_i$. Given some time $t \geq 0$, the earliest start time $S_i \geq t$ observing constraints (2) can then be found in $\mathcal{O}(B_i)$ time.

There is no one-to-one correspondence between start times $S_i$ and completion times $C_i$ of activities $i$ if the processing of activities may be suspended at arbitrary points in time (in machine scheduling, interruptions are then referred to as *preemption*, cf. Brucker, 2001). Nevertheless, in our case the assumption that activities can only be interrupted at the beginning of a break and have to be resumed immediately at the end of the break means that in interval $[S_i, C_i[$, activity $i$ is in progress at time $t$ exactly if $b_i(t) = 1$ (recall that by assumption, the time lag between two breaks in calendar $b_i$ is not less than $\varepsilon_i$). Thus, given start time $S_i$ the completion time of activity $i \in V$ is uniquely determined by

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

(notice that since $\int_{S_i}^{t} b_i(\tau)\, d\tau$ is a continuous function in $t$, the minimum always exists). $C_i$ depends on start time $S_i$ and activity calendar $b_i$. We have $C_i(S_i) - S_i \geq p_i$ for all $i \in V^{bi}$ whereas $C_i(S_i) - S_i = p_i$ for all $i \in V^{ni}$.

Minimum and maximum time lags may depend on calendars, too. For example, if a spare part has to be delivered within three working days, we have to consider a corresponding calendar. Therefore, we introduce a *time lag calendar* $b_{ij}$ for each prescribed time lag $d_{ij}^{min}$ and $d_{ji}^{max}$, which has to be specified when modeling the problem. Point in time $t$ is taken into account when computing the (working) time lag between the start of activities $i$ and $j$ exactly if $b_{ij}(t) = 1$, i.e., $\int_{S_i}^{S_j} b_{ij}(\tau)\, d\tau$ 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. In general, the time lag calendar $b_{ij}$ for a minimum time lag $d_{ij}^{min}$ coincides with $b_i$ or $b_j$. The time lag calendar $b_{ij}$ of a maximum time lag $d_{ji}^{max}$ may contain the spare times of activity calendars $b_h$, where activities $h$ lie on a path from $i$ to $j$ in project network $N$. The case where time lags $d_{ij}^{min}$ and $d_{ij}^{max}$ are independent of calendars is obviously contained as the special case where $b_{ij}(t) = 1$ for all $t \geq 0$.

The actual minimum difference $\Delta_{ij}$ between start times $S_j$ and $S_i$ induced by minimum time lag $d_{ij}^{min}$ or maximum time lag $d_{ji}^{max}$, respectively, depends on start time $S_i$ and calendar $b_{ij}$:

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

where $\delta_{ij} = d_{ij}^{min}$ in case of a minimum time lag between the start of activities $i$ and $j$ and $\delta_{ij} = -d_{ji}^{max}$ if there is a maximum time lag between the start of activities $j$ and $i$. $S_i + \Delta_{ij}(S_i)$ is the earliest point in time $t$ 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 $b_i(t) \in \{0, 1\}$ for all $t \geq 0$, it holds that $|\Delta_{ij}(S_i)| \geq |\delta_{ij}|$, and $\Delta_{ij}(S_i)$ and $\delta_{ij}$ have the same sign.

As a consequence, the temporal constraints (1) have to be replaced by

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

which can also be written as

$$\int_{S_i}^{S_j} b_{ij}(\tau)\, d\tau \geq \delta_{ij} \quad (\langle i,j\rangle \in E) \tag{3}$$

The interpretation of (3) 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 start of activity $i$ at time $S_i$ and the start of activity $j$ at time $S_j$ must be at least $\delta_{ij}$. 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 in case of $\delta_{ij} \geq 0$, inequality (3) is tighter than the ordinary temporal constraint $S_j - S_i \geq \delta_{ij}$, whereas for $\delta_{ij} < 0$, inequality (3) does not imply $S_j - S_i \geq \delta_{ij}$. For given start time $S_i$ of activity $i$, the minimum start time $S_j$ of activity $j$ satisfying (3) is

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

$t^*$ can obviously be computed in $\mathcal{O}(B_{ij})$ time, where $B_{ij}$ designates the number of breaks in calendar $b_{ij}$.

