Models and solution procedures for the resource-constrained project scheduling problem with general temporal constraints and calendars

Stefan Kreter, Julia Rieck, Jürgen Zimmermann

Clausthal University of Technology, Institute of Management and Economics, Operations Research Group, Julius-Albert-Str. 2, 38678 Clausthal-Zellerfeld, Germany

Abstract

In this paper, the resource-constrained project scheduling problem with general temporal constraints is extended by the concept of break-calendars in order to incorporate the possible absence of renewable resources. Three binary linear model formulations are presented that use either start-based or changeover-based or execution-based binary decision variables. In addition, a priority-rule method as well as a scatter search procedure are proposed in order to solve the problem heuristically. All exact and heuristic solution procedures use a new and powerful time planning method, which identifies all time- and calendar-feasible start times for activities as well as all corresponding absolute time lags between activities. In a comprehensive performance analysis, small- and medium-scale instances are solved with CPLEX 12.6. Furthermore, large-scale instances of the problem are tackled with scatter search, where the results are compared with a random search procedure.

Keywords: Resource-constrained project scheduling, Calendars, Minimum and maximum time lags, Binary linear model formulations, Scatter search

1. Introduction

Resource-constrained project scheduling is concerned with the assignment of execution time intervals to activities such that given temporal constraints between activities are satisfied, the prescribed resource capacities are not exceeded, and some objective, e.g., the project duration, is minimized (see, e.g., Neumann et al., 2003b or Józefowska and Węglarz, 2006). Temporal constraints are usually given by simple precedence constraints (the problem is then abbreviated

*Corresponding authors

Email addresses: stefan.kreter@tu-clausthal.de (Stefan Kreter), julia.rieck@tu-clausthal.de (Julia Rieck), juergen.zimmermann@tu-clausthal.de (Jürgen Zimmermann)

by RCPSP) or by general minimum and maximum time lags (RCPSP/max). Applications of resource-constrained project scheduling can be found, e.g.,

- in make-to-order production, where resource-demanding operations, which need to be performed for the manufacturing of customer orders, have to be scheduled (Neumann and Schwindt, 1997),
- at software implementations, where resources have to be selected to perform a task, resources are generally multi-skilled, and the time required for completing a task depends on the number of assigned resources (Alba and Chicano, 2007), and
- at service centers, where tasks have to be scheduled and assigned to a multi-skilled workforce with heterogeneous efficiencies (Valls et al., 2009).

For projects with a short-term planning horizon, breaks like week-ends or holidays have to be incorporated, where some of the renewable resources needed are not available. Particularly, the consideration of "break-calendars" (or "calendars") is necessary for many operative RCPSP-applications in which persons (e.g., employees, teachers, or customers) are involved. In practice, different renewable resources generally have different *break-calendars*. For example, the employees' working days per week vary depending on the type of job, the season, or the labor contract. A typical office employee works Monday to Friday, and a possible 4-weeks work plan for a nurse is 6–3–7–3–7–2 (work days and days off mentioned alternately). In contrast to persons, renewable resources like machines or equipment can be treated as always available at a company, since maintenance work can be modeled as separate activities that have to be performed in regular intervals. Thus, no breaks have to be considered in the respective calendars.

For practical purposes, we have to distinguish between activities that can be *interrupted* during breaks as well as activities that must be performed *without interruption*. The first group contains, e.g., assembling, drilling, and cutting processes, or activities of the quality management that must be executed manually in order to maintain a preferred level of quality. The second group contains, e.g., heating, cooling, drying, and pouring processes, or training activities, where staff have to travel over long distances to reach off-site training locations. For some interruptible activities, a *start-up phase* must be used during which the activity has to be processed without suspending the execution. It is obvious that, e.g., management consultants should not start working at a company on a Friday morning. If their corresponding work plan contains a 5-day week, a start-up phase of two days ensures that a consulting phase only starts on a day between Monday to Thursday. Moreover, activities should only be paused during breaks to guarantee that employees are focused on a small number of activities and do not change frequently between different types of work. Hence, a general preemption of activities at all integer points in time (in addition to break start times) is not acceptable in order to reduce processing errors and guarantee high quality of intermediate and end-products (Coviello et al., 2010).

Additionally, many machines require skilled staff to operate, so that complex activities can only be performed if both machines and employees are on hand. Since machines are usually blocked by, e.g., components or individual parts during breaks, they must *stay engaged* even if the personal is not on site. In order to ensure that the machines are blocked as short as possible, an activity should only be interrupted for the duration of the longest break of the assigned staff. By only allowing interruptions of activities at breaks, an introductory training course (for safety, health, or environmental requirements) must only be considered once and can be included into the processing time of an activity.

Typically, an activity requires several renewable resources for execution. Hence, an "activity calendar" can be obtained on the basis of all calendars of the associated resources. In addition to resources and activities, minimum and maximum time lags between activities may also depend on break-calendars. An example is the delivery of a spare part that has been processed at a plant and must be delivered to another plant for subsequent manufacturing processes. If the delivery can only be performed by a specific less-than-truckload carrier that needs three work days for transportation, then, the "absolute" delivery time may take five days when a week-end is included.

Consequently, the consideration of break-calendars for resources, activities, and time lags, interruptible and non-interruptible activities, resources that are either engaged or released during breaks, and activity start-up phases is highly important and should not be ignored for operative RCPSP-applications. Therefore, the aim of the paper on hand is to incorporate all of the above mentioned features and to develop targeting models and solution procedures.

The remainder of this paper is organized as follows: Section 2 presents related literature of resource-constrained project scheduling problems with general temporal and calendar constraints (RCPSP/max-cal). In Section 3 the proposed RCPSP/max-cal is formally described and an example is given to illustrate the construction of an associated activity-on-node network. Section 4 contains an enhanced approach for finding the set of feasible start times of all activities and all corresponding absolute time lags between activities. In Section 5, three different binary linear model formulations for the RCPSP/max-cal are presented. Section 6 is devoted to solution procedures in order to solve the problem heuristically. The binary models as well as the heuristic methods benefit from the time planning procedure significantly. The results of a comprehensive performance analysis are given in Section 7, where it is shown that many instances with 50 activities can only be solved to optimality, if the time planning procedure is incorporated. Finally, conclusions are presented in Section 8.

2. Literature review

Scheduling project activities subject to break-calendars (referred to as "calendarization") has been rarely considered in the literature and no paper includes all of the features presented in the introduction (particularly, resources that are either engaged or released during breaks and start-up phases). The RCPSP/max with a single resource calendar has been introduced by Zhan (1988, 1992), where a model formulation and a time planning procedure are presented and no resource capacity is taken into account. Franck (1999) proposed the calendarization with multiple calendars and without considering a start-up phase for activities. Efficient methods for computing earliest and latest start and completion times of activities are devised and priority-rule methods for minimizing the project duration are introduced. Franck et al. (2001a) as well as Neumann et al. (2003b, Sect. 2.11) described procedures to determine the earliest and latest schedule, where a start-up phase is included. Moreover, the authors sketched how priority-rule methods for the RCPSP/max can be adapted to break-calendars, but all resources stay engaged during interruptions, i.e., a release of resources during breaks is not incorporated. The priority-rule methods presented in Franck (1999), Franck et al. (2001a), and Neumann et al. (2003b, Sect. 2.11) provide that the time planning method is passed in each iteration, which is computationally inefficient.

Schwindt and Trautmann (2000) considered a batch scheduling problem with minimum and maximum time lags between successive production levels, sequence-dependent facility setup times, finite intermediate storages, production breaks, and activity break-calendars. The solution procedure is based on a relaxation, where sequence-dependent setup times, safety stock restrictions, storage and resource capacities are neglected. The method (approximately) solves an industrial problem within a reasonable amount of time. Trautmann (2001) developed an alternative approach to the calendarization of projects, which makes use of calendar-independent start-start, start-end, end-start, and end-end time lags. Cheng et al. (2014) introduced the RCPSP with "non-preemptive activity splitting", where an activity in process is allowed to pause only when resource levels are temporarily insufficient. Further, the activity must be resumed at the next eligible point in time. The authors examined the differences between RCPSP with no activity splitting, RCPSP with non-preemptive activity splitting, and preemptive RCPSP (which allows activities to be interrupted in any time period and resumed later at no additional cost) in terms of problem settings, mathematical formulations, and optimal solution properties.

In their survey, Hartmann and Briskorn (2010) noted that the calendarization has similarities with the problem involving resource capacities varying with time. However, the main differences of the problem under consideration to the problem with time-dependent resource capacities (RCPSP/t or RCPSP/max-t) are that not only resources but also time lags can depend on calendars and resources may remain engaged during a break. The RCPSP/t with no activity splitting has been discussed, e.g., in Klein (2000a), Bomsdorf and Derigs (2008), as well as Hartmann (2014). Furthermore, the preemptive RCPSP/t has been investigated by Buddhakulsomsiri and Kim (2006, 2007).

3. Problem description

A project can be treated as a set V of interacting activities requiring time and renewable resources for their execution. The project in question consists of $n + 2$ activities, numbered from 0 to $n + 1$, where fictitious activity 0 represents the project beginning and fictitious activity $n + 1$ the project completion, respectively. Each activity i is associated with a processing time $p_i \in \mathbb{N}_0$. All fictitious activities i with $p_i = 0$, i.e., activities 0, $n + 1$, as well as milestones, are classified into set $V^f \subset V$, and all real activities i with $p_i > 0$ are included into set $V^r := V \setminus V^f$. The start time (completion time) of activity i is denoted by $S_i \in \mathbb{N}_0$ ($C_i \in \mathbb{N}_0$). We assume that the project on hand is started at time zero, i.e., $S_0 := 0$. The project duration (i.e., the makespan) is given by S_{n+1}. A vector of start times $S = (S_i)_{i=0,\dots,n+1}$ with $S_i \geq 0$ and $S_0 = 0$ is termed a "schedule".

In practice, minimum and maximum time lags between activities, which result from technological and organizational constraints, are given, and form the temporal constraints of the project. Activities and time lags are depicted by an activity-on-node network $N := (V, A; \delta)$, where V represents the set of nodes (i.e., activities), A represents the set of arcs (i.e., time lags) and δ describes the arc weights. For a minimum time lag $d_{ij}^{\min} \in \mathbb{N}_0$ between activities i and j, an arc $\langle i, j\rangle$ having weight $\delta_{ij} := d_{ij}^{\min}$ is introduced, and for a maximum time lag $d_{ij}^{max} \in \mathbb{N}_0$, a backward arc $\langle j, i\rangle$ with weight $\delta_{ji} := -d_{ij}^{\max}$ is inserted into network N. The assumption that the durations and time lags are to be integers is generally satisfied, where time is usually measured in minutes, hours, or days.

If a break appears in the underlying calendar during activity execution, the respective activity must be interrupted for the duration of the break, otherwise, the activity continues without stopping (in contrast to preemptive activities that can be interrupted at any point in time and as long as necessary; cf., e.g., Cheng et al., 2014). The set of (break-)interruptible activities is denoted by $V^{bi} \subset V$ and the set of non-interruptible activities by $V^{ni} = V \setminus V^{bi}$. Note that an activity i with $p_i = 0$ is declared as non-interruptible. For each interruptible activity i, a start-up or training phase $\varepsilon_i \in \mathbb{N}$ is prescribed during which i has to be in progress without interruption. For activities $i \in V^{ni}$, we set $\varepsilon_i := p_i$, i.e., i has to be processed during p_i time units without suspending the execution.

In order to schedule activities subject to breaks, we consider a time axis $[0, \overline{d}]$, where $\overline{d} \in \mathbb{N}$ is a prescribed maximum project duration which must not be exceeded. A suitable determination of $\overline{d}$ is necessary in order to define the time horizon for calendars. In what follows, we proceed on the assumption that the time axis is divided into intervals $[0, 1), [1, 2), \dots, [\overline{d} - 1, \overline{d})$. Activities may only be started (whether at first or after an interruption) at the beginning of such intervals or at time $\overline{d}$, i.e., at points in time $t \in T := \{0, 1, \dots, \overline{d}\}$. Since p_i and δ_{ij} are integers for all $i, j \in V$, there always exists an optimal solution with integer start times, i.e., $S_i \in \{0, 1, \dots, \overline{d}\}, i \in V$ (Neumann et al., 2003a, Sect. 1.3). Let $\mathcal{R}$ be the set of renewable resources available at each point in time, independently of their utilization formerly. Each resource $k \in \mathcal{R}$ is associated with a limited resource capacity $R_k \in \mathbb{N}$. Moreover, $r_{ik} \in \mathbb{N}_0$ represents the amount of resource k used constantly by activity $i \in V$ during an "operating phase", i.e., a phase consisting of one or more periods in which resource k is engaged by activity i (see also condition (1)). For fictitious activities $j \in V^f$, we set $r_{jk} := 0$ for all $k \in \mathcal{R}$. A resource calendar can be defined as follows:

Definition 1. *A calendar for resource $k \in \mathcal{R}$ is a step function $\mathbf{Cal}_k(\cdot) : [0, \overline{d}) \to \{0,1\}$ continuous from the right at the jump points, where the condition*

$$\mathbf{Cal}_k(t) := \begin{cases} 1, & \text{if period } [\lfloor t \rfloor, \lfloor t+1 \rfloor) \text{ is a working period for } k \\ 0, & \text{if period } [\lfloor t \rfloor, \lfloor t+1 \rfloor) \text{ is a break period for } k \end{cases}$$

is satisfied.

