European Journal of Operational Research 221 (2012) 27–37

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

Mixed-integer linear programming for resource leveling problems

Julia Rieck [*], Jürgen Zimmermann, Thorsten Gather

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

ARTICLE INFO

Article history:
Received 7 July 2011
Accepted 4 March 2012
Available online 16 March 2012

Keywords:
Project scheduling
Resource leveling
Minimum and maximum time lags
Lower and upper bounds
Cutting planes

ABSTRACT

We consider project scheduling problems subject to general temporal constraints, where the utilization of a set of renewable resources has to be smoothed over a prescribed planning horizon. In particular, we consider the classical resource leveling problem, where the variation in resource utilization during project execution is to be minimized, and the so-called "overload problem", where costs are incurred if a given resource-utilization threshold is exceeded. For both problems, we present new mixed-integer linear model formulations and domain-reducing preprocessing techniques. In order to strengthen the models, lower and upper bounds for resource requirements at particular points in time, as well as effective cutting planes, are outlined. We use CPLEX 12.1 to solve medium-scale instances, as well as instances of the well-known test set devised by Kolisch et al. (1999). Instances with up to 50 activities and tight project deadlines are solved to optimality for the first time.

1. Introduction

A project is a unique and temporary endeavor that can be subdivided into various activities that require time and renewable resources, such as machines, equipment, or manpower, for their execution. Usually, projects involve general temporal constraints among activities resulting from technological or organizational restrictions. Project scheduling consists of determining start times for all activities such that temporal and/or resource constraints are satisfied and some objective is optimized (see e.g. Józefowska and Węglarz, 2006).

Resource leveling problems (RLPs) arise whenever it is expedient to reduce the fluctuations in patterns of resource utilizations over time, while maintaining compliance with a prescribed project completion time. In particular, in cases where even slight variations in resource needs represent financial burden or heightened risks of accidents, a resource leveling approach helps to schedule the project activities such that the resource utilization will be as smooth as possible over the entire planning horizon (cf. Demeulemeester and Herroelen, 2002). Under resource leveling, no resource limits are typically imposed. Therefore, only the time lags between individual activities form the project constraints.

Resource leveling has received little attention in the academic literature. A bunch of instances with 30 activities (cf. Kolisch et al., 1999) remain to be solved optimally. In order to compensate for that dearth of research, we consider exact methods for the "classical resource leveling problem", where variations in resource utilizations within the project duration are to be minimized (cf. Burgess and Killebrew, 1962). In addition, we study the "overload problem", where costs are incurred if either a given supply of some renewable resources or a threshold for the resource utilization is exceeded (cf. Easa, 1989). New mixed-integer linear models and domain-reducing preprocessing techniques are devised for both problems. We obtain promising results on the well-known test instances of Kolisch et al. (1999) using CPLEX 12.1. For the first time, all problem instances with 30 activities are solved to optimality with respect to the minimum project duration.

In Section 2, we formally describe the resource leveling problem using two different objective functions and present its mathematical background. In Section 3, we investigate an interesting application of resource leveling that substantiates both the objective functions we have proposed and the structuring of the problem instances we have used in our experimental performance analysis. Section 4 is devoted to a literature review on exact solution methods for resource leveling, where we sketch the most common approaches and present known mathematical model formulations. Based on those models, we proceed to describe methods for linearizing the corresponding objective functions and improving the quality of the resulting formulations in terms of computation time and solution gap (cf. Section 5). The results of a comprehensive performance analysis are given in Section 6. Finally, conclusions are presented in Section 7.

2. Problem description

In the remainder of this paper, we consider projects specified by activity-on-node networks $N = (V, A; \delta)$, where V is the set of vertices and A is the set of arcs with weight δ. Vertex set

[*] Corresponding author.
E-mail addresses: julia.rieck@tu-clausthal.de (J. Rieck), juergen.zimmermann@tu-clausthal.de (J. Zimmermann).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ejor.2012.03.003

$V := \{0, 1, \ldots, n, n+1\}$ consists of $n \geqslant 1$ activities, $1, \ldots, n$, that have to be carried out without interruption, and two fictitious activities, 0 and $n+1$, that represent the beginning and completion of the underlying project, respectively. Each activity has to be started no earlier than the project beginning, and must be completed by project termination.