In what follows, we refer to constraints (2) and (3) as *calendar constraints*. A schedule $S$ complying with all calendar constraints is termed *calendar-feasible*. The problem of finding the earliest calendar-feasible schedule can be formulated as follows:

$$\left.\begin{array}{ll} \text{Minimize} & \sum_{i\in V} S_i \\ \text{subject to} & (2) \text{ and } (3) \\ & S_0 = 0 \end{array}\right\} \tag{4}$$

Note that the requirement that activities $i \in V^{bi}$ are not interrupted during working time and immediately resumed after a break is implicitly satisfied by the definition of the corresponding completion time $C_i(S_i)$. The *temporal scheduling problem* (4) has been addressed for the first time by Zhan (1992), who has devised a pseudo-polynomial solution method of type label-correcting. In the next section, we will discuss a similar approach with polynomial time complexity, where polynomiality is achieved by considering only events such as the beginning or termination of breaks or the end of time lags instead of time periods. The procedures generalize algorithms for finding an earliest and a latest calendar-feasible schedule discussed in Franck (1999, Ch. 3) for the case of integral start times and without minimum execution times.

## 3 Temporal scheduling

For solving problem (4), we use a modification of a label-correcting longest-path algorithm (cf. Franck, 1999, Sect. 3.3). We start the algorithm with $ES =$

$(0, -\infty, \ldots, -\infty)$ and successively delay activities until all calendar constraints are satisfied. Let $Q$ be a queue containing activities for which a (tentative) earliest start time $ES_i$ has been determined. In each iteration, we delete an activity $i$ from $Q$. First, we check whether or not start time $ES_i$ complies with calendar $b_i$ by computing the earliest point in time $t^* \geq ES_i$ for which interval $[t^*, t^* + \varepsilon_i[$ does not contain spare times (cf. constraints (2)). In case of $ES_i < t^*$, the start of activity $i$ must be delayed until time $t^*$. Next, we check inequalities (3) and (2) for all direct successors $j \in Succ(i)$ of activity $i$ in project network $N$. To this end, we compute the earliest start time $t^* := \min\{t \geq \max(0, ES_j) \mid \int_{ES_i}^{t} b_{ij}(\tau)\, d\tau \geq \delta_{ij}\}$ 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 $\tau \in [t^*, t^* + \varepsilon_j[$, we add $j$ to $Q$ if $j \notin Q$. Figure 2 summarizes this procedure, where for convenience we define $\min \emptyset := \infty$.

```
**For** all $i \in V \setminus \{0\}$ **do** $ES_i := -\infty$
$ES_0 := 0$, $Q := \{0\}$ $(*\, Q$ is a queue $*)$
**While** $Q \neq \emptyset$ **do**
    Delete $i$ from queue $Q$
    Determine $t^* := \min\{t \geq ES_i \mid b_i(\tau) = 1$ for all $\tau \in [t, t + \varepsilon_i[\}$
    **If** $t^* = \infty$ **then** terminate $(*$ there is no feasible solution $*)$
    **else if** $ES_i < t^*$ **then** $ES_i := t^*$
    **For** all $j \in Succ(i)$ **do**
        Determine $t^* := \min\{t \geq \max(0, ES_j) \mid \int_{ES_i}^{t} b_{ij}(\tau)\, d\tau \geq \delta_{ij}\}$
        **If** $ES_j < t^*$ **or** $b_j(\tau) = 0$ for some $\tau \in [t^*, t^* + \varepsilon_j[$ **then**
            **If** $ES_j < t^*$ **then** $ES_j := t^*$
            **If** $j \notin Q$ **then** push $j$ onto queue $Q$
        **end** $(*$ if $*)$
    **end** $(*$ for $*)$
**end** $(*$ while $*)$
**Return** $ES$
```

**Fig. 2.** Calendarization — earliest schedule

Let $B := \sum_{i \in V} B_i + \sum_{\langle i,j \rangle \in E} B_{ij}$ denote the number of breaks in all activity and time lag calendars. Clearly, the (tentative) earliest start times $ES_i$ of all activities $i \in V$ are nondecreasing in the course of the algorithm. If temporal scheduling problem (4) is solvable, the algorithm of Figure 2 yields schedule $ES$ after having inspected each arc $\langle i, j \rangle$ at most $|V|(B + 1)$ times, where each of the $B$ breaks is considered at most once. If the calendars are given as sorted lists of start and end times of breaks, the time complexity of the algorithm of Figure 2 is $\mathcal{O}[|V||E|(B + 1)]$. There is no calendar-feasible schedule exactly if after $|V||E|(B+1)$ iterations, queue $Q$ still contains some activity $i \in V$.