Based on resource calendars, calendars for each project activity may be determined. The set of resources that is necessary to carry out activity $i \in V$ is denoted by $\mathcal{R}_i := \{k \in \mathcal{R} \mid r_{ik} > 0\}$. An activity calendar $\mathbf{C}_i(\cdot) : [0, \overline{d}) \to \{0,1\}$ for activity i can then be specified by

$$\mathbf{C}_i(t) := \begin{cases} \min_{k \in \mathcal{R}_i} \mathbf{Cal}_k(t), & \text{if } \mathcal{R}_i \neq \emptyset \\ 1, & \text{otherwise.} \end{cases}$$

The execution of activity $i \in V^{bi}$ must be interrupted at times t with $\mathbf{C}_i(t) = 0$, and the execution must be continued at the next point in time $\tau > t$ with $\mathbf{C}_i(\tau) = 1$. Consequently, a non-interruptible activity $i \in V^{ni}$ can only be started at a time $t \in T$ such that p_i working periods follow in the respective activity calendar of i. Thus, the completion time of $i \in V^{ni}$ may be determined by $C_i(S_i) := S_i + p_i = S_i + \sum_{t=S_i}^{S_i+p_i-1} \mathbf{C}_i(t)$. In contrast, an interruptible activity $i \in V^{bi}$ may start at all times $t \in T$ with $\sum_{\tau=t}^{t+\varepsilon_i-1} \mathbf{C}_i(\tau) = \varepsilon_i$, and the completion time can be computed by $C_i(S_i) := \min\{t \geq S_i + p_i | \sum_{\tau=S_i}^{t-1} \mathbf{C}_i(\tau) = p_i\}$.

A time lag between activities i and j is of type start-start (SS), start-end (SE), end-start (ES), or end-end (EE), and it refers to a time lag calendar which depends on resource set $\mathcal{R}_{ij} \subseteq \mathcal{R}$ (Franck, 1999, Sect. 3.1). Usually, $\mathcal{R}_{ij}$ is equal to $\emptyset, \mathcal{R}_i, \mathcal{R}_j$, or $\mathcal{R}_i \cup \mathcal{R}_j$. Practical examples are:

- Activity j with $p_j = 5$ must be executed exactly 40% before activity i ends. Then, time lags ${}^{\text{ES}}d_{ij}^{\min} = -2$ and ${}^{\text{ES}}d_{ij}^{\max} = -2$ need resources $\mathcal{R}_j$, i.e., $\mathcal{R}_{ij} = \mathcal{R}_j$.
- Activity j must be started when activity i with $p_i = 5$ is executed exactly 60%. In that case, time lags ${}^{\text{SS}}d_{ij}^{\min} = 3$ and ${}^{\text{SS}}d_{ij}^{\max} = 3$ involve resources $\mathcal{R}_i$, i.e., $\mathcal{R}_{ij} = \mathcal{R}_i$.
- A maximum time lag $d_{ij}^{\max}$ exists between activities i and j, and minimum time lags exist between activities i and l, l and m, as well as m and j. Then, $d_{ij}^{\max}$ might need resources $\mathcal{R}_{ij} = \mathcal{R}_i \cup \mathcal{R}_l \cup \mathcal{R}_m \cup \mathcal{R}_j$.

Based on this information, a calendar $\mathbf{C}_{ij}(\cdot) : [0, \overline{d}) \to \{0,1\}$ for each time lag between activities i and j may be computed by

$$\mathbf{C}_{ij}(t) := \begin{cases} \min_{k \in \mathcal{R}_{ij}} \mathbf{Cal}_k(t), & \text{if } \mathcal{R}_{ij} \neq \emptyset \\ 1, & \text{otherwise.} \end{cases}$$

In what follows, we want to concentrate on activity-on-node networks $N := (V, A; \delta)$ with start-start time lags. In order to convert end-start, start-end, and end-end time lags into start-start ones, sometimes dummy activities l or l, q, as well as dummy arcs $\langle i, l\rangle, \langle l, j\rangle$, or $\langle i, l\rangle, \langle l, q\rangle, \langle q, j\rangle$ have to be inserted (Franck, 1999, Sect. 3.1). For example, if activity i manufactures a product at a high temperature and the product must be cooled at least three time units before it can be processed again (in activity j), the end-start time lag between i and j can only be transformed into start-start time lags by introducing a dummy activity that symbolizes the end of activity i or the start of the cooling process, respectively. The different transformation rules are given in Table 1, note that $T' := T \setminus \{\overline{d}\}$.

$\delta_{ij}^{\text{SS}} := \delta_{ij}^{\text{ES}} + p_i,$	if $\mathbf{C}_{ij}(t) = 1$ for all $t \in T', i \in V^{ni}$ or $\mathbf{C}_{ij} = \mathbf{C}_i$
$\delta_{il}^{\text{SS}} := p_i, \mathcal{R}_{il} := \mathcal{R}_i,\ \delta_{lj}^{\text{SS}} := \delta_{ij}^{\text{ES}}, \mathcal{R}_{lj} := \mathcal{R}_{ij},$	otherwise
$\delta_{ij}^{\text{SS}} := \delta_{ij}^{\text{SE}} - p_j,$	if $\mathbf{C}_{ij}(t) = 1$ for all $t \in T', j \in V^{ni}$ or $\mathbf{C}_{ij} = \mathbf{C}_j$
$\delta_{il}^{\text{SS}} := \delta_{ij}^{\text{SE}}, \mathcal{R}_{il} := \mathcal{R}_{ij},\ \delta_{lj}^{\text{SS}} := -p_j, \mathcal{R}_{lj} := \mathcal{R}_j,$	otherwise
$\delta_{ij}^{\text{SS}} := \delta_{ij}^{\text{EE}} + p_i - p_j,$	if $\mathbf{C}_{ij}(t) = 1$ for all $t \in T', i, j \in V^{ni}$
$\delta_{il}^{\text{SS}} := p_i, \mathcal{R}_{il} := \mathcal{R}_i,\ \delta_{lj}^{\text{SS}} := \delta_{ij}^{\text{EE}} - p_j, \mathcal{R}_{lj} := \mathcal{R}_{ij},$	if $\mathbf{C}_{ij} = \mathbf{C}_j$
$\delta_{il}^{\text{SS}} := \delta_{ij}^{\text{EE}} + p_i, \mathcal{R}_{il} := \mathcal{R}_{ij},\ \delta_{lj}^{\text{SS}} := -p_j, \mathcal{R}_{lj} := \mathcal{R}_j,$	if $\mathbf{C}_{ij} = \mathbf{C}_i$
$\delta_{il}^{\text{SS}} := p_i, \mathcal{R}_{il} := \mathcal{R}_i,\ \delta_{lq}^{\text{SS}} := \delta_{ij}^{\text{EE}}, \mathcal{R}_{lq} := \mathcal{R}_{ij},\ \delta_{qj}^{\text{SS}} := -p_j, \mathcal{R}_{qj} := \mathcal{R}_j,$	otherwise.

Table 1: Transformation rules for time lags

In practice, some renewable resources remain engaged during a break of activity $i \in V^{bi}$, like machines or equipment, and some renewable resources are released during a break, e.g., workers or tools. In order to incorporate this situation (in contrast to Franck et al., 2001a and Neumann et al., 2003b, Sect. 2.11), parameter $\rho_k, k \in \mathcal{R}$, is used as follows:

$$\rho_k := \begin{cases} 1, & \text{if resource } k \text{ stays engaged during an interruption} \\ 0, & \text{otherwise.} \end{cases}$$

For a given schedule S, the set V^r of real activities that are in execution at time $t \in [0, \overline{d})$ is termed the "active set". The active set is given by

$$\mathcal{A}(S, t) := \{i \in V^r \mid S_i \le t < C_i(S_i)\},$$

where each activity i is supposed to be carried out during the half-open time interval $[S_i, C_i(S_i))$ (see, e.g., Neumann et al., 2003a). Thus, the resource utilization $r_k^{\text{cal}}(S, t)$ of resource $k \in \mathcal{R}$ at time t according to schedule S and calendars $\mathbf{C}_i, i \in V$, can be computed by

$$r_k^{\text{cal}}(S, t) := \sum_{i \in \mathcal{A}(S,t) \mid \mathbf{C}_i(t)=1} r_{ik} + \sum_{i \in \mathcal{A}(S,t) \mid \mathbf{C}_i(t)=0} r_{ik}\, \rho_k. \tag{1}$$

Function $r_k^{\text{cal}}(S,\cdot)$ is called the resource profile of resource k. It observes the "operating phases" of all real activities.

The aim of the problem under consideration is to find a schedule such that all time lags, resource capacities, and calendar constraints are satisfied and the project makespan is minimized. Based on Franck (1999, Sect. 3.1), the following conceptual model for the RCPSP/max-cal can be formulated:

$$\begin{aligned}
&\text{Minimize} \quad S_{n+1} && && (2)\\
&\text{subject to} && && \\
&\sum_{t=S_i}^{S_i+\varepsilon_i-1} \mathbf{C}_i(t) = \varepsilon_i && i \in V && (3)\\
&\sum_{t=S_i}^{S_j-1} \mathbf{C}_{ij}(t) - \sum_{t=S_j}^{S_i-1} \mathbf{C}_{ij}(t) \geq \delta_{ij} && \langle i,j\rangle \in A && (4)\\
&r_k^{\text{cal}}(S,t) \leq R_k && k \in \mathcal{R}, t \in T' && (5)\\
&S_i \in \{0,1,\ldots,\overline{d}\} && i \in V. && (6)
\end{aligned}$$

Objective function (2) represents the makespan, i.e., the start time of the project completion, which is to be minimized. Equalities (3) ensure on the one hand that non-interruptible activities $i \in V^{ni}$ are carried out without interruption and on the other hand that interruptible activities $i \in V^{bi}$ are started at the beginning of a working period and observe the start-up phase of ε_i time units. Constraints (4) make sure that the temporal constraints are satisfied and inequalities (5) guarantee that the given resource capacities R_k are not exceeded at any time. Finally, constraints (6) describe all possible start times of the activities.

In order to illustrate the construction of an activity-on-node network within calendarization and to demonstrate the properties of a solution, we introduce a practical example. Integrated production and maintenance planning is a task that lots of companies have to deal with (Budai et al., 2008). In what follows, we consider the production and maintenance planning for a common printing press. The resulting project consists of five real activities and requires four renewable resources (cf. Table 2).

no.	activity i
0	project beginning
1	printing (order 1)
2	maintenance of printing press
3	test run of printing press
$3'$	end of test run
4	printing (order 2)
5	installation of switchbox
6	project completion

no.	resource k	R_k	ρ_k
1	external specialist	2	0
2	machine operator	2	0
3	printing press	1	1
4	electrician	1	0

Table 2: Activities and renewable resources

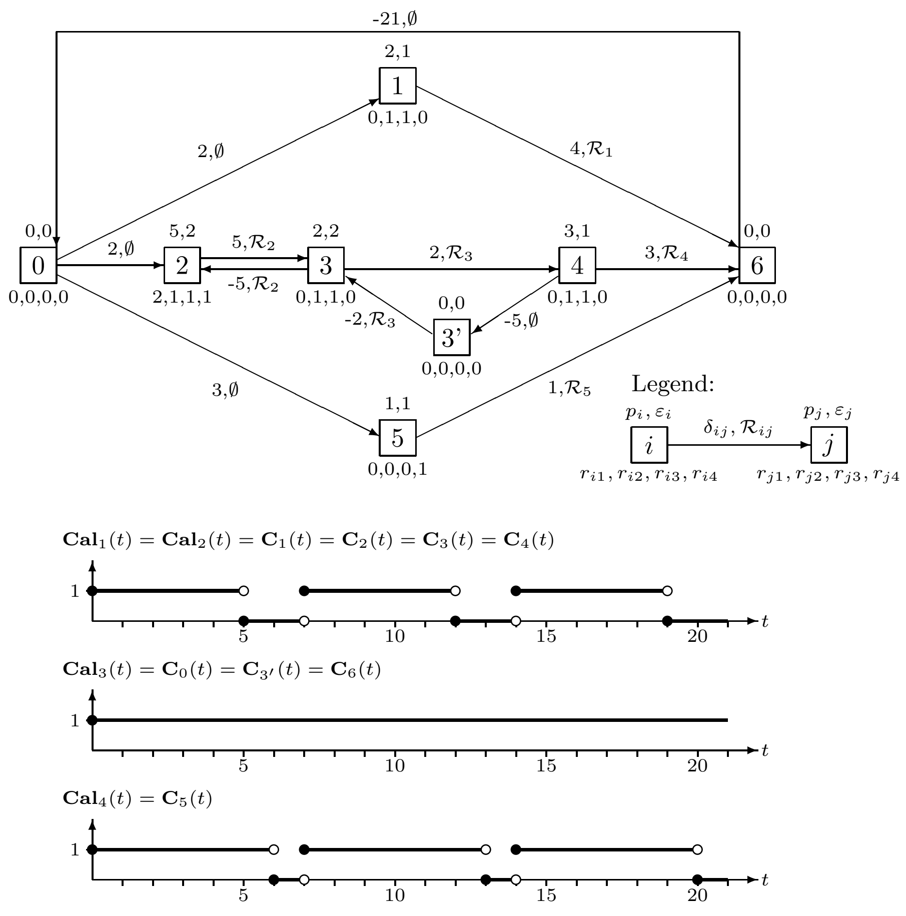


Figure 1: Activity-on-node network and resource calendars

The associated activity-on-node network as well as the resource calendars are depicted in Figure 1. Each node $i \in V$ is weighted with the processing time p_i, the length ε_i of the start-up phase, and the resource requirements r_{ik}, $k = 1, \ldots, 4$. For example, activity 1 (printing order 1) has a processing time of $p_1 = 2$ days, a start-up phase of $\varepsilon_1 = 1$ day, and needs one machine operator as well as the printing press for its execution ($r_{12} = r_{13} = 1$, $r_{11} = r_{14} = 0$). The resource calendar $\mathbf{Cal}_3(\cdot)$ of the printing press is always one. Consequently, the activity calendar of printing order 1 equals the resource calendar of the machine operator, i.e., $\mathbf{C}_1(\cdot) = \mathbf{Cal}_2(\cdot)$. The maintenance of the printing press (activity 2) requires two external specialists, one machine operator, the printing press, and an electrician. Hence, the resulting activity calendar $\mathbf{C}_2(\cdot)$ includes a 5-day working week (on the analogy of $\mathbf{Cal}_k(\cdot), k = 1, 2$). Since external specialists should not start an activity on Friday, a start-up phase of $\varepsilon_2 = 2$ days is also included. The test run (activity 3) must be executed without interruptions. Therefore, we set $p_3 = \varepsilon_3 = 2$.