We denote the start time of activity $i \in V$ by S_i and assume that every project begins at time zero, i.e. $S_0 := 0$. Then, S_{n+1} equals the project duration. If activity j cannot be started earlier than $d_{ij}^{min} \in \mathbb{Z}_{\geqslant 0}$ time units after activity i (minimum time lag), i.e. $S_j - S_i \geqslant d_{ij}^{min}$, we introduce an arc $\langle i, j \rangle$ having weight $\delta_{ij} := d_{ij}^{min}$ into network N. In the event that activity j can be begun as soon as activity i has been concluded, i.e. $d_{ij}^{min} = p_i$, the minimum time lag is referred to as a "precedence constraint". If activity j must be started no later than $d_{ij}^{max} \in \mathbb{Z}_{\geqslant 0}$ time units after activity i (maximum time lag), i.e. $S_j - S_i \leqslant d_{ij}^{max}$, we introduce a backward arc $\langle j, i \rangle$ with weight $\delta_{ji} := -d_{ij}^{max}$. The resulting arc set A contains at most $|V||V-1|$ arcs representing the temporal constraints $S_j - S_i \geqslant \delta_{ij}$ among the start times of activities $i, j \in V$.

A sequence of start times $S = (S_0, S_1, \ldots, S_{n+1})$, where $S_i \geqslant 0$, $i \in V$, and $S_0 = 0$, is termed a "schedule". A schedule is said to be feasible if it satisfies all temporal constraints of the project given by minimum and maximum time lags. The set of all feasible schedules is denoted by $\mathcal{S}_T$. Let $\bar{d} \geqslant 0$ be a prescribed maximum project duration (i.e. a project-completion deadline). The problem of finding an optimal (feasible) schedule for some objective function $f : \mathbb{R}_{\geqslant 0}^{n+2} \to \mathbb{R}$ to be minimized may be formulated as follows:

$$\left.\begin{array}{ll} \text{Minimize} & f(S) \\ \text{subject to} & S_j - S_i \geqslant \delta_{ij} \quad \langle i,j \rangle \in A \\ & S_0 = 0 \\ & S_{n+1} \leqslant \bar{d} \\ & S_i \geqslant 0 \quad i \in V. \end{array}\right\} \tag{1}$$

For an activity $i \in V$, inequality $S_i \geqslant 0$ is already implied by $S_0 := 0$ and the assumption that no activity can be started prior to the project beginning. Furthermore, since we assume in the following that network N contains an arc $\langle n+1, 0 \rangle$ having weight $\delta_{n+1,0} := -\bar{d}$ in order to ensure compliance with the prescribed project deadline, inequality $S_{n+1} \leqslant \bar{d}$ also becomes redundant. As has been shown by Bartusch et al. (1988), the feasible region $\mathcal{S}_T$ of problem (1) is non-empty, iff N contains no cycle of positive length, which can be checked in polynomial time (cf. Ahuja et al., 1993, Section 5.5).

Due to the prescribed project deadline, the set of feasible start times of activity $i \in V$ forms a proper time window $[ES_i, LS_i]$, where ES_i is the earliest and LS_i the latest start time of activity i with respect to the given temporal constraints. By definition, $ES_0 = LS_0 := 0$. For a specified activity $i \in V \backslash \{0\}$, both the earliest start time, ES_i, which equals the length of a longest path from node 0 to node i, and the latest start time, LS_i, which equals the negative of the longest path length from node i to node 0, can be determined by applying some label-correcting algorithm (see e.g. Ahuja et al., 1993, Section 5.4). The total float, $TF_i := LS_i - ES_i$, $i \in V$, is the maximum length of time, by which the start of activity i may be delayed beyond its earliest start time, without causing project completion to be delayed beyond the final deadline given by $\bar{d}$. An activity i is termed "critical" if a delay in its start will cause a delay in completing the entire project. The total float is therefore zero for critical activities, and has some positive value for non-critical activities.