The latest schedule $LS$ maximizing $\sum_{i \in V} S_i$ can be computed as follows. Let $\overline{d}$ denote the maximum project duration given either by a prescribed deadline for the termination of the project or by an upper bound on the minimum project duration. We start with $LS = (0, \infty, \ldots, \infty, \overline{d})$ and $Q = \{0, n + 1\}$. In each iteration, an activity $j$ is removed from queue $Q$ and $LS_j$ is set equal to the latest time $t^*$

for which calendar $b_j$ allows the start of activity $j$. Then, the tentative latest start times of all $i \in Pred(j)$ are set equal to the largest times $t^* \leq \min(\bar{d}, LS_i)$ with $\int_{t^*}^{LS_j} b_{ij}(\tau)\, d\tau \geq \delta_{ij}$. The corresponding procedure is given by Figure 3, where $\max \emptyset := -\infty$.

```
**For** all $j \in V \setminus \{0, n+1\}$ **do** $LS_j := \infty$
$LS_0 := 0$, $LS_{n+1} := \bar{d}$, $Q := \{0, n+1\}$ ($*$ $Q$ is a queue $*$)
**While** $Q \neq \emptyset$ **do**
    Delete $j$ from queue $Q$
    Determine $t^* := \max\{t \leq LS_j \mid b_j(\tau) = 1$ for all $\tau \in [t, t + \varepsilon_j[\}$
    **If** $t^* = -\infty$ **then** terminate ($*$ there is no feasible solution $*$)
    **else if** $LS_j > t^*$ **then** $LS_j := t^*$
    **For** all $i \in Pred(j)$ **do**
        Determine $t^* := \max\{t \leq \min(\bar{d}, LS_i) \mid \int_t^{LS_j} b_{ij}(\tau)\, d\tau \geq \delta_{ij}\}$
        **If** $LS_i > t^*$ **or** $b_i(\tau) = 0$ for some $\tau \in [t^*, t^* + \varepsilon_i[$ **then**
            **If** $LS_i > t^*$ **then** $LS_i := t^*$
            **If** $i \notin Q$ **then** push $i$ onto queue $Q$
        **end** ($*$ if $*$)
    **end** ($*$ for $*$)
**end** ($*$ while $*$)
**Return** $LS$
```

**Fig. 3.** Calendarization — latest schedule

## 4 Resource-constrained scheduling

We now turn to the problem of minimizing the project duration $S_{n+1}$ subject to calendar constraints (2) and (3) and renewable-resource constraints which arise from the use of scarce manpower or machinery. For each resource $k \in \mathcal{R}$, we have a limited capacity $R_k \in \mathbb{N}$. The processing of real activities $i = 1, \ldots, n$ in interval $[S_i, C_i[$ takes up $r_{ik} \in \mathbb{Z}_{\geq 0}$ units of resource $k$. In particular, we assume that the interruption of activities does not release resources. Note that otherwise each break would incur a new setup of the resources before resuming the interrupted activities. With $r_k(S,t) := \sum_{i \in V : S_i \leq t < C_i} r_{ik}$ denoting the utilization of resource $k$ at time $t$, the resource constraints read as

$$r_k(S,t) \leq R_k \quad (k \in \mathcal{R}; t \geq 0) \tag{5}$$

A schedule $S$ which satisfies inequalities (5) is termed *resource-feasible*. The problem of finding a resource- and calendar-feasible schedule is strongly NP-hard even if $b_i \equiv 1$ for all $i \in V$ and $b_{ij} \equiv 1$ for all $\langle i,j \rangle \in E$ (see Bartusch et al., 1988).