Arc $\langle 6, 0 \rangle$ displays the planning horizon of $\overline{d} = 21$ days. Furthermore, release dates of activities $i = 1, 2, 5$ are modeled using arcs $\langle 0, i \rangle$. The test run of the printing press (activity 3) must be started directly after the end of the maintenance work (activity 2), i.e., an end-start time lag with $\delta_{23}^{\mathrm{ES}} = 0$ and a start-end time lag with $\delta_{32}^{\mathrm{SE}} = 0$ is given ($\mathcal{R}_{23} = \mathcal{R}_{32} = \mathcal{R}_2$). Using the transformation rules declared in Table 1, we obtain $\delta_{23} = 5$ and $\delta_{32} = -5$. Additionally, printing order 2 (activity 4) is time sensitive and must be scheduled five days after the end of the test run at the latest, i.e., time lag $\delta_{43}^{\mathrm{SE}} = -5$ is required ($\mathcal{R}_{43} = \emptyset$). The start-end time lag is converted by introducing a dummy activity $3'$ and two start-start time lags $\delta_{43'} = -5, \mathcal{R}_{43} = \emptyset$ and $\delta_{3'3} = -2, \mathcal{R}_{3'3} = \mathcal{R}_3$.

The resource profiles of an optimal solution $S^* = (0, 11, 2, 9, 11, 15, 5, 18)$ are given in Figure 2. During interruptions, the printing press stays engaged while all other resources are released. Moreover, the electrician with a 6-day working week can mount a switchbox (activity 5) on the first Saturday (period $[5, 6)$) when the maintenance activity is interrupted. Furthermore, the resource profiles visualize the difference between "relative" and "absolute" processing times as well as time lags. All values given in network N are "relative", because they only include working days (in contrast to "absolute" values that involve non-working and working days). For example, the solution considers an "absolute" processing time of activity 1 which is equal to 4 ($\neq p_1 = 2$) and an "absolute" time lag between activities 2 and 3 which is equal to 7 ($\neq \delta_{23} = 5$). The calculation of absolute time lags between start times of activities $i, j \in V$ is treated in the following section.

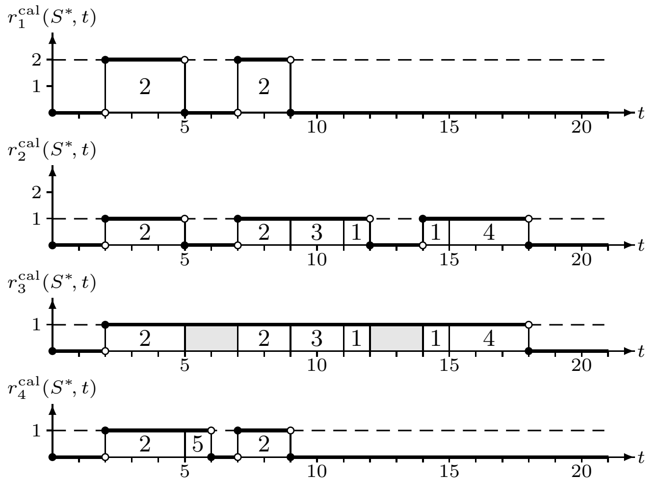


Figure 2: Resource profiles of an optimal solution $S^* = (0, 11, 2, 9, 11, 15, 5, 18)$

4. Temporal planning

In contrast to procedures presented in the literature, which contain only methods to identify the earliest and latest schedule, the temporal planning described in this section determines all feasible start times of project activities as well as all corresponding "absolute" time lags between activities. Hence, the procedure must only be performed in a preprocessing step and not in all iterations of a solution algorithm.

In order to compute the earliest ES_i and latest LS_i start times of activity $i \in V$, a modification of the label-correcting algorithm (Ahuja et al., 1993, Sect. 5.4) for finding the longest paths in networks can be used (Franck et al., 2001a). The algorithm begins with $ES_0 := 0$ and $ES_i := -\infty, i \in V \setminus \{0\}$. Then, activities are successively delayed until all calendar constraints are satisfied. Moreover, the latest start times of activities are initialized by $LS_0 := 0$, $LS_i := \infty, i \in V \setminus \{0, n+1\}$, and $LS_{n+1} := \overline{d}$. In each iteration, a particular activity j is chosen and LS_j is set equal to the latest time for which all temporal constraints are satisfied and activity calendar $\mathbf{C}_j(\cdot)$ allows the start of j. Afterwards, the tentative latest start times of all predecessors i of activity j are set equal to the largest times $t^* \leq \min\{\overline{d}, LS_i\}$ for which the time lag between i and j is satisfied. Since the start times of activities are integers, we are able to restrict the potential start times of activity i to a set of discrete times $U_i := \{ES_i, \ldots, LS_i\} \subseteq T$. However, current set U_i may contain points in time that are infeasible start times for activity i according to the underlying calendar constraints. In order to eliminate these times and insert them in a set $N_i, i \in V$, a novel procedure for temporal planning is shown in Algorithm 1.

Algorithm 1 Time planning procedure

```
$W_i := U_i$ and $N_i := \emptyset$, for all $i \in V$, stop := false (* *Initialization* *)
for all $i \in V$ and $t \in W_i$ do (* *Consideration of start-up phases* *)
    if $\sum_{\tau=t}^{t+\varepsilon_i-1} \mathbf{C}_i(\tau) \neq \varepsilon_i$ then
        Set $W_i := W_i \setminus \{t\}$ and $N_i := N_i \cup \{t\}$
while stop = false do (* *Enhanced version of the triple algorithm* *)
    stop := true
    Set $d_{iit} := 0$ for $i \in V$ and $t \in W_i$
    Set $d_{ijt}$ for $i, j \in V, i \neq j$, $t \in W_i$ as follows:
    $d_{ijt} := \begin{cases} \min\{\tau \in W_j \mid \sum_{z=t}^{\tau-1} \mathbf{C}_{ij}(z) \geq \delta_{ij}\} - t, & \text{if } \delta_{ij} > 0 \\ \min\{\tau \in W_j \mid \sum_{z=\tau}^{t-1} \mathbf{C}_{ij}(z) \leq -\delta_{ij}\} - t, & \text{if } \delta_{ij} \leq 0 \\ -\infty, & \text{otherwise} \end{cases}$
    for all $\nu \in V$ do
        for all $i \in V \setminus \{\nu\}$ do
            for all $t \in W_i$ with $d_{i\nu t} > -\infty$ do
                for all $j \in V \setminus \{\nu\}$ with $d_{ijt} < d_{i\nu t} + d_{\nu,j,t+d_{i\nu t}}$ do
                    Set $d_{ijt} := d_{i\nu t} + d_{\nu,j,t+d_{i\nu t}}$
    for all $i \in V$ and $t \in W_i$ do
        if $d_{iit} > 0$ then
            Set $W_i := W_i \setminus \{t\}$, $N_i := N_i \cup \{t\}$ and stop := false
return $W_i$, $N_i$, and matrix $D$ for all $i, j \in V, t \in W_i$.            ($\star$)
```

Within the procedure, we first initialize sets $W_i := U_i$ and $N_i := \emptyset$. Then, all start times $t \in W_i$ are eliminated for which the start-up phase of ε_i time units cannot be observed. Afterwards, the lengths of the longest paths between all activities are computed using an enhanced version of the well-known Floyd-Warshall triple algorithm (Ahuja et al., 1993, Sect. 5.6). Note that the length of the longest path between activities i and j depends on the start times of i and j and the corresponding time lag calendar $\mathbf{C}_{ij}(\cdot)$. First, the length of the longest path d_{iit} from i to i at time $t \in W_i$ is set to zero. Then, the current "absolute" time lags or distances respectively (include non-working days besides working days) between activities i and j, $\langle i,j\rangle \in A$, are computed. The absolute time lags are based on $t \in W_i$ and are determined such that $t + d_{ijt} \in W_j$ is satisfied. If $\langle i,j\rangle \notin A$ and $i \neq j$, we set $d_{ijt} := -\infty$. The main part of the triple algorithm consists in verifying whether a path from i to j via activity ν exists that is longer than the current longest path from i to j. If the algorithm provides $d_{iit} > 0$ for some $i \in V$ and $t \in W_i$, there is a cycle of positive length and t is infeasible. The algorithm returns sets $W_i \subseteq U_i$ that contain only feasible (i.e., "calendar-feasible") start times of activities i, and $N_i \subset U_i$ that include all infeasible start times between ES_i and LS_i. Furthermore, the current matrix $D := (d_{ijt})_{i,j\in V, t\in W_i}$ of "absolute" time lags is returned. Mixed-integer programming formulations profit a lot from the time planning procedure (cf. the performance study in Sect. 7). Moreover, matrix D enables us to adapt RCPSP/max heuristic procedures directly, without passing the time planning method in each iteration as it is done in Franck (1999), Franck et al. (2001a), and Neumann et al. (2003b, Sect. 2.11).

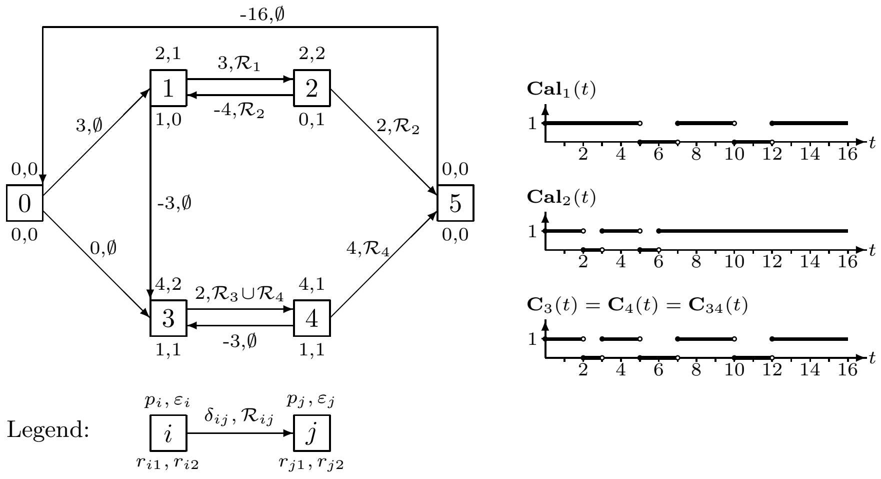


Figure 3: Activity-on-node network with two renewable resources

In order to illustrate the temporal planning, we consider a project with four real activities and two renewable resources. Activities 0, 2, and 5 are non-interruptible, whereas activities 1, 3, and 4 may be interrupted during breaks, i.e., $V^{ni} = \{0, 2, 5\}$ and $V^{bi} = \{1, 3, 5\}$. The project network N as well as the relevant resource, activity, and time lag calendars are given in Figure 3.

Table 3 shows the results of the different phases of Algorithm 1. Particularly, the number of elements in sets $W_i, i = 1, 3, 4$, is decreased. The new sets of feasible start times can be used within model formulations and solution procedures. In our performance study, the reduction of sets W_i caused by our time planning procedure will be precisely investigated (cf. Sect. 7).

	initialization	start-up phase	triple algorithm
W_0	$\{0\}$	$\{0\}$	$\{0\}$
W_1	$\{3, 4, \ldots, 7\}$	$\{3, 4, 7\}$	$\{3, 4, 7\}$
W_2	$\{8, 9, 10, 11\}$	$\{8, 9, 10, 11\}$	$\{8, 9, 10, 11\}$
W_3	$\{0, 1, \ldots, 7\}$	$\{0, 3, 7\}$	$\{0, 7\}$
W_4	$\{3, 4, \ldots, 9\}$	$\{3, 4, 7, 8, 9\}$	$\{3, 9\}$
W_5	$\{10, 11, \ldots, 16\}$	$\{10, 11, \ldots, 16\}$	$\{10, 11, \ldots, 16\}$

Table 3: Temporal planning

5. Model formulations

In this section, we present three compact binary linear model formulations for the proposed RCPSP/max-cal. All formulations represent extensions of models which has been studied in the literature for the pure RCPSP. The main differences of the models are the representation of solutions. The first formulation applies binary variables that allocate a feasible start time to each activity (cf. Sect. 5.1). The second model uses binary variables in order to determine if an activity starts at time t or earlier (cf. Sect. 5.2). Finally, the third formulation is based on binary variables which indicate whether an activity is in execution at time t or not (cf. Sect. 5.3).