Let $\mathcal{R}$ be the set of renewable resources required for carrying out the project activities. Every activity $i \in V$ has a given processing time $p_i \in \mathbb{Z}_{\geqslant 0}$, and requires $r_{ik} \in \mathbb{Z}_{\geqslant 0}$ units of resource $k \in \mathcal{R}$ taken up by processing activity i, commencing with its start time S_i (inclusively), through to its completion time $S_i + p_i$ (exclusively). An activity i is referred to as "event" if $p_i = 0$; otherwise, it is regarded as a real activity. Every real activity i is presumed to be performed during the half-open time interval $[S_i, S_i + p_i[$. In case of the fictitious activities, we set $p_0 = p_{n+1} := 0$ and $r_{0k} = r_{n+1,k} := 0$ for all $k \in \mathcal{R}$. Given some schedule S, the set of (real) activities in progress at time t, which is also termed the "active set", is given by $\mathcal{A}(S,t) := \{i \in V | S_i \leqslant t < S_i + p_i\}$. Thus, $r_k(S,t) := \sum_{i \in \mathcal{A}(S,t)} r_{ik}$ represents the total amount of resource $k \in \mathcal{R}$ required for those activities in progress at time t. The resource profiles $r_k(S, \cdot) : [0, \bar{d}] \to \mathbb{R}_{\geqslant 0}$ are step functions continuous from the right at their jump points.

If the resources necessary to carry out the activities involved should be distributed evenly over the time horizon, we speak of *resource leveling*. Different objective functions are considered in the literature (see e.g. Neumann and Zimmermann, 1999, 2000), depending on how variations in resource utilizations are measured. In what follows, we consider two resource leveling functions having broad areas of application.

In practice, companies often want to realize smooth resource profiles for a given project duration, and aim at penalizing high resource utilizations more than low resource utilizations. Let $c_k \geqslant 0$ be the cost incurred per unit of resource $k \in \mathcal{R}$, and per time unit. The "classical resource leveling objective function" will then be given by

$$f(S) := \sum_{k \in \mathcal{R}} c_k \int_{t \in [0,\bar{d}]} r_k^2(S,t)\, dt. \tag{RL1}$$

(RL1) represents the total squared utilization cost for a given schedule S (cf. Burgess and Killebrew, 1962; Harris, 1990). A possible application can be found in make-to-order manufacturing operations, where an even-workload distribution of resources is required (cf. Ballestin et al., 2007). Moreover, (RL1) may be used for avoiding large deviations from prescribed resource-utilization thresholds $Y_k, k \in \mathcal{R}$, since the conditions

$$\begin{aligned} &\sum_{k \in \mathcal{R}} c_k \int_{t \in [0,\bar{d}]} (r_k(S,t) - Y_k)^2 dt \\ &= \sum_{k \in \mathcal{R}} c_k \left(\int_{t \in [0,\bar{d}]} r_k(S,t)^2 dt - 2\, Y_k \sum_{i \in V} r_{ik}\, p_i + \bar{d}\, Y_k^2 \right) \\ &= \sum_{k \in \mathcal{R}} c_k \int_{t \in [0,\bar{d}]} r_k^2(S,t) dt + K \end{aligned}$$

are satisfied with some $K \in \mathbb{R}$.

Employers are usually required to pay overtime premiums to employees who work more than the standard hours. Additional costs for covering the positive deviations from the desired resource utilizations $Y_k, k \in \mathcal{R}$, will therefore be incurred (cf. Easa, 1989; Bandelloni et al., 1994). In order to take this option into account, we consider the "total overload cost function"

$$f(S) := \sum_{k \in \mathcal{R}} c_k \int_{t \in [0,\bar{d}]} (r_k(S,t) - Y_k)^+ dt. \tag{RL2}$$

In case no thresholds Y_k, e.g. the standard weekly hours, have been prescribed, Y_k may be chosen equal to the (rounded) average resource utilizations, i.e. $Y_k := \sum_{i \in V} \lceil r_{ik}\, p_i / \bar{d} \rceil$.