In what follows, we sketch the adaptation of a priority-rule method (see Franck et al., 2001) for the problem of minimizing the project duration $S_{n+1}$ subject to temporal and resource constraints to the following more general *resource-constrained calendarized scheduling problem*

$$\begin{array}{ll} \text{Minimize} & S_{n+1} \\ \text{subject to} & (2) \text{ and } (3) \\ & S_0 = 0 \\ & r_k(S,t) \leq R_k \quad (k \in \mathcal{R}; t \geq 0) \end{array}$$

The priority-rule method by Franck et al. (2001) is, in principle, as follows. Let $\prec$ be some strict order in set $V$, e.g. given by $i \prec j$ exactly if either (a) $d_{ij} > 0$ or (b) $d_{ij} = 0$ as well as $d_{ji} < 0$, where $d_{ij}$ is the longest path length from node $i$ to node $j$ in network $N$. The algorithm is based on the *serial schedule generation scheme*, which schedules the activities $i \in V$ one after the other. Let $\mathcal{C}$ be the *completed set* of activities that have already been scheduled. In each iteration, an *eligible activity* $j^*$ (i.e. an activity $j \in \overline{\mathcal{C}} := V \setminus \mathcal{C}$ for which $h \in \mathcal{C}$ holds for all activities $h$ with $h \prec j$) is scheduled at its earliest resource-feasible start time $t^* \geq ES_{j^*}$ provided that $t^* \leq LS_{j^*}$. If there are several eligible activities $j$, activity $j^*$ is chosen according to some *priority rule*. Then, the earliest and latest start times of all activities $j \in \overline{\mathcal{C}}$ as well as the available resource capacities are updated. If $t^* > LS_{j^*}$, activity $j^*$ cannot be scheduled without violating a maximum time lag $d_{ij^*}^{max}$ between some scheduled activity $i \in \mathcal{C}$ and $j^*$. In that case, we perform an *unscheduling step* which deletes all activities $h \in \mathcal{C}$ with $S_h \leq S_i$ from set $\mathcal{C}$ and increases the earliest start time of activity $i$ by $t^* - LS_{j^*}$.

For computing the latest start times, the algorithm requires an upper bound $\overline{d}$ on the minimum project duration, which in case of calendars can be found as follows. For each arc $\langle i,j \rangle \in E$, we determine an upper bound $\overline{\Delta}_{ij}$ on $\Delta_{ij}(S_i)$ for any start time $S_i \geq ES_i$. If $\delta_{ij} \leq 0$, it follows from $\Delta_{ij}(S_i) \leq 0$ and $|\Delta_{ij}(S_i)| \geq |\delta_{ij}|$ that $\delta_{ij} \geq \Delta_{ij}(S_i)$. If $\delta_{ij} > 0$, $\overline{\Delta}_{ij}$ can be chosen to be the largest time lag $\Delta_{ij}(t) = t' - t$ between the start of activity $i$ at some time $S_i = t$ and the earliest point in time $t' \geq t + \delta_{ij}$ for which the total working time in interval $[t, t'[$ equals $\delta_{ij}$. Thus, we set

$$\overline{\Delta}_{ij} := \begin{cases} \delta_{ij}, & \text{if } \delta_{ij} \leq 0 \\ \max\limits_{t \geq ES_i} \Delta_{ij}(t), & \text{otherwise} \end{cases}$$

An upper bound $\overline{p}_i$ on the time lag between some time $t \geq ES_i$ and the earliest completion of activity $i \in V$ after point in time $t$ can be determined as follows. The smallest start time $S_i \geq t$ of activity $i$ is the minimum time $t' \geq t$ satisfying $b_i(\tau) = 1$ for all $\tau \in [t', t' + \varepsilon_i[$. By taking the "largest" time lag $C_i(t') - t$ between $t$ and the earliest completion of $i$ at time $C_i(t') \geq t + p_i$ with respect to all times $t \geq ES_i$ (where the maximum does not necessarily exist), we obtain

$$\overline{p}_i := \sup_{t \geq ES_i} \min_{t' \geq t} \{C_i(t') - t \mid b_i(\tau) = 1 \text{ for all } \tau \in [t', t' + \varepsilon_i[\}$$

Figure 4 illustrates the computation of upper bound $\overline{p}_i$ for an activity $i \in V^{bi}$ with $ES_i = 0, p_i = 6$, and $\varepsilon_i = 1$. For $5 < t \leq 8$, we have $t' = 8$ and $C_i(t') = 17$. Since for all other points in time $t$, $C_i(t') - t$ is less than or equal to 11 (where we

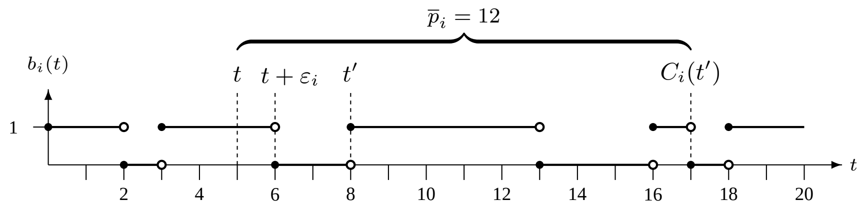

**Fig. 4.** Upper bound $\overline{p}_i$

suppose that $b_i(t) = 1$ for all $t \geq 18$), $\overline{p}_i$ equals $\sup_{5<t\leq 8}(17-t) = 17-5 = 12$. Note that there is no maximum $C_i(t') - t$ with respect to $t$.