Besides the adapted models, some other formulations for the RCPSP may be found in the literature (see, e.g., Artigues et al., 2014). Artigues et al. (2003) considered a flow-based formulation that cannot be easily extended to the RCPSP/max-cal, since the transfer of resources at calender-breaks cannot be described. Zapata et al. (2008), as well as Koné et al. (2011), and Artigues et al. (2013) proposed continuous-time formulations based on events. Thereby, start (end) variables a^+_{ie} (a^-_{ie}) which indicate whether activity i starts (ends) at event e and on/off variables $\bar{a}_{ie}$ which denote whether activity i is in progress at event e are used. Additionally, continuous event date variables $t_e \geq 0$ are involved that determine the time at which event e occurs. As t_e represents a decision variable for event e, matrix $D := (d_{ijt})_{i,j \in V, t \in W_i}$ of absolute time lags as well as completion times $C_i(t)$ of activities $i \in V$ cannot be computed in advance. Hence, the event-based formulations are not suitable for the RCPSP/max-cal. Alvarez-Valdés and Tamarit (1993) described a minimal-forbidden-set and Mingozzi et al. (1998) a feasible-subset formulation. Both models could be conditionally stated for the problem under consideration, where a comprehensive preprocessing phase must be passed in order to determine all “forbidden sets” (i.e., sets containing activities which cannot be processed simultaneously due to resource constraints) and all "feasible subsets" (i.e., sets containing activities that can be processed simultaneously without exceeding the resource capacities). In general, the cardinality of both sets and therefore the number of constraints of the corresponding formulations grows exponentially in the number of activities. Furthermore, the different sets have to be generated depending on the underlying points in time, since, e.g., a subset of activities may be resource-feasible at a break and infeasible otherwise. Since Brucker and Knust (2012, Sect. 3.3) stated that models with an exponential number of constraints cannot be used in practice, we restrict our study to formulations which require only the temporal planning (cf. Sect. 4) as preprocessing.

5.1. Start-based model

Most mathematical formulations for the RCPSP and its variants are adopted from the start-based models presented in Pritsker et al. (1969) or Christofides et al. (1987). Thereby, binary decision variables x_{it} are specified for each activity $i \in V$ and each time $t \in W_i$ as follows:

$$x_{it} := \begin{cases} 1, & \text{if activity } i \text{ starts at time } t \\ 0, & \text{otherwise.} \end{cases}$$

Considering sets W_i, N_i, as well as matrix D, the discrete time-based model for the RCPSP/max-cal has the form:

$$\text{Minimize} \quad \sum_{t \in W_{n+1}} t\, x_{n+1,t} \tag{7}$$

subject to

$$\sum_{t \in W_i} x_{it} = 1 \qquad i \in V \tag{8}$$

$$\sum_{t \in W_j} t\, x_{jt} \geq \sum_{t \in W_i} (t + d_{ijt})\, x_{it} \qquad \langle i,j \rangle \in A \tag{9}$$

$$\sum_{i \in V^r} r_{ik} \sum_{\tau \in \mathcal{T}_{it}} x_{i\tau} C_{itk} \leq R_k \qquad k \in \mathcal{R}, t \in T' \tag{10}$$

$$x_{it} \in \{0,1\} \qquad i \in V, t \in W_i. \tag{11}$$

The makespan, which is to be minimized in objective function (7), is given by the start time of the project completion. Constraints (8) ensure that each activity receives exactly one start time that is feasible with respect to calendar constraints. Since we are able to compute the start time of activity i from $S_i = \sum_{t \in W_i} t\, x_{it}$, inequalities (9) guarantee that the temporal constraints involved will be satisfied. Set

$$\mathcal{T}_{it} := \{\beta_{it} + 1, \beta_{it} + 2, \ldots, t\} \cap W_i \tag{12}$$

contains all calendar-feasible start times of real activity $i \in V^r$, requiring that i is in the active set at time t, i.e.,

$$\beta_{it} := \max\{\tau \in \{0, 1, \ldots, t - p_i\} \mid \sum_{z=\tau}^{t-1} \mathbf{C}_i(z) \geq p_i\},$$

and we set $\max \emptyset := -1$. Moreover, parameter

$$C_{itk} := \mathbf{C}_i(t) + \rho_k(1 - \mathbf{C}_i(t)) \tag{13}$$

is equal to one if activity $i \in V^r$ demands resource k at time t, i.e., period $[t, t + 1)$ is an operating phase for i. Hence, inequalities (10) guarantee that the resource constraints will be met at every point in time. Finally, (11) are the binary constraints on the assignment variables. The start-based formulation incorporates $\mathcal{O}(|V|^2 + |\mathcal{R}||T'|)$ constraints as well as $\sum_{i \in V} |W_i|$ binary decision variables.

In order to obtain a stronger linear programming (LP) relaxation, inequalities (9) can be replaced by the following disaggregated constraints:

$$\sum_{\tau \in W_i \mid \tau \leq \xi_{ijt}} x_{i\tau} - \sum_{\tau \in W_j \mid \tau \leq t} x_{j\tau} \geq 0 \quad \langle i, j \rangle \in A, t \in W_j. \tag{14}$$

The idea of formulating disaggregated constraints to improve the LP-relaxation goes back to Christofides et al. (1987). Within (14), parameter

$$\xi_{ijt} := \max\{\tau \in W_i \mid \tau + d_{ij\tau} \leq t\} \tag{15}$$

equals the largest feasible start time of activity i such that the temporal constraint between activities i and j is satisfied at $S_j \geq t$ (Artigues et al., 2014).

5.2. Changeover-based model

The changeover-based model has been considered by Sankaran et al. (1999) and Klein (2000b, Sect. 3.2) for the RCPSP. The formulation uses binary decision variables $y_{it}, i \in V, t \in W_i$, which are defined as follows:

$$y_{it} := \begin{cases} 1, & \text{if activity } i \in V \text{ starts at time } t \text{ or earlier} \\ 0, & \text{otherwise.} \end{cases}$$

Hence, a feasible solution for activity i consists of a consecutive block of ones usually preceded by a block of zeros. The changeover between two blocks takes place at that point in time at which activity i is started. Since we only consider times $t \in W_i$ for each activity $i \in V$, we define

$$\varrho_{it} := \begin{cases} \max\{\tau \in W_i \mid \tau < t\}, & \text{if } t > ES_i \\ ES_i - 1, & \text{if } t = ES_i \end{cases}$$

that is equal to the next smaller time $\tau < t$ in set W_i or to $ES_i - 1$. The changeover-based model can then be formulated by

$$\text{Minimize} \quad \sum_{t \in W_{n+1}} t\,(y_{n+1,t} - y_{n+1,\varrho_{n+1,t}}) \tag{16}$$

subject to

$$y_{i,ES_i-1} = 0 \qquad i \in V \tag{17}$$

$$y_{i,LS_i} = 1 \qquad i \in V \tag{18}$$

$$y_{i,\varrho_{it}} \leq y_{it} \qquad i \in V, t \in W_i \tag{19}$$

$$\sum_{t \in W_j} t\,(y_{jt} - y_{j,\varrho_{jt}}) \geq \sum_{t \in W_i} (t + d_{ijt})\,(y_{it} - y_{i,\varrho_{it}}) \qquad \langle i,j \rangle \in A \tag{20}$$

$$\sum_{i \in V^r} r_{ik} \sum_{\tau \in \mathcal{T}_{it}} (\,y_{i\tau} - y_{i,\varrho_{i\tau}})\,C_{itk} \leq R_k \qquad k \in \mathcal{R}, t \in T' \tag{21}$$

$$y_{it} \in \{0,1\} \qquad i \in V, t \in W_i. \tag{22}$$

Objective function (16) represents the makespan which is to be minimized. Equalities (17) make sure that no activity i starts at time $ES_i - 1$ or earlier and equalities (18) incorporate that each activity is scheduled. For an activity i, constraints (19) guarantee that all decision variables $y_{i\tau}$ with $\tau \geq S_i$ are equal to one. Inequalities (20) ensure that the temporal constraints will be satisfied. Resource constraints are given in (21), where it is exploited that activity i is started at time $\tau \in \mathcal{T}_{it}$ and requires resources when parameter C_{itk} is equal to one (cf. conditions (12) and (13)). The last constraints (22) define the binary variables. The model contains $\mathcal{O}(|V|^2 + \sum_{i \in V} |W_i| + |\mathcal{R}||T'|)$ constraints along with $\sum_{i \in V} |W_i|$ binary decision variables.

Inequalities (20) can again be replaced by disaggregated constraints as follows:

$$\sum_{\tau \in W_i \,|\, \tau \leq \xi_{ijt}} (y_{i\tau} - y_{i,\varrho_{i\tau}}) - \sum_{\tau \in W_j \,|\, \tau \leq t} (y_{j\tau} - y_{j,\varrho_{j\tau}}) \geq 0 \quad \langle i,j \rangle \in A, t \in W_j, \tag{23}$$

where ξ_{ijt} is taken directly from condition (15).

The changeover-based and the start-based model can be translated into each other by using the transformations

$$x_{it} = y_{it} - y_{i,\varrho_{it}} \quad \text{and} \quad y_{it} = \sum_{\tau \in W_i | \tau \leq t} x_{i\tau}$$

for $i \in V$ and $t \in W_i$. Thus, the formulations are "equivalent" and the transformation does not change the value of the LP-relaxation. However, the models usually require different solution times within a branch-and-cut approach (cf. Sect. 7).

5.3. Execution-based model

The execution-based model has initially been introduced by Kaplan (1988) for the RCPSP with preemption, by Klein (2000b, Sect. 3.2) for the RCPSP, as well as by Neumann and Morlock (2002, Sect. 3.7) for the resource leveling problem. The formulation makes use of the fact that real activity $i \in V^r$ can be executed somewhere during the half-open time interval $[ES_i, LC_i)$, where LC_i is the latest completion time of activity i, and fictitious activity $j \in V^f$ has to be scheduled within interval $[ES_j, LS_j]$. Hence, binary decision variables z_{it} are used for all $i \in V$ and $t \in \mathcal{W}_i := \{ES_i, ES_i + 1, \ldots, \max\{LC_i - 1, LS_i\}\}$ in the following way:

$$z_{it} := \begin{cases} 1, & \text{if activity } i \in V \text{ is in execution at time } t \in \mathcal{W}_i \\ 0, & \text{otherwise.} \end{cases}$$

A feasible solution for activity i consists of a block of ones usually preceded and succeeded by a block of zeros. The first switch of blocks occurs at that point in time at which activity i is started and the second switch appears when i is completed, i.e., at time $C_i(S_i)$. The execution-based model for the RCPSP/max-cal can be given by

$$\text{Minimize} \quad \sum_{t \in W_{n+1}} t\, z_{n+1,t} \tag{24}$$

subject to

$$z_{i,ES_i-1} = 0 \qquad i \in V \tag{25}$$

$$\sum_{t \in W_i} z_{it} = 1 \qquad i \in V^f \tag{26}$$

$$\sum_{t \in N_i} z_{it} = 0 \qquad i \in V^f \tag{27}$$

$$\sum_{t \in \mathcal{W}_i} z_{it}\, \mathbf{C}_i(t) = p_i \qquad i \in V^r \tag{28}$$

$$z_{it} \le z_{i,t-1} \qquad i \in V^r, t \in T_i^- \tag{29}$$

$$z_{it} \le z_{i,t+1} \qquad i \in V^r, t \in T_i^+ \tag{30}$$

$$\sum_{\tau=t}^{C_i(t)-1} z_{i\tau} \ge (z_{it} - z_{i,t-1})(C_i(t) - t) \qquad i \in V^r, t \in W_i \tag{31}$$

$$\sum_{\tau \in \mathcal{T}_{ijt}} z_{j\tau}\, \mathbf{C}_j(\tau) \ge (z_{it} - z_{i,t-1}) \max(p_j, 1) \qquad \langle i,j \rangle \in A, t \in W_i \tag{32}$$

$$\sum_{i \in V^r | t \in \mathcal{W}_i} r_{ik}\, z_{it}\, C_{itk} \le R_k \qquad k \in \mathcal{R}, t \in T' \tag{33}$$

$$z_{it} \in \{0,1\} \qquad i \in V, t \in \mathcal{W}_i. \tag{34}$$

Objective function (24) represents the makespan which is to be minimized.

The first constraints (25) ensure that no activity i is in execution at time $ES_i - 1$. Equalities (26) and (27) guarantee that the fictitious activities are scheduled only once and at feasible start times. Real activity i must be processed exactly p_i time units (cf. constraints (28)). Using set $T_i^- := N_i \cup \{\tau \in \{LS_i + 1, LS_i + 2, \ldots, LC_i - 1\} \mid \mathbf{C}_i(\tau) = 0\}$, inequalities (29) ensure that no real activity is started at an infeasible point in time or at a break appearing after the latest start time. Moreover, constraints (30) make sure that the last execution time of real activity i, i.e., $C_i(S_i) - 1$, is not positioned in a break period, where $T_i^+ := \{\tau \in \{ES_i + 1, ES_i + 2, \ldots, LC_i - 1\} \mid \mathbf{C}_i(\tau) = 0\}$. Inequalities (31) guarantee that a feasible solution involves only one block of ones for each activity i. Set

$$\mathcal{T}_{ijt} := \{t + d_{ijt}, t + d_{ijt} + 1, \ldots, \max\{LC_j - 1, LS_j\}\}$$

contains all points in time at which activity j can be in execution assuming that activity i starts at time $t \in W_i$, i.e., the points in time are feasible according to the temporal constraints. The $\max\{\ldots\}$-term is necessary in order to consider fictitious activities with $LS_j > LC_j - 1$ as well as real activities with $LS_j \leq LC_j - 1$. The right hand side of (32) receives the value p_j (the value 1) if real activity $i \in V^r$ (fictitious activity $i \in V^f$) starts at time $t \in W_i$. Inequalities (33) guarantee that the resource constraints will be met and (34) are the binary constraints. The model incorporates $\mathcal{O}(|V| \sum_{i \in V} |W_i| + |V^r||T| + |\mathcal{R}||T'|)$ constraints as well as $\sum_{i \in V} |\mathcal{W}_i|$ binary decision variables. Since the number of binary variables is larger than in the start-based (or changeover-based) model, the models cannot be translated directly into one another and cannot be identified as "equivalent".