If time t is discrete, the integrals appearing in (RL1) and (RL2) could be replaced by summations. As has been shown by Neumann et al. (2003), problem (1) is $\mathcal{NP}$-hard in the strong sense in case of both resource leveling variants. However, both objective functions and the set of feasible solutions have nice properties that can be exploited along the way to an optimal solution. Firstly, the feasible region represents a convex polytope of dimension $n+1$ if network N contains neither any redundant time lags nor cycles of length zero. An algorithm for eliminating redundant arcs may be found in Habib et al. (1993) or Gather et al. (2011). The activities of some cycle of length zero will be strictly interlinked and may be replaced by a single node (cf. Neumann et al., 2003). Furthermore, every binding temporal constraint, $S_j = S_i + \delta_{ij}$, $i, j \in V$, defines a facet of the feasible region (cf. Hagmayer, 2006). Secondly, objective functions (RL1) and (RL2) are r-monotone, i.e. the condition $f(S) \leqslant f(S')$ is implicitly specified for (partial) schedules S and S', where $r_k(S, t) \leqslant r_k(S', t)$ for all $k \in \mathcal{R}$ and $t \in [0, \bar{d}]$. Moreover, since both functions are continuous and locally concave, there invariably exists always a quasistable schedule that will be optimal for the problem under consideration (cf. Neumann et al., 2000).

Neumann et al. (2003) have shown that every quasistable schedule marks an extreme point of some order polytope and may be represented by a spanning tree of the corresponding order network. As a result, there is an optimal schedule for problem (1) in conjunction with (RL1) or (RL2) that may be defined by a spanning tree of some order network, where every arc represents a binding precedence or temporal constraint. Since we assume that the input data, p_i and δ_{ij}, $i, j \in V$, consists of integers and that $S_0 = 0$, every quasistable schedule will be integer-valued.

In order to illustrate the construction of a project network $N = (V, A; \delta)$, and demonstrate the different properties as well as the non-equivalence of the two proposed objective functions, we consider a project with five real activities and two renewable resources (cf. Fig. 1). Let $c_1 := 3$ and $c_2 := 1$ be the respective costs per unit of resources 1 and 2, and per time unit. In this case, an optimal solution for the first model may be represented by $S^* = (0, 2, 4, 1, 0, 3, 6)$ and for the second model by $S^{\star} = (0, 1, 4, 2, 0, 3, 6)$, where $Y_1 := 1$, $Y_2 := 2$.

We typically obtain better objective function values using either function (RL1) or (RL2) if we prohibit overlappings of some activities. Nevertheless, it might happen that an optimal solution does not correspond to an extreme point of an order polytope belonging to some inclusion-maximal strict order, i.e. a strict order with maximum number of precedence relations. In this context, we consider a problem instance with three real activities and a single renewable resource. Activities 1 and 2 are critical while activity 3 has the following start time window $[ES_3, LS_3] = [1, 2]$. Fig. 2 depicts the resource profiles of two possible solutions, S^1 and S^2. Schedule $S^1 = (0, 0, 4, 1, 6)$ is an optimal solution for both the classical resource leveling problem and the overload problem with $Y = 2$, in spite of the fact that we have $O^1 = \{(1, 2)\} \subset O^2 = \{(1, 2), (1, 3)\}$ for the orders O^1, O^2 induced by schedules S^1 and S^2.

3. Practical application for resource leveling

An important and interesting application for resource leveling can be found in the field of overhauling power plants. In what follows, we discuss revisions based on the example of nuclear power plants. In Germany (and elsewhere in Europe), revisions must be conducted once annually at every plant. In the course of a revision of a nuclear power plant, all of its safety-related components, such as engines, pumps, and transformers, as well as piping, and valves are thoroughly inspected. Turbine maintenance is also carried out and spent fuel rods are replaced by new ones (cf. Eidgenössisches Nuklearsicherheitsinspektorat, 2009). Engines, transformers, and generators must be subjected to general overhauls and reactor pressure vessels hydrostatically tested for leakage at regular intervals of 5–10 years.

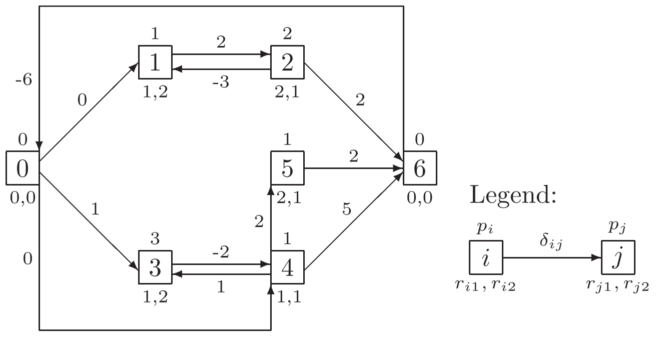


Fig. 1. Activity-on-node project network N with two renewable resources.