Analogously to the case without calendars (cf. Franck et al., 2001),

$$\overline{d} := \sum_{i\in V} \max(\overline{p}_i, \max_{j\in Succ(i)} \overline{\Delta}_{ij})$$

is an upper bound on the minimum project duration provided that there exists a schedule observing the calendar and resource constraints. Since upper bounds $\overline{\Delta}_{ij}$ and $\overline{p}_i$ for $\langle i,j\rangle \in E$ and $i \in V$, respectively, can be determined by considering starts and ends of breaks, $\overline{d}$ can be computed in $\mathcal{O}[\max(|E|, B)]$ time.

A calendar- and resource-feasible schedule can be found by the priority-rule method using an adaptation of the serial schedule generation scheme. For updating the earliest and latest start times $ES_j$ and $LS_j$ of activities $j \in \overline{\mathcal{C}}$, we use the algorithms from Figures 2 and 3, respectively. In contrast to the problem without calendars, not all points in time $t \in [ES_{j^*}, LS_{j^*}]$ with $r_k(S,t) \leq R_k$ for all $k \in \mathcal{R}$ represent feasible start times for the activity $j^*$ selected. Thus, it may happen that a resource-feasible start time $t^*$ with $ES_{j^*} \leq t^* \leq LS_{j^*}$ for activity $j^*$ has to be rejected because start time $S_{j^*} = t^*$ does not comply with constraints (2). In that case, the start of $j^*$ must be delayed by increasing $ES_{j^*}$ up to the earliest point in time $t$ with $t^* < t \leq LS_{j^*}$ for which activity $j^*$ can be processed without interruption for at least $\varepsilon_{j^*}$ units of time.

The priority-rule method has been tested by Franck (1999, Sect. 5.2) using a set of 1080 projects with 100 activities and five resources each (test set $\mathrm{B}^{cal}$). This test set extends a collection of benchmark instances without calendars by adding activity and time lag calendars, where 20% of the time periods fall into a break. The experimental performance analysis has shown that for the resource-constrained project scheduling problem with calendar constraints, the priority-rule method requires about ten times as much computing time as for the corresponding problem with ordinary temporal constraints on the average. This is mainly due to a larger number of unscheduling steps and the more time-consuming temporal scheduling algorithm (where temporal scheduling with calendars is the more expensive the greater is the number $B$ of all breaks in calendars). For both problems, the minimum latest start time first (LST) rule selecting the eligible activity $j$ with smallest $LS_j$ provides the best results among the five priority rules tested (LST, minimum slack time first MST, most total successors first MTS, longest path following first LPF, and resource scheduling method RSM; for details see Franck et al., 2001). In particular, the LST rule is well-suited for both finding a large number of feasible schedules and computing schedules with small project duration.

## References

Bartusch M, Möhring RH, Radermacher FJ (1988) Scheduling project networks with resource constraints and time windows. Annals of Operations Research 16:201–240

Brucker P (2001) Scheduling Algorithms. Springer, Berlin Heidelberg New York

Franck B (1999) Prioritätsregelverfahren für die ressourcenbeschränkte Projektplanung mit und ohne Kalender. Shaker, Aachen

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

Neumann K, Schwindt C (1997) Activity-on-node networks with minimal and maximal time-lags and their application to make-to-order production. OR Spektrum 19:205–217

Neumann K, Schwindt C, Trautmann N (2001) Short-term planning of batch plants in process industries. In: Kischka P, Leopold-Wildburger U, Möhring RH, Radermacher FJ (eds.) Models, methods and decision support for management, pp. 211–226. Physica, Heidelberg

Schwindt C, Trautmann N (2000) Batch scheduling in process industries: An application of resource-constrained project scheduling. OR Spektrum 22:501–524

Zhan J (1992) Calendarization of time-planning in MPM networks. ZOR – Methods and Models of Operations Research 36:423–438