All model formulations (either start- or changeover- or execution-based) can be used to solve small- and medium-scale RCPSP/max-cal instances to optimality (or with a small solution gap within a prescribed time limitation) using a common MIP-solver. In order to generate upper bounds for problem instances or to solve large-scale instances, heuristic solution procedures are required.

6. Heuristic solution procedures

Several efficient solution procedures for resource-constrained project scheduling problems are based on schedule generation schemes that fix the start times of project activities iteratively. If the activity to be scheduled next is chosen applying a predefined priority rule, the whole procedure is described as *priority-rule method*. In Section 6.1, a priority-rule method is introduced that constructs a feasible (i.e., calendar- and resource-feasible) schedule by using the "serial" schedule generation scheme (SSGS) and a targeting priority rule. The resulting procedure is much more efficient than the existing procedure presented in Franck (1999, Sect. 5.3). In order to generate many different solutions, the priority-rule method is further embedded into a multi-start algorithm. In addition, a *scatter search procedure* is proposed in Section 6.2 in order to improve solutions found by the multi-start algorithm.

6.1. Priority-rule method

The following priority-rule method for the RCPSP/max-cal uses a SSGS, where exactly one activity $i \in V$ is scheduled in each iteration. In the standard version of the SSGS (Kelly, 1963; Talbot and Patterson, 1978), the earliest and latest start times of all activities are computed in a preprocessing step. Once activity i has been scheduled within the algorithm, the earliest and latest start times of all activities not yet scheduled are updated taking account of $\delta_{ij}, i, j \in V$. Since the time lags are "absolute" instead of "relative" within calendarization, a typical updating procedure cannot be used. Therefore, Franck (1999, Sect. 5.3), performs, after fixing an activity start time, the whole procedure for determining earliest and latest start times in order to update ES- and LS-values. In contrast, the main advantage of the method presented in this section is that the matrix $D := (d_{ijt})_{i,j \in V, t \in W_i}$ enables us to return to the standard version and updating procedure of the SSGS (with only few adaptations), which produces significantly better run times than the Franck (1999) version. In what follows, all parts of our priority-rule method (i.e., SSGS, priority-rules, and unscheduling) are described in detail.

The completed set $\Omega \subseteq V$ contains all activities that have already received a start time and can be regarded as completed. At the beginning of the priority-rule method, we put $S_0 := 0$ and $\Omega := \{0\}$. Then, a sequence of feasible "partial" schedules $S^{\Omega} := (S_i)_{i \in \Omega}$ is constructed until a feasible schedule $S := (S_i)_{i \in V}$ is attained. The eligible set $\mathcal{E} \subseteq V \setminus \Omega$ incorporates those activities which are eligible for scheduling. Generally, $\mathcal{E}$ contains all available activities j whose immediate $\prec_{dist}$–predecessors have already been scheduled, i.e., $\mathcal{E} := \{j \in V \setminus \Omega \mid Pred^{\prec_{dist}}(j) \subseteq \Omega\}$.

Definition 2. *Distance order $\prec_{dist}$ is a strict order in node set V. Considering two activities $i, j \in V, i \neq j$, then, condition $i \prec_{dist} j$ holds iff*

- *$d_{ijt} > 0$ for all $t \in W_i$ or*
- *$d_{ijt} \geq 0$ for all $t \in W_i$ and $d_{jit} < 0$ for all $t \in W_j$.*

The activity to be scheduled next is always that activity $j \in \mathcal{E}$ with best priority value $\pi(j)$ and lowest index number. Thus, current activity j is chosen as $j := \min\{i \in \mathcal{E} \mid \pi(i) = \text{ext}_{h \in \mathcal{E}} \pi(h)\}$, where ext $\in \{\min, \max\}$. A large number of different priority rules have been examined in literature for the RCPSP with and without maximum time lags (see, e.g., Neumann et al., 2003b, Sect. 2.6). Under preliminary tests, we found out that the following rules are efficient for the RCPSP/max-cal:

LCT-rule (smallest latest completion time first), i.e.,
$$\underset{h \in \mathcal{E}}{\text{ext}}\, \pi(h) = \min_{h \in \mathcal{E}} LC_h$$

FCS-rule (fewest calendar-feasible start times first), i.e.,
$$\underset{h \in \mathcal{E}}{\text{ext}}\, \pi(h) = \min_{h \in \mathcal{E}} |W_h|.$$

Once activity j has been chosen, its earliest calendar- and resource-feasible start time $t^* \in W_j$ must be determined. The resource profile of partial schedule $S^{\Omega,j} := (S_i)_{i \in \Omega \cup \{j\}}$ is given by

$$r_k^{\text{cal}}(S^{\Omega,j}, t) := \sum_{i \in \mathcal{A}(S^{\Omega,j},t) | \mathbf{C}_i(t)=1} r_{ik} + \sum_{i \in \mathcal{A}(S^{\Omega,j},t) | \mathbf{C}_i(t)=0} r_{ik}\, \rho_k \qquad k \in \mathcal{R}, t \in [0, \overline{d})$$

with $\mathcal{A}(S^{\Omega,j}, t) := \{i \in \Omega \cup \{j\} \,|\, S_i \leq t < C_i(S_i)\}$. For activity j, time t^* can then be set as $t^* := \min\{t \in W_j \mid t \geq ES_j^{\Omega}$ and $r_k^{\text{cal}}(S^{\Omega,j}, \tau) \leq R_k$ for $t \leq \tau < C_j(t)$ and all $k \in \mathcal{R}\}$, where $\min \emptyset := \infty$. In case t^* is larger than the current latest start time of activity j, a so-called *unscheduling step* is performed. Otherwise, we set $S_j := t^*$ and $\Omega := \Omega \cup \{j\}$.

Earliest and latest start times depend on the completed set and are therefore marked with the superscript Ω. After fixing $S_j, j \in \mathcal{E}$, the earliest and latest start times of activities $h \in V \setminus \Omega$ must be updated according to

$$\begin{aligned} ES_h^{\Omega} &:= \max\{ES_h^{\Omega}, S_j + d_{jhS_j}\} \\ LS_h^{\Omega} &:= \min\{LS_h^{\Omega}, \max\{t \in W_h \mid t + d_{hjt} \leq S_j\}\}. \end{aligned}$$

The whole priority-rule method (SSGS and activity choice on the basis of a priority rule) is given in Algorithm 2, where u is the number of unscheduling steps. Moreover, the priority rule is used in a dynamic version, i.e., $\pi(i), i \in \mathcal{E}$, depends on the current partial schedule S^{Ω} and must be computed each time the activity to be scheduled next has to be selected.

Algorithm 2 Priority-rule method for RCPSP/max-cal

```
$S_0 := 0,\ \Omega := \{0\},\ u := 0$ (∗ *Initialization* ∗)
**for all** $i \in V$ **do** Set $ES_i^{\Omega} := ES_i,\ LS_i^{\Omega} := LS_i$
**while** $\Omega \neq V$ **do** (∗ *Main loop* ∗)
    Determine $\mathcal{E} := \{i \in V \setminus \Omega \mid Pred^{\prec_{dist}}(i) \subseteq \Omega\}$
    **for all** $i \in \mathcal{E}$ compute $\pi(i)$
    $j := \min\{i \in \mathcal{E} \mid \pi(i) = \text{ext}_{h \in \mathcal{E}} \pi(h)\}$
    $t^* := \min\{t \in W_j \mid t \geq ES_j^{\Omega}$ and $r_k^{\text{cal}}(S^{\Omega,j}, \tau) \leq R_k$ for $t \leq \tau < C_j(t)$ and all $k \in \mathcal{R}\}$
    **if** $t^* > LS_j^{\Omega}$ **then** $u := u + 1$ and **Unschedule**$(j, t^*)$
    **else** (∗ *Schedule $j$ at time $t^*$* ∗)
        Set $S_j := t^*,\ \Omega := \Omega \cup \{j\}$
        **for all** $h \in V \setminus \Omega$ **do** (∗ *Update $ES_h^{\Omega}$ and $LS_h^{\Omega}$* ∗)
            $ES_h^{\Omega} := \max\{ES_h^{\Omega}, S_j + d_{jhS_j}\}$
            $LS_h^{\Omega} := \min\{LS_h^{\Omega}, \max\{t \in W_h \mid t + d_{hjt} \leq S_j\}\}$
**return** Schedule $S$
```

In case that condition $t^* > LS_j^{\Omega}$ is satisfied, the current latest start time LS_j^{Ω} of activity j not yet scheduled must be extended in an unscheduling step (see also Neumann et al., 2003b, Sect. 2.6). The latest start time of activity j results from the start time of an activity $i \in \Omega$ minus the length of the longest path from j to i assuming that j starts at time LS_j^Ω. Let

$$\mathcal{U} := \{i \in \Omega \mid LS_j^\Omega = S_i - d_{j,i,LS_j^\Omega}\}$$

be the set of all those activities. To increase LS_j^Ω, we unschedule all activities $i \in \mathcal{U}$ and specify their earliest start times by $ES_i^\Omega := t^* + d_{jit^*}$. In addition, we unschedule all activities $i \in \Omega$ with $S_i > \min_{h\in\mathcal{U}} S_h$, which due to the right-shift of the activities from set $\mathcal{U}$ may possibly be started earlier. Afterwards, we have to compute the earliest and latest start times for all activities $h \in V \setminus \Omega$ again which were valid before scheduling activities $i \in \mathcal{U}$. If there is activity zero in set $\mathcal{U}$, a maximum number of unscheduling steps is reached, or $ES_i^\Omega > LS_i$ holds for an activity $i \in \mathcal{U}$, the generation scheme terminates without a feasible solution. The unscheduling procedure is given in Algorithm 3, where $\overline{u}$ describes the maximum number of unscheduling steps.

Algorithm 3 Unschedule(j, t^*)

```
$\mathcal{U} := \{i \in \Omega \mid LS_j^\Omega = S_i - d_{j,i,LS_j^\Omega}\}$
**if** $0 \in \mathcal{U}$ or $u > \overline{u}$ then terminate (∗ *No feasible schedule found* ∗)
**for all** $i \in \mathcal{U}$ **do** (∗ *Right-shift of activities* $i \in \mathcal{U}$ ∗)
    $ES_i^\Omega := t^* + d_{jit^*}$, $\Omega := \Omega \setminus \{i\}$
    **if** $ES_i^\Omega > LS_i$ then terminate (∗ *No feasible schedule found* ∗)
**for all** $i \in \Omega$ with $S_i > \min_{h\in\mathcal{U}} S_h$ **do** (∗ *Unschedule activities* $i$ ∗)
    $\Omega := \Omega \setminus \{i\}$
**for all** $h \in V \setminus \Omega$ **do** (∗ *Update* $ES_h^\Omega$ *and* $LS_h^\Omega$ ∗)
    $ES_h^\Omega := \max\{ES_h, \max_{i\in\mathcal{U}}\{ES_i^\Omega + d_{i,h,ES_i^\Omega}\}\}$
    $LS_h^\Omega := LS_h$
    **for all** $i \in \Omega$ **do**
        $ES_h^\Omega := \max\{ES_h^\Omega, S_i + d_{ihS_i}\}$
        $LS_h^\Omega := \min\{LS_h^\Omega, \max\{t \in W_h \mid t + d_{hit} \le S_i\}\}$
```

We demonstrate the priority-rule method by scheduling the project depicted in Figure 3, for which we have already performed the temporal planning. The resource capacities are $R_1 = 2$ and $R_2 = 3$ and both resources are released during a break (i.e., $\rho_i = 0, i = 1, 2$). Eligible activities are scheduled according to the (dynamic) FCS-rule. Table 4 shows the iterations of Algorithm 2, where column “update” refers to increasing earliest start times or decreasing latest start times after the scheduling of selected activity j at time t^*. Moreover, Figure 4 depicts the resource profiles of the solution $S = (0, 3, 8, 7, 9, 15)$ obtained after running the procedure.

The serial schedule generation scheme is embedded into a multi-start algorithm, where the selection probabilities for activities are determined by using the so-called “regret based biased random sampling” (Sect. 5.3 Kolisch, 1995).

iter.	Ω	$\mathcal{E}$	j	t^*	update
1	{0}	{1,3}	3	0	$LS_1^\Omega = 3$, $LS_2^\Omega = 8$, $LS_4^\Omega = 3$
2	{0,3}	{1,4}	1	3	
3	{0,1,3}	{2,4}	2	8	
4	{0,1,2,3}	{4}	4	$9 > LS_4^\Omega = 3$	
Unschedule$(4,9)$: $\mathcal{U} = \{1,2,3\}$ and $ES_1^\Omega = 3, LS_1^\Omega = 7, ES_2^\Omega = 8,$ $LS_2^\Omega = 11, ES_3^\Omega = LS_3^\Omega = 7, ES_4^\Omega = LS_4^\Omega = 9, ES_5^\Omega = 15$					
5	{0}	{1,3}	3	7	
6	{0,3}	{1,4}	4	9	
7	{0,3,4}	{1}	1	3	$LS_2^\Omega = 8$
8	{0,1,3,4}	{2}	2	8	
9	{0,1,2,3,4}	{5}	5	15	

Table 4: Serial schedule generation scheme with FCS-rule

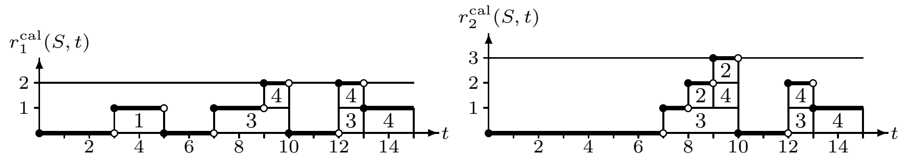


Figure 4: Resource profiles of solution $S = (0, 3, 8, 7, 9, 15)$

In each iteration, the selection probability ψ_i of activity $i \in \mathcal{E}$ is calculated by

$$\psi_i := \frac{\max\limits_{j\in\mathcal{E}}\{\pi(j)\} - \pi(i) + 1}{\sum\limits_{h\in\mathcal{E}}(\max\limits_{j\in\mathcal{E}}\{\pi(j)\} - \pi(h) + 1)}.$$

Therefore, the activity with the smallest priority value for both the LCT- and the FCS-rule obtains the highest selection probability.