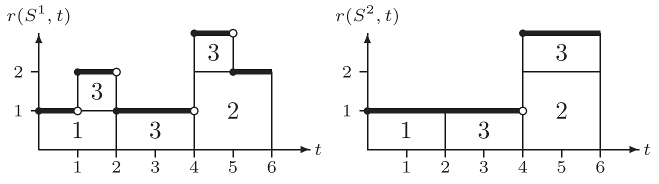


Fig. 2. Resource profiles of schedules S^1 and S^2.

All in all, around 5000 individual tasks (cf. Schreiber, 2007) that, for planning purposes, may be clustered according to the mechanical/electrical-engineering tasks and their respective locations within plants have to be performed. The clustering results in 20–50 work packages involving, e.g. inspection and maintenance of all cooling-water pumps or inspection of all safety valves present within the containment. The execution of these work packages is complicated by the constraint that, e.g. the cooling of reactors and fuel-rod pools must guaranteed at all times. Consequentially, the (in Germany redundantly designed) cooling systems of the plant cannot all be simultaneously released for inspection and maintenance operations. Another complication arises from the mandatory presence of independent experts who supervise nearly all inspection and maintenance work packages on behalf of the responsible regulatory agency (cf. Ilg, 2009).

Since nuclear power plants have to be shut down during a revision, and every day they are shut down entails around € 1 million in lost earnings (cf. Das Gupta, 2009), it should be obvious that revisions will be performed such that they will take no longer than needed to complete the work package accounting for the greatest length of time, or, in case of several work packages that must be executed in succession, the length of time needed to complete the longest sequence of such work packages. In view of the magnitude of the aforementioned lost of earnings, the personal costs involved are more or less negligible. Power plant operations thus engage around 1000 external specialists, in addition to their internal staff, for handling revisions in order that all work to be performed will usually be completed within a time frame of less than four weeks.

Detailed planning will be required if all work packages are to be completed as quickly as possible using a large number of workers, most of whom will be unfamiliar with the particular plant. Experience gained from recent revisions has shown that even utilization of workers and even occupancy of confined areas within the reactor building throughout the duration of revisions are of special importance. In particular, it has turned out that the occupation of the narrow annular catwalks within the containment must be comprehensively planned and coordinated, since they provide access to large numbers of pipes and valves that need to be inspected, while their narrow confines and occupational safety also have to be taken into account. Productive working by more than 15–20 persons working in parallel is thus precluded.

A resource leveling approach that will allow maintaining compliance with a prescribed project completion time thus appears advisable. The individual work packages of a revision are identified with activities of a so-called *revision project*. Workers, supervisors, as well as the aforementioned densely-occupied, confined areas are modeled by a set of renewable resources. Consideration of the classical resource leveling objective function leads to low resource utilizations of all resources at all times. In particular, if a serious incident should occur, restricting the number of workers working in confined areas allows safe and orderly evacuations.

The total overload cost function could be used with a threshold of Y in order to bring down peak periods, when more than 15 workers are simultaneously working on the annular catwalks within the containment.

4. Literature review

This section is devoted to surveying exact methods for solving resource leveling problems. Solution procedures presented in the literature may be segregated into three different categories:

- Enumeration schemes that enumerate the feasible integral start times of project activities.
- Enumeration schemes that enumerate the spanning trees (extreme points) of all feasible order networks (order polytopes) of the underlying project.
- Mixed-integer programming (MIP) models that consider time-indexed binary variables for the activities involved.

The first and the third categories require a discretization of the time horizon. Since the components $S_i^*, i \in V$, of at least one optimal schedule S^* are integers (assuming that $p_i, \delta_{ij} \in \mathbb{Z}_{\geqslant 0}, i,j \in V$, cf. Section 2), we are able to restrict the possible start times of activity $i \in V$ to a set of discrete times, $W_i := \{ES_i, \ldots, LS_i\} \subseteq T := \{0, 1, \ldots, \bar{d}\}$.