6.2. Scatter search procedure

Scatter Search is a population-based metaheuristic that constructs solutions by systematically combining solutions of a reference set. The generic scatter search procedure presented in Glover (1977) or Laguna and Martí (2003) is based on five steps. In the first step, an initial population is generated. Then, a reference set *RefSet* is chosen that is a repository of “good” and “disperse” solutions (step 2). Subsets are iteratively selected from the reference set (step 3), and the solutions of each selected subset are combined, where good characteristics are considered in order to get new high quality solutions (step 4). Step 5 consists in updating the reference set. Steps 3 to 5 are iterated with the new reference set until a stop criterion is met.

The choice of a scatter search procedure for the RCPSP/max-cal has been inspired by the promising results provided by Debels et al. (2006) and Mahdi Mobini et al. (2009) for the basic RCPSP, by Alvarez-Valdés et al. (2006) for the RCPSP with partially renewable resources, by Ranjbar et al. (2009) for trade-off problems in project scheduling, and by Vanhoucke (2010) for maximizing the net present value of resource-constrained projects. Particularly for the RCPSP, where a lot of heuristic solution procedures exist, Kolisch and Hartmann (2006) as well as Mahdi Mobini et al. (2009) showed that scatter search is one of the best approaches to generate good solutions. In what follows, each step of our scatter search procedure is described in detail.

The *initial population* (step 1) is generated by calling $b^2 = |RefSet|^2$ times the multi-start version of the priority-rule method (cf. Algorithm 2). In order to obtain "good" as well as "disperse" solutions, the LCT-, the FCS-, and a "random"-rule (where an activity $i \in \mathcal{E}$ is selected randomly) are used in equal properties. Within the multi-start algorithm, the best solution S^{best} found so far is stored. In the case that condition $S_{n+1} < 1.1 \cdot S^{best}_{n+1}$ holds for some solution S, a forward-backward procedure is used to improve S (Valls et al., 2005; Tormos and Lova, 2001). Thereby, in a backward pass, the activities are considered from right to left and scheduled at the latest feasible time (i.e., they are shifted to the right). Subsequently, in the forward pass, they are considered from left to right and scheduled at the earliest feasible time (i.e., they are shifted back to the left).

From the initial population, a set of b solutions is selected to form the *reference set*, *RefSet*, (step 2). Thereby, b_1 "good" solutions are chosen according to the objective function value (i.e., solutions with shortest makespan), and $b_2 = b - b_1$ "disperse" solutions are determined according to the distances to good solutions. A disperse solution S' is selected if

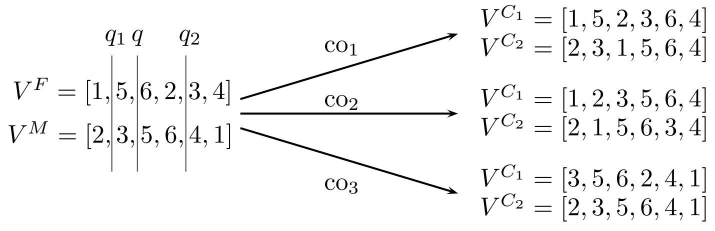
$$\min_{S \in RefSet(b_1)} \left\{ \sum_{i=0}^{n+1} |S_i - S'_i| \right\}$$

is maximal, where set $RefSet(b_1)$ contains all good solutions (Alvarez-Valdés et al., 2006). In order to guarantee that the same solution is not inserted twice in the reference set, a check procedure is performed. It may happen that b^2 runs of the multi-start algorithm (step 1) only produce $\gamma < b$ feasible and different (with respect to start times) schedules. In this case, the number of solutions in the reference set is reduced to γ, (if $\gamma \geq 4$ is satisfied), i.e., $b := \gamma$; otherwise step 1 is executed again.

Within the *subset selection* (step 3), all combinations of two solutions $S, S' \in$ *RefSet*, $S \neq S'$, are considered.

The *combination of solutions* (step 4) is performed on the basis of a specific representation. In particular, the representation of a solution is determined by the sequence in which activities $i \in V \setminus \{0\}$ are scheduled within the SSGS. A vector $[\vartheta_1, \vartheta_2, \ldots, \vartheta_{n+1}]$ consisting of varying integer numbers, $\vartheta_i \in \{1, \ldots, n+1\}$, implies that the start time S_i of activity i is fixed, after $\vartheta_i - 1$ other activities have been scheduled. For example, vector $[3,4,1,2]$ means that (after setting $S_0 := 0$) activity 3 is scheduled first and then, the start times of activities 4, 1, and 2 are determined consecutively. Since the sequence in which activities are scheduled may vary before and after an unscheduling step, we store the last sequence in the corresponding vector.

The current selected subset consists of a pair of solutions which are represented by vectors V^M (mother) and V^F (father). We obtain a new representation V^C (child) by applying one of the following three crossover operators, which manipulates the vectors of father and mother (see, e.g., Hartmann, 1998; Franck and Selle, 1998). The first crossover operator (co_1) is an one-point crossover, where we randomly draw a number q from set $\{1,\ldots,n\}$. The first q integer numbers of vector V^F are transferred directly to new vector V^C. The remaining $n+1-q$ numbers are taken from vector V^M, starting at the first position of the vector and transferring the missing numbers to V^C. The second and the third operators are two-point crossovers, where two numbers q_1 and q_2 with $q_1 < q_2$ are randomly chosen from set $\{1,\ldots,n\}$. The first two-point operator (co_2) assigns the first q_1 numbers of V^F to new vector V^C, the next $q_2 - q_1$ numbers are taken from V^M, where the transfer of numbers is started at the first position of V^M, and the remaining numbers are again taken from V^F, where the transfer is started at position $q_1 + 1$ of V^F. The second two-point operator (co_3) copies the numbers between positions $q_1 + 1$ and q_2 of vector V^F directly to the same positions of vector V^C. The remaining positions are filled using vector V^M, where the transfer of numbers is started at position $q_2 + 1$. Figure 5 illustrates the application of all crossover operators. Depending on the choice of mother and father representation, child representation V^{C_1} or V^{C2} is generated. Within our scatter search, we consider both child representations.



q_1 q q_2
$V^F = [1, 5, 6, 2, 3, 4]$
$V^M = [2, 3, 5, 6, 4, 1]$
co_1
co_2
co_3
$V^{C_1} = [1,5,2,3,6,4]$
$V^{C_2} = [2,3,1,5,6,4]$
$V^{C_1} = [1,2,3,5,6,4]$
$V^{C_2} = [2,1,5,6,3,4]$
$V^{C_1} = [3,5,6,2,4,1]$
$V^{C_2} = [2,3,5,6,4,1]$

Figure 5: Crossover operators used within the solution combination

In order to evaluate a representation $V^C := [\vartheta_1, \vartheta_2, \ldots, \vartheta_{n+1}]$ and to determine the corresponding schedule S, the SSGS is used again. Thereby, ϑ_i serves as priority-rule value of activity i and an activity with a small priority-rule value is preferred. Thus, activity j to be scheduled next is chosen as $j \in \mathcal{E}$ for which $\pi(j) = \min_{h \in \mathcal{E}} \vartheta_h$. If the SSGS terminates without a feasible solution found, the representation is rejected. Furthermore, in the event that condition $S_{n+1} < 1.1 \cdot S_{n+1}^{best}$ holds for current solution S, the forward-backward procedure is used.

In the *reference set update* (step 5), all feasible solutions found within the combination procedure as well as solutions currently in reference set *RefSet* are considered. The new reference set is built by inserting again b_1 good as well as $b_2 = b - b_1$ disperse solutions. The subset selection, the solution combination, and the reference set update are iterated until $\Lambda \gg 0$ (feasible and infeasible) schedules are determined.

7. Computational results

This section covers the results of a comprehensive performance analysis undertaken in order to investigate the performance of the MIP-formulations as well as the heuristic solution procedures devised in Sections 5 and 6. We start by describing the composition and generation of problem instances used for testing the different methods (cf. Sect. 7.1). CPLEX 12.6 has been used to solve small- and medium-scale instances to optimality. The results of the branch-and-cut methods provided by CPLEX are compared to the results generated by the different heuristic solution procedures (cf. Sect. 7.2). In addition, the impact of our new time planning procedure is shown. All tests have been performed on an Intel Core i7 CPU 990X with 3.47 GHz and 24GB RAM under Windows 7.

7.1. Benchmark instances

The computational tests have been performed on problem-specific instances derived from the UBO-test set. The test set contains 360 instances with 10, 20, 50, and 100 activities as well as 5 renewable resources, respectively (Franck et al., 2001b).

Within a planning horizon of $\overline{d} = 2\sum_{i\in V}\max\{p_i, \max_{\langle i,j\rangle\in A}\{\delta_{ij}\}\}$ time units, three different resource calendars are used: two for workforce or machines requiring personal, and one for machines constantly in execution. The first calendar can be specified by 5–2–5–2 etc. and the second calendar by 6–1–6–1 etc., where work days and days off mentioned alternately. The third calendar contains exclusively work days.

For each problem instance, a resource calendar is assigned randomly to each resource. Then, holidays are incorporated in the first two calendars by switching a workday to a holiday with a probability of $3.\bar{3}\%$. In the event that resource $k \in \mathcal{R}$ obtains the first or the second calendar, we set $\rho_k := 0$, otherwise $\rho_k := 1$ holds. Using the resource calendars, activity calendars may be determined. If calendar $\mathbf{C}_i(\cdot)$ of real activity i does not contain a time $t \in T$ such that $p_i > 0$ working periods follow, activity i is declared as interruptible, i.e., $i \in V^{bi}$. Furthermore, set V^{bi} is filled by choosing the other real activities randomly and consecutively as long as a percentage of $\Psi \in \{60\%, 80\%\}$ is reached (the resulting test set contains 720 RCPSP/max-cal instances). The activities $i \in V^{bi}$ receive a starting phase of $\varepsilon_i \in \{1, 2, \ldots, \lfloor p_i/4\rfloor\}$ time units. Afterwards, all activities that are not yet assigned to set V^{bi} are inserted in set V^{ni}. In order to obtain a well-defined problem instance, additional time lags $d_{i,n+1}^{\min}$ with $\delta_{i,n+1} := p_i$ (if they are missing) are introduced. Moreover, all time lags between activities i

and $n+1$ require resources $\mathcal{R}_{i,n+1} := \mathcal{R}_i$. The remaining time lags, say e.g., the time lag between activities i and j, receive a resource set $\mathcal{R}_{ij}$ randomly from set $\{\emptyset, \mathcal{R}_i, \mathcal{R}_j, \mathcal{R}_i \cup \mathcal{R}_j\}$.

The problem instances are generated using control parameters that influence the behavior of solution procedures (e.g., number of maximum time lags, resource factor, resource strength, and restrictiveness of Thesen; Schwindt, 1998). Particularly, the resource strength (RS), that describes how scarce resources are, affects the run time of our MIP-formulations. The RS-value defines for a specific resource $k \in \mathcal{R}$ its capacity as

$$R_k := \max_{i \in V}\{r_{ik}\} + \lceil RS\ (\max_{t \in [0,\overline{d})}\{r_k(\tilde{ES}, t)\} - \max_{i \in V}\{r_{ik}\})\rceil,$$

where $r_k(\tilde{ES}, \cdot)$ represents the resource profile of schedule $S = \tilde{ES}$; $\tilde{ES}_i$ is the earliest start time of activity i without considering calendar constraints (Kolisch and Sprecher, 1996). Each test set with an equal number of real activities obtains RS-values from set $\{0.1, 0.25, 0.5\}$. In particular, we set $RS := 0.1$ for UBO-instances 1–30, $RS := 0.25$ for UBO-instances 31–60, and $RS := 0.5$ for UBO-instances 61–90.

7.2. Performance study

In our performance study, we initially considered the three MIP-models presented in Section 5, where the temporal inequalities in the start-based and changeover-based models are formulated by disaggregated constraints. Preliminary tests have shown that models with disaggregated constraints are particularly suitable for the calendarization, since, in contrast to models with aggregated constraints, 8 (18) further instances with 50 activities and $\Psi = 60\%$ ($\Psi = 80\%$) could be solved to optimality within our computation environment. Hence, we distinguish between models:

- M_1, start-based model: (7), (8), (10), (11), (14),
- M_2, changeover-based model: (16)–(19), (21)–(23), and
- M_3, execution-based model: (24)–(34).

For every instance, an initial solution has been generated by calling $b^2 = 15^2$ times the multi-start version of the priority-rule method, where the LCT-, the FCS-, and the random-rule are used in equal properties, i.e., the initial population of the scatter search procedure was build. The best solution found was used as a start solution and posted to the CPLEX-solver. Moreover, we have investigated the effectiveness of general CPLEX-cuts during optimization. For all models, mixed-integer rounding (MIR) and generalized upper bound cover (GUB) cuts are added. Since the RCPSP/max-cal is an $\mathcal{NP}$-hard optimization problem, we cannot expect that a branch-and-cut approach will terminate within a reasonable time limitation, which is why we allow a maximum computation time of 3 hours, after which the best solution found up to that point is returned.