For resource leveling problems with precedence constraints, exact methods based on enumeration schemes that enumerate the (feasible) integral start times of project activities have been presented by Ahuja (1976), as well as Younis and Saad (1996). Ahuja (1976) considered the sum of the squared changes in the resource profiles, whereas Younis and Saad (1996) treated the sum of the absolute deviations of the resource request from a desired resource profile. Moreover, Bandelloni et al. (1994) devised a dynamic programming approach based on the integral floats of activities. For problems with general temporal constraints, Neumann and Zimmermann (2000) proposed a time-window based branch-and-bound procedure that uses, and continues, the concepts of the aforementioned approaches. Among other functions, (RL1) and (RL2), were investigated.

Nübel (1999) and Gather (2011) proposed tree-base enumeration schemes that enumerate all quasistable schedules (extreme points of order polytopes), where different techniques for avoiding redundancies were employed. In order to enumerate the extreme points of all order polytopes $\mathcal{S}_T(O)$, the corresponding spanning trees were generated by consecutively fixing activities' start times. More precisely, let set Ω contain a pair $(\mathcal{C}, S^{\mathcal{C}})$ for every partial schedule $S^{\mathcal{C}}$ (i.e. for each subtree) that has already been constructed. Then, the procedure is initiated using $\mathcal{C} = \{0\}$ and $S_0 = 0$. In each iteration, a pair $(\mathcal{C}, S^{\mathcal{C}})$ is removed from Ω. If $\mathcal{C} = V$, an extreme point of some order polytope has been found; otherwise, the current partial schedule $S^{\mathcal{C}}$ is extended as follows: For every $j \in V \setminus \mathcal{C}$, a set $\mathcal{D}_j$ of tentative start times, $t \in [ES_j(S^{\mathcal{C}}), LS_j(S^{\mathcal{C}})]$, for which there is an activity $i \in \mathcal{C}$ such that

(i) $t = S_i + \delta_{ij}$, i.e. temporal constraint $S_j - S_i \geqslant \delta_{ij}$ is binding, or
(ii) $t = S_i - \delta_{ji}$, i.e. temporal constraint $S_i - S_j \geqslant \delta_{ji}$ is binding, or
(iii) $t = S_i + p_i$, i.e. precedence constraint $S_j - S_i \geqslant p_i$ is binding, or
(iv) $t = S_i - p_j$, i.e. precedence constraint $S_i - S_j \geqslant p_j$ is binding,

is determined. Then for every $t \in \mathcal{D}_j$, the corresponding extended partial schedule $S^{\mathcal{C}'}$, where $\mathcal{C}' = \mathcal{C} \cup \{j\}$ and $S_j = t$, is added to Ω. Nübel (1999) and Gather (2011) introduced the concept of so-called "T-minimal trees" in order to avoid constructing two different trees that represent one and the same quasistable schedule. Additionally, Gather (2011) employed an enhanced variation on the bridge-concept devised by Gabow and Myers (1978), which has the advantage that no partial tree, except for the current one, must be stored. A workload-based lower bound for removing enumeration nodes is considered under both approaches. Apart from the calculation of the lower bound, the methods are almost independent of the scaling of the time axis.

Mixed-integer models for resource leveling problems are inspired by the work of Pritsker et al. (1969), Easa (1989), and Selle (2002). We start off by considering a basic discrete-time formulation proposed by Pritsker et al. (1969) employing binary variables x_{it} that allocate a feasible start time $t \in W_i$ to each activity $i \in V$, i.e.

$$x_{it} := \begin{cases} 1, & \text{if activity } i \text{ starts at time } t \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

The problem of finding an optimal feasible schedule for an objective function f may then be formulated as follows:

$$\begin{aligned} &\text{Minimize } f(x) \\ &\text{subject to} \sum_{t \in W_i} x_{it} = 1 \quad i \in V && (3) \\ &\sum_{t \in W_j} t\, x_{jt} - \sum_{t \in W_i} t\, x_{it} \geqslant \delta_{ij} \quad \langle i,j \rangle \in A && (4) \\ &x_{00} = 1 && (5) \\ &x_{it} \in \{0,1\} \quad i \in V, \quad t \in W_i. && (6) \end{aligned}$$

Constraints (3) ensure that each activity receives exactly one start time. Since $S_i = \sum_{t \in W_i} t\, x_{it}$ for all $i \in V$, inequalities (4) guarantee that the temporal constraints given by minimum and maximum time lags will be satisfied. Condition (5) sets the start time for the project to zero.

In order to present a time-indexed formulation for the objective functions (RL1) and (RL2), auxiliary variables $z_{kt} \geqslant 0$, which indicate the total resource requirements for resource $k \in \mathcal{R}$ and time t, are introduced. A (real) activity i is in progress, and requires resources at some time t, if $S_i \in \{t - p_i + 1, \ldots, t\}$. Inequalities (7) thus estimate the resource requirements of all activities for resource k and time t

$$z_{kt} \geqslant \sum_{i \in V} r_{ik} \sum_{\tau = \max\{ES_i, t - p_i + 1\}}^{\min\{t, LS_i\}} x_{i\tau} \qquad k \in \mathcal{R}, t \in T \setminus \{\bar{d}\} \tag{7}$$

and the objective functions may be specified by (cf. Selle, 2002)

$$f(x) := \sum_{k \in \mathcal{R}} c_k \sum_{t=0}^{\bar{d}-1} z_{kt}^2 \tag{RL1a}$$

$$f(x) := \sum_{k \in \mathcal{R}} c_k \sum_{t=0}^{\bar{d}-1} |z_{kt} - Y_k|. \tag{RL2a}$$

The resulting models contain $|V| + |V|^2 - (n+2) + |\mathcal{R}|(|T| - 1) + 1$ constraints, as well as $\sum_{i \in V} |W_i|$ binary and $|\mathcal{R}|(|T| - 1)$ continuous variables, whereas function (RL1a) is quadratic, and function (RL2a) is piecewise linear.

In the case of the "overload problem" with precedence constraints and one renewable resource, a float-based model has been presented by Easa (1989). The projects considered are given by activity-on-arc networks, without fictitious or dummy activities being required for modeling some types of simple, or special, precedence relationships (cf. Kelly, 1961). Here, binary variables x_{iq} that state the extent of shifting, $q \in \{1, \ldots, TF_i\}$, of activity $i \in V$ beyond its earliest start time are used, i.e.

$$x_{iq} := \begin{cases} 1, & \text{if activity } i \text{ is shifted } q \text{ time units after } ES_i \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

Condition (8) implies that $x_{iq} := 0$, iff $S_i = ES_i$ is satisfied.

We extended the Easa (1989) model in order to deal with general temporal constraints and a set of renewable resources. In doing so, we obtained a model that describes our overload problem and may be used for comparison purposes. Besides auxiliary variables $z_{kt} \geqslant 0$, auxiliary variables $z_{ikt} \geqslant 0$ that indicate the requirements of resource $k \in \mathcal{R}$ at time unit $t \in \{ES_i, \ldots, EC_i + TF_i - 1\}$ of an activity $i \in V$ are considered, where EC_i represents the earliest completion time of activity i.

$$\text{Minimize } \sum_{k\in\mathcal{R}} c_k \sum_{t=0}^{\bar{d}-1} z_{kt} \tag{RL2b}$$

subject to

$$\sum_{q=1}^{TF_i} x_{iq} \leqslant 1 \quad i \in V \tag{9}$$

$$ES_j - ES_i + \sum_{q=1}^{TF_j} q\ x_{jq} - \sum_{q=1}^{TF_i} q\ x_{iq} \geqslant \delta_{ij} \quad \langle i,j\rangle \in A \tag{10}$$

$$z_{ikt} \geqslant \left(1 - \sum_{q=1}^{TF_i} x_{iq}\right) r_{ik} + \sum_{q=1}^{\min\{t-ES_i, TF_i\}} x_{iq}\ r_{ik}$$
$$i \in V, k \in \mathcal{R}, ES_i \leqslant t < EC_i \tag{11}$$

$$z_{ikt} \geqslant \sum_{q=1+\min\{t-EC_i, TF_i\}}^{\max\{t-ES_i, TF_i\}} x_{iq} r_{ik} \quad i \in V, k \in \mathcal{R}, EC_i \leqslant t < EC_i + TF_i \tag{12}$$