Table 5 lists the results for instances with 10 and 20 real activities, where $\Psi = 60\%$ (80%) of activities are interruptible. The different test sets are given by the names $|V^r|$-RS in order to demonstrate the impact of the resource strength. Column "t_{cpu}" describes the average computation times [seconds] and column "opt" ("inf") displays the number of instances solved to proven optimality (infeasibility) within the time limit. Infeasible instances appear if the resource capacity cannot be observed while considering the combination of time lags and calendars. Therefore, it is obvious that a large resource strength results in fewer infeasible solutions. Column "unk" displays the number of instances for which the solvability status is unknown, i.e., neither a feasible schedule is found nor infeasibility is shown.

	$\Psi = 60\%$			$\Psi = 80\%$				$\Psi = 60\%$			$\Psi = 80\%$			
	t_{cpu}	opt	inf	t_{cpu}	opt	inf		t_{cpu}	opt	inf	t_{cpu}	opt	inf	unk
M$_1$														
10-0.1	0.09	21	9	0.09	21	9	20-0.1	2.12	20	10	3.03	19	11	0
10-0.25	0.04	25	5	0.06	24	6	20-0.25	0.71	23	7	0.54	25	5	0
10-0.5	0.03	26	4	0.03	26	4	20-0.5	0.14	28	2	0.13	28	2	0
M$_2$														
10-0.1	0.06	21	9	0.12	21	9	20-0.1	2.18	20	10	4.86	19	11	0
10-0.25	0.04	25	5	0.06	24	6	20-0.25	0.41	23	7	0.46	25	5	0
10-0.5	0.02	26	4	0.02	26	4	20-0.5	0.17	28	2	0.13	28	2	0
M$_3$														
10-0.1	0.17	21	9	0.14	21	9	20-0.1	7.26	20	10	367.25	19	10	1
10-0.25	0.08	25	5	0.13	24	6	20-0.25	3.94	23	7	13.91	25	5	0
10-0.5	0.03	26	4	0.05	26	4	20-0.5	0.45	28	2	0.24	28	2	0

Table 5: Results for MIP-models: instances with 10 and 20 activities

As expected, all model formulations perform very well for instances involving 10 activities. No superiority of a procedure over another can be ascertained. The average run times are invariably less than one second. However, instances with $RS = 0.1$ (i.e., with scarce resources) have a slightly longer run time than instances with $RS = 0.5$ (i.e., with rather high resource capacities). The results for instances with 20 activities show that instances with $\Psi = 80\%$ are harder to solve than instances with $\Psi = 60\%$. The difficulty results from a larger number of feasible activity start times, i.e., $\sum_{i \in V} |W_i|$ is usually larger if 80% of activities are interruptible. In this case, the number of binary decision variables as well as constraints are increased and the solver needs more time for the branching scheme. Furthermore, the execution-based model produces the largest run times, and it proved incapable of terminating the enumerations for all instances; one instance has an unknown solvability status. For instances with the status "unknown", a run time of 3 hours is included in the calculation of t_{cpu}. The increase in run times is mainly evoked by the fact that the setting of a decision variable $z_{it}, i \in V, t \in \mathcal{W}_i$, to one does not produce information on other decision variables $z_{it'}, t' \neq t$. Note that $x_{it} = 1, i \in V, t \in W_i$, leads to $x_{it'} = 0$ for all $t' \neq t$ in the start-based model, and $y_{it} = 1, i \in V, t \in W_i$, leads to $y_{it'} = 1$ for all $t' > t$ in the changeover-based model.

The results for instances with 50 activities are given in Table 6. Column "feas" shows the number of instances for which a feasible schedule is found whose optimality could not be proven. Furthermore, column "gap" describes the mean percentage deviation [%] of the makespan found from the best lower bound. Note that when the optimality of an instance is not proven, t_{cpu} is set to 3 hours. In case of large instances, models M_1 and M_2 work particularly well. In particular, model M_1 produces the best results for instances with $RS = 0.1$ and $\Psi = 60\%$. Considering instances with $RS \in \{0.1, 0.25\}$ and $\Psi = 80\%$, M_1 is able to solve two more instances to proven optimality than M_2, but two instances remain with an unknown solvability status. For the other instances, model M_2 produces the lowest run times and seems to be the best choice. Model M_3 yields again the worst performance. The average solution gaps of feasible solved instances are 29.4% for the start-based, 15.9% for the changeover-based, and 19.1% for the execution-based model.

	$\Psi = 60\%$						$\Psi = 80\%$					
	t_{cpu}	opt	inf	unk	feas	gap	t_{cpu}	opt	inf	unk	feas	gap
M_1												
50-0.1	1946.43	20	6	2	2	6.98	3741.13	19	4	1	6	36.84
50-0.25	507.78	26	4	0	0	0.00	545.50	26	3	1	0	0.00
50-0.5	0.83	30	0	0	0	0.00	1.53	30	0	0	0	0.00
M_2												
50-0.1	2312.43	20	6	2	2	8.95	3672.89	18	5	0	7	20.95
50-0.25	292.42	26	4	0	0	0.00	875.33	25	3	0	2	5.08
50-0.5	0.55	30	0	0	0	0.00	0.78	30	0	0	0	0.00
M_3												
50-0.1	6347.37	13	1	9	7	17.64	7703.66	10	1	9	10	20.08
50-0.25	2534.73	23	1	6	0	0.00	2464.28	24	1	5	0	0.00
50-0.5	3.07	30	0	0	0	0.00	4.63	30	0	0	0	0.00

Table 6: Results for MIP-models: instances with 50 activities

Before we present the outcomes of heuristic solution procedures, the advantageous of our time planning procedure should be emphasized. All model formulations benefit from temporal planning significantly. In what follows, we focus on the start-based formulation. A start-based MIP-model that considers each point in time $t \in U_i := \{ES_i, \ldots, LS_i\}$ as potential start time of activity $i \in V$ (and not set $W_i \subseteq \{ES_i, \ldots, LS_i\}$) can be formulated by

$$\text{Minimize} \quad \sum_{t \in U_{n+1}} t \, x_{n+1,t} \tag{35}$$

subject to

$$\sum_{t \in U_i} x_{it} = 1 \qquad i \in V \tag{36}$$

$$\sum_{t \in U_i} x_{it} \sum_{\tau=t}^{t+\varepsilon_i-1} \mathbf{C}_i(\tau) = \varepsilon_i \qquad i \in V^r \tag{37}$$

$$\sum_{t \in U_i} x_{it} \sum_{\tau=t}^{\overline{d}} \mathbf{C}_{ij}(\tau) - \sum_{t \in U_j} x_{jt} \sum_{\tau=t}^{\overline{d}} \mathbf{C}_{ij}(\tau) \geq \delta_{ij} \qquad \langle i,j \rangle \in A \tag{38}$$

$$\sum_{i \in V^r} r_{ik} \sum_{\tau=\max(ES_i,\beta_{it}+1)}^{\min(LS_i,t)} x_{i\tau}\, C_{itk} \leq R_k \qquad k \in \mathcal{R}, t \in T' \tag{39}$$

$$x_{it} \in \{0,1\} \qquad i \in V, t \in U_i. \tag{40}$$

Binary decision variable x_{it}, $i \in V, t \in \{ES_i, \dots, LS_i\}$, is equal to 1 if activity i starts at time t, and 0 otherwise. Equalities (36) guarantee that each activity receives precisely one start time. Constraints (37) ensure that non-interruptible activities are carried out without interruption and interruptible activities observe the start-up phase. Inequalities (38) and (39) make sure that the given temporal as well as resource constraints are satisfied.

Table 7 shows the results for instances with 50 activities obtained by CPLEX using model M$_4$: (35)–(40). In contrast to the start-based model M$_1$, where the results of the time planning procedure are used, the model M$_4$ solves fewer instances to optimality within 3 hours. Additionally, the average run times are very high due to the larger number of binary decision variables and constraints; the model incorporates $\sum_{i \in V} |U_i|$ variables as well as $\mathcal{O}(|V|^2 + \sum_{i \in V} |U_i| + |\mathcal{R}||T'|)$ constraints. In particular, M$_1$ is able to solve 23 more instances to proven optimality and for 11 more instances the infeasibility could be shown.

	$\Psi = 60\%$						$\Psi = 80\%$					
	t_{cpu}	opt	inf	unk	feas	gap	t_{cpu}	opt	inf	unk	feas	gap
M$_4$												
50-0.1	6929.06	13	0	10	7	20.20	7525.06	9	2	8	11	13.46
50-0.25	1580.51	23	3	1	3	19.29	2455.28	23	1	4	2	2.75
50-0.5	0.52	30	0	0	0	0.00	0.99	30	0	0	0	0.00

Table 7: Results for model M$_4$: instances with 50 activities

The main differences between M$_4$ and M$_1$ are the underlying sets of possible start times, U_i and $W_i \subseteq U_i$, for activities $i \in V$. The time planning procedure reduces U_i in two steps (starting from the initialization), so that W_i is obtained. In the first step, the start-up phases of activities are considered, and in the second step, the length of the longest path between all activities are computed using the triple algorithm (cf. Algorithm 1). Table 8 depicts the average sizes of sets U_i and W_i in the course of the reduction. Thereby, the results are specified for all test sets with 10, 20, 50, and 100 real activities. After the first and the second step, the average reductions [%] with respect to the initial values are given in brackets.

activities	initialization $\lvert U_i\rvert, i \in V$	start-up phase		triple algorithm $\lvert W_i\rvert, i \in V$	
10	70.73	47.34	(33.1%)	46.28	(34.6%)
20	142.76	93.49	(34.5%)	92.10	(35.5%)
50	381.49	257.88	(32.4%)	254.69	(33.2%)
100	762.33	514.01	(32.6%)	509.20	(33.2%)

Table 8: Average sizes of sets U_i and $W_i, i \in V$

The average size of sets $U_i, i \in V$, can be decreased by 33.1% through the start-up phase and by 34.1% through the combination of start-up phase and triple algorithm. Therefore, it makes sense to include only relevant decision variables and constraints in a model and omit the rest of irrelevant variables/constraints.

Finally, we tried to solve instances with 100 activities and model M_1 (because M_1 seems to be most efficient). Unfortunately, the time needed by CPLEX to build the model in memory was very long and the model could not be adequately handled within branch-and-cut. However, the results obtained by CPLEX are used for comparison purposes in Table 11.

In order to solve large-scale problem instances with up to 100 activities, the described scatter search procedure can be used. We limited the number of generated schedules in order to obtain a stopping criterion that is independent of the computer platform. Therefore, we generated $\Lambda \in \{1000, 5000\}$ schedules within our scatter search. Furthermore, we set $b = 15$, $b_1 = 10$, and $b_2 = 5$. When dealing with stochastic optimization methods, like scatter search, it is interesting to analyze the efficiency of incorporated mechanisms (i.e., the choice of reference set and crossover operators), which hopefully let the algorithm generate near-optimal solutions. For that reason, we compared the results of the scatter search (SS) with a pure random search (R). Within the pure random search, we generated Λ schedules with the multi-start algorithm and the random priority rule. Table 9 summarizes the results, where the test sets are denoted by the names $|V^r|$-Ψ-Λ. As the resource strength does not influence the results of the scatter search procedure, a further distinction of RS-values is not necessary. Column "feas$^{\text{SS}}$" ("feas$^{\text{R}}$") shows the number of instances for which a feasible solution could be found and column "t_{cpu}^{SS}" ("t_{cpu}^{R}") depicts the average computation times for scatter search (random search). In column "SS=R" ("SS<R") the number of scatter search solutions with the same (a smaller) objective function value as generated by the random search is given and column "$\delta_<$" shows the respective average negative deviations [%]. Note that the summation of values in columns "SS=R" and "SS<R" is always 90, since infeasible instances

are also incorporated. If both methods terminate with a status "no solution found", a one is added to the value in column "SS=R". If scatter search stops with a feasible solution and random search with no solution, a one is added to the value in column "SS<R", but the calculation of "$\delta_<$" is not influenced.

	feas^{SS}	feas^{R}	t^{SS}_{cpu}	t^{R}_{cpu}	SS=R	SS<R	$\delta_<$
10-60-1000	72	72	0.06	0.04	89	1	-6.25
10-80-1000	71	70	0.06	0.04	89	1	—
20-60-1000	71	70	0.76	0.48	82	8	-6.31
20-80-1000	72	71	0.73	0.49	80	10	-3.34
50-60-1000	75	67	17.98	10.86	49	41	-5.46
50-80-1000	74	66	17.15	11.34	52	38	-6.47
100-60-1000	63	40	188.84	136.71	47	43	-3.52
100-80-1000	62	44	179.95	129.44	49	41	-4.04
10-60-5000	72	72	0.32	0.18	90	0	—
10-80-5000	71	71	0.31	0.18	90	0	—
20-60-5000	71	70	3.79	2.37	82	8	-3.95
20-80-5000	72	72	3.57	2.44	88	2	-2.30
50-60-5000	76	70	87.44	54.18	55	35	-4.38
50-80-5000	77	69	84.43	56.60	50	40	-5.34
100-60-5000	66	44	911.98	683.86	49	41	-4.40
100-80-5000	69	46	884.85	647.49	45	45	-4.35

Table 9: Comparison of random search and scatter search

A comparison of the second and third column shows that the scatter search procedure produces always the same or a larger number of feasible solutions. Furthermore, the average negative deviation, which is equal to 4.7%, demonstrate the good performance of our scatter search procedure. As expected, the average run times of scatter search (that performs steps to combine solutions and to update the reference set) are larger than the average run times of random search. In contrast to the results presented in Tables 5 and 6, instances with $\Psi = 60\%$ seems to be harder to solve than instances with $\Psi = 80\%$, because the computation times are slightly larger.