$$z_{kt} \geqslant \sum_{i\in V} z_{ikt} - Y_k \quad k \in \mathcal{R}, t \in T \setminus \{\bar{d}\} \tag{13}$$

$$x_{iq} \in \{0,1\} \quad i \in V, q \in \{1, \ldots, TF_i\}. \tag{14}$$

Constraints (9) guarantee that the decision variables, x_{iq}, for an activity i will equal unity at, at most, one time (in the event that the activity is shifted, i.e. whenever $S_i > ES_i$). Since we are able to compute the start time of activity $i \in V$ from $S_i = ES_i + \sum_{q=1}^{TF_i} q\ x_{iq}$, inequalities (10) ensure that the temporal constraints involved will be satisfied. Provided that an activity i is not shifted beyond its earliest start time, inequalities (11) will arrange matters such that the resource requirements will be met at every point in time $t \in \{ES_i, \ldots, EC_i - 1\}$. Furthermore, if an activity i is shifted, inequalities (12) guarantee that the resource requirements will be considered for all times while activity i is in progress at its temporally shifted position. Finally, constraints (13) identify the deviations of the total resource requirements from a prescribed resource rate or threshold at time t. The model incorporates $|V| + |V|^2 - (n+2) + \sum_{i\in V}(TF_i + p_i - 1) + |\mathcal{R}|(|T| - 1)$ constraints, along with $|\mathcal{R}|(|T| - 1) + \sum_{i\in V}(TF_i + p_i - 1)$ real-valued auxiliary variables and $\sum_{i\in V} TF_i$ binary variables.

From a contemporary perspective, the discrete-time models presented above represent the sole practicable means for describing resource leveling problems. Event-based model formulations (cf. Koné et al., 2011) and flow-based model formulations (cf. Artigues et al., 2003) are less applicable, since general temporal constraints or the duration of the overlapping of activities, which are needed for determining the values of objective function (RL1a) and (RL2a), can hardly be prescribed.

5. Improvements in modeling

MIP-models with linear constraints and a linear objective function are proved to be a key factor for obtaining exact solutions to combinatorial optimization problems in reasonable time. However, objective functions (RL1a) and (RL2a) are nonlinear. In order to determine how nonlinearity affects a solver's performance, Gather (2011) solved the model (RL1a), (3)–(7) using the CPLEX-solver for quadratic programs. Since the running times were quite long, preprocessing and linearization techniques will need to be considered in order to reduce, or simplify, such optimization problems. The methods developed below will be of assistance in finding optimal solutions to benchmark problems that thus far have never been solved to optimality. In Section 5.1, we determine upper and lower bounds on resource requirements in order to limit the domains of auxiliary variables. Section 5.2 considers analytical techniques for linearizing objective functions (RL1a) and (RL2a). Finally, in Section 5.3, we introduce additional constraints that may be employed as cutting planes in the model.

5.1. Upper and lower resource requirement bounds

A tighter linear programming relaxation usually facilitates the solution process within a branch-and-bound or branch-and-cut framework, since fewer nodes need to be evaluated. In our preprocessing step, we start off by restricting the domains of auxiliary variables to be employed by the model. To that end, we identify $P_{kt} \geqslant 0$ with the minimum and $H_{kt} \geqslant 0$ with the maximum requirements for resource $k \in \mathcal{R}$ that can occur at time $t \in \{0, \ldots, \bar{d} - 1\}$, for the case of any feasible schedule.

The values of $P_{kt}, k \in \mathcal{R}, t \in \{0, \ldots, \bar{d} - 1\}$, may be obtained by considering the unavoidable time interval, $[LS_i, EC_i[$, for every activity $i \in V$. Obviously, every real activity i must be in progress during its unavoidable time interval, regardless of its start time, $S_i \in [ES_i, LS_i]$. The corresponding resource profiles, $\tilde{r}_k(\cdot), k \in \mathcal{R}$, have at most $2n + 1$ jump points (start and completion times of activities $0, \ldots, n$), $\tau \in \{0, \ldots, \bar{d} - 1\}$. The active set $\widetilde{\mathcal{A}}(\tau)