In order to compare the results of scatter search with a stopping criterion of 1000 schedules (SS_{1000}) and a stopping criterion of 5000 schedules (SS_{5000}) in more detail, Table 10 can be applied. In column "$\text{feas}^{\text{SS}_{1000}}$" ("$\text{feas}^{\text{SS}_{5000}}$") the number of feasible solutions after 1000 (5000) schedules is given. Moreover, columns "SS_{5000}=SS_{1000}" and "SS_{5000} <SS_{1000}" demonstrate the number of instances for which the objective function values are equal or for which the objective function values are smaller after 5000 schedules have been build. Column "$\delta_<$" again depicts the average negative deviations [%].

The results indicate that a numerous schedule generation within scatter search leads to better results. The number of feasible solutions in SS_{5000} is

	$\text{feas}^{\text{SS}_{1000}}$	$\text{feas}^{\text{SS}_{5000}}$	SS_{5000}=SS_{1000}	SS_{5000} <SS_{1000}	$\delta_<$
10-60	72	72	90	0	0.00
10-80	71	71	88	2	-9.21
20-60	71	71	84	6	-4.29
20-80	72	72	88	2	-2.75
50-60	75	76	72	18	-2.46
50-80	74	77	67	23	-2.77
100-60	63	66	74	16	-3.69
100-80	62	69	60	30	-2.98

Table 10: Comparison of scatter search solutions after 1000 and 5000 schedules

higher than in SS_{1000} and the solution quality is improved by about 3.5%.

Finally, we compared each solution obtained by SS_{5000} to the best solution obtained by one of the described MIP-models (with a computation time limit of 3 hours). The results are given in Table 11, where in column "$\delta_>$" the average positive deviations [%] are integrated, if the scatter search produces a larger objective function value than the MIP-model, i.e., SS>MIP.

	feas^{SS}	feas^{MIP}	SS=MIP	SS<MIP	$\delta_<$	SS>MIP	$\delta_>$
10-60	72	72	88	0	0	2	1.94
10-80	71	71	90	0	0	0	0.00
20-60	71	71	84	0	0	6	3.81
20-80	72	72	86	0	0	4	3.23
50-60	76	78	53	1	0	36	6.24
50-80	77	81	46	2	-23.89	42	5.54
100-60	66	64	49	12	-29.69	29	5.21
100-80	69	59	51	20	-3.61	19	6.24

Table 11: Comparison of scatter search with 5000 schedules and MIP

For nearly all small-scale instances with 10 and 20 activities, the optimal solution could be found by scatter search. The majority of instances with 50 and 100 activities reach the same objective function value with both solution procedures. Furthermore, the average negative deviation of scatter search in comparison to MIP-models is 14.0% and the average positive deviation is 5.8%. For instances with 100 activities, scatter search finds significantly more feasible solutions than the MIP-models. If more schedules are generated, e.g., 20,000 schedules (average run time is one hour) instead of 5000 schedules, the results can be improved, i.e., six more feasible solutions for instances with 100 activities are reached in our test runs. Therefore, scatter search seems to be an appropriate metaheuristic for the problem under consideration.

8. Conclusions

In this paper, the resource-constrained project scheduling problem subject to general temporal as well as calendar constraints is treated. In practice, calendars have to be taken into account in order to incorporate that resources, like workers or tools, are not available during breaks, e.g., weekends or holidays. The problem is motivated by several real-life conditions that require a modeling through calendarization. After describing the problem formally, we presented a novel time planning procedure that finds the feasible start times of activities and the longest paths between start times of activities (absolute minimum time lags). The procedure is used in MIP-formulations and solution algorithms. In order to solve the RCPSP/max-cal, either exactly or heuristically, three binary linear model formulations, a priority-rule method, as well as a scatter search algorithm are presented.

The results of our performance analysis show that many decision variables in MIP-models can be eliminated and run times can be reduced if the time planning procedure is executed in a preprocessing step. In particular, both the start-based and the changeover-based model perform very well; they are able to solve most of the hard instances with 50 activities and $RS = 0.1$. The execution-based model is found to be of little value as alternative solution approach. In addition, we introduced a priority-rule method that essentially improves existing priority-rule methods for the RCPSP/max-cal. Existing methods are based on determining the earliest and latest start times for all activities not yet scheduled in each iteration. In contrast, our priority-rule method uses the longest path lengths between activities, which are determined in the time planning procedure, in order to update earliest and latest start times. The priority-rule method is extended to a multi-start procedure. Furthermore, a scatter search algorithm is developed. Computational results reveal that scatter search is a good choice to solve the problem under consideration, since it finds optimal solution for nearly all small instances (10 and 20 activities, respectively), it obtains good results in comparison to MIP-models, and it performs much better than a random search procedure.

We believe that the recovery of the RCPSP/max-cal and the introduction of new problem instances can be a starting point for other researchers designing exact and heuristic solution procedures. The best known exact solution method for the RCPSP/max is based on lazy clause generation, which combines constraint programming and boolean satisfiability solving (Schutt et al., 2013, 2014). Therefore, it seems to be promising to consider lazy clause generation in order to solve RCPSP/max-cal instances. Finally, the study of different objectives like resource leveling or resource investment in combination with calendarization is an interesting topic.

Acknowledgements

The benchmarks for the RCPSP/max-cal presented herein may be downloaded from http://www.wiwi.tu-clausthal.de/abteilungen/unternehmensforschung/forschung/.

References

Ahuja, R., Magnanti, T., Orlin, J., 1993. Network Flows. Prentice Hall, Englewood Cliffs.

Alba, E., Chicano, J. F., 2007. Software project management with gas. Information Sciences 177, 2380–2401.

Alvarez-Valdés, R., Crespo, E., Tamarit, J. M., Villa, F., 2006. A scatter search algorithm for project scheduling under partially renewable resources. Journal of Heuristics 12, 95–113.

Alvarez-Valdés, R., Tamarit, J. M., 1993. The project scheduling polyhedron: Dimension, facets and lifting theorems. European Journal of Operational Research 67, 204–220.

Artigues, C., Brucker, P., Knust, S., Koné, O., Lopez, P., Mongeau, M., 2013. A note on event-based MILP models for resource-constrained project scheduling problems. Computers & Operations Research 40(4), 1060–1063.

Artigues, C., Koné, O., Lopez, P., Mongeau, M., 2014. Mixed-integer linear programming formulations. In: Schwindt, C., Zimmermann, J. (Eds.), Handbook on Project Management and Scheduling, Vol. 1. Springer, New York, Ch. 2.

Artigues, C., Michelon, P., Reusser, S., 2003. Insertion techniques for static and dynamic resource-constrained project scheduling. European Journal of Operational Research 149(2), 249–267.

Bomsdorf, F., Derigs, U., 2008. A model, heuristic procedure and decision support system for solving the movie shoot scheduling problem. OR Spectrum 30, 751–772.

Brucker, P., Knust, S., 2012. Complex Scheduling, 2nd Edition. Springer, Berlin.

Budai, G., Dekker, R., Nicolai, R., 2008. Maintenance and production: a review of planning models. In: Kobbacy, K., Murthy, D. (Eds.), Complex Systems Maintenance Handbook, Springer Series in Reliability Engineering. Springer, Berlin, pp. 321–344.

Buddhakulsomsiri, J., Kim, D. S., 2006. Properties of multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. European Journal of Operational Research 175, 279–295.

Buddhakulsomsiri, J., Kim, D. S., 2007. Priority rule-based heuristic for multi-mode resource-constrained project scheduling problems with resource vacations and activity splitting. European Journal of Operational Research 178, 374–390.

Cheng, J., Fowler, J., Kempf, K., Mason, S., 2014. Multi-mode resource-constrained project scheduling problems with non-preemptive activity splitting. Computers & Operations Research, http://dx.doi.org/10.1016/j.cor.2014.04.018.

Christofides, N., Alvarez-Valdés, R., Tamarit, J. M., 1987. Project scheduling with resource constraints: A branch and bound approach. European Journal of Operational Research 29(3), 262–273.

Coviello, D., Ichino, A., Persico, N., 2010. Don't spread yourself too thin: The impact of task juggling on workers' speed of job completion. NBER Working Paper 16502, Cambridge, MA.

Debels, D., De Reyck, B., Leus, R., Vanhoucke, M., 2006. A hybrid scatter search/electromagnetism meta-heuristic for project scheduling. European Journal of Operational Research 169, 638–653.

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

Franck, B., Neumann, K., Schwindt, C., 2001a. Project scheduling with calendars. OR Spektrum 23, 325–334.

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

Franck, B., Selle, T., 1998. Metaheuristics for the resource-constrained project scheduling with schedule-dependent time windows. WIOR-Report 546, Institute for Economic Theory and Operations Research, University Karlsruhe, Germany.

Glover, F., 1977. Heuristics for integer programming using surrogate constraints. Decision Sciences 8, 156–166.

Hartmann, S., 1998. A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics 45, 733–750.

Hartmann, S., 2014. Time-varying resource requirements and capacities. In: Schwindt, C., Zimmermann, J. (Eds.), Handbook on Project Management and Scheduling, Vol. 1. Springer, Berlin, Ch. 8.

Hartmann, S., Briskorn, D., 2010. A survey of variants and extensions of the resource-constrained project scheduling problem. European Journal of Operational Research 207, 1–14.

Józefowska, J., Węglarz, J., 2006. Perspectives in Modern Project Scheduling. Springer, New York.

Kaplan, L., 1988. Resource-constrained project scheduling with preemption of jobs. Unpublished PhD dissertation, University of Michigan, Michigan, USA.

Kelly, J. E., 1963. Critical-path method: Resource planning and scheduling. In: Muth, J. F., Thompson, G. L. (Eds.), Industrial Scheduling. Prentice Hall, Englewood Cliffs, pp. 347–365.

Klein, R., 2000a. Project scheduling with time-varying resource constraints. International Journal of Production Research 38, 3937–3952.

Klein, R., 2000b. Scheduling of Resource-Constrained Projects. Kluwer, Boston.

Kolisch, R., 1995. Project Scheduling under Resource Constraints: Efficient Heuristics for Several Problem Classes. Physica, Heidelberg.

Kolisch, R., Hartmann, S., 2006. Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research 174, 23–37.

Kolisch, R., Sprecher, A., 1996. PSPLIB – a project scheduling problem library. European Journal of Operational Research 96, 205–216.

Koné, O., Artigues, C., Lopez, P., Mongeau, M., 2011. Event-based MILP models for resource-constrained project scheduling problems. Computers & Operations Research 38, 3–13.

Laguna, M., Martí, R., 2003. Scatter Search: Methodology and Implementations. Kluwer Academic Publishers, Boston.

Mahdi Mobini, M. D., Rabbani, M., Amalnik, M. S., Razmi, J., Rahimi-Vahed, A. R., 2009. Using an enhanced scatter search algorithm for a resource-constrained project scheduling problem. Soft Computing 13, 597–610.

Mingozzi, A., Maniezzo, V., Ricciardelli, S., Bianco, L., 1998. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. Management Science 44, 714–729.

Neumann, K., Morlock, M., 2002. Operations Research, 2nd Edition. Carl Hanser, München.

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., Zimmermann, J., 2003a. Order-based neighborhoods for project scheduling with nonregular objective functions. European Journal of Operational Research 149, 325–343.

Neumann, K., Schwindt, C., Zimmermann, J., 2003b. Project Scheduling with Time Windows and Scarce Resources, 2nd Edition. Springer, Berlin.

Pritsker, A. A. B., Watters, L. J., Wolfe, P. M., 1969. Multi-project scheduling with limited resources: A zero-one programming approach. Management Science 16, 93–108.

Ranjbar, M., De Reyck, B., Kianfar, R., 2009. A hybrid scatter search for the discrete time/resource trade-off problem in project scheduling. European Journal of Operational Research 193, 35–48.

Sankaran, J. K., Bricker, D. L., Juang, S.-H., 1999. A strong fractional cutting-plane algorithm for resource-constrained project scheduling. International Journal of Industrial Engineering 6(2), 99–111.

Schutt, F., Feydy, T., Stuckey, P., Wallace, M., 2013. Solving RCPSP/max by lazy clause generation. Journal of Scheduling 16(3), 273–289.

Schutt, F., Feydy, T., Stuckey, P., Wallace, M., 2014. A satisfiability solving approach. In: Schwindt, C., Zimmermann, J. (Eds.), Handbook on Project Management and Scheduling, Vol. 1. Springer, New York, Ch. 7.

Schwindt, C., 1998. Generation of resource-constrained project scheduling problems subject to temporal constraints. WIOR-Report 543, Institute for Economic Theory and Operations Research, University Karlsruhe, Germany.

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

Talbot, F. B., Patterson, J. H., 1978. An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. Management Science 24, 1163–1174.

Tormos, P., Lova, A., 2001. A competitive heuristic solution technique for resource-constrained project scheduling. Annals of Operations Research 102, 65–81.

Trautmann, N., 2001. Calendars in project scheduling. In: Fleischmann, B., Lasch, R., Derigs, U., Domschke, W., Rieder, U. (Eds.), Operations Research Proceedings 2000. Springer, Berlin, pp. 388–392.

Valls, V., Ballestin, F., Quintanilla, M. S., 2005. Justification and rcpsp: A technique that pays. European Journal of Operational Research 165, 375–386.

Valls, V., Pérez, A., Quintanilla, S., 2009. Skilled workforce scheduling in service centres. European Journal of Operational Research 193, 791–804.

Vanhoucke, M., 2010. A scatter search heuristic for maximising the net present value of a resource-constrained project with fixed activity cash flows. International Journal of Production Research 48(7), 1983–2001.

Zapata, J. C., Hodge, B. M., Reklaitis, G. V., 2008. The multimode resource constrained multiproject scheduling problem: Alternative formulations. AIChE Journal 54(8), 2101–2119.

Zhan, J., 1988. Kalendrierung der Terminplanung in MPM-Netzplänen. WIOR-Report 334, Institute for Economic Theory and Operations Research, University Karlsruhe, Germany.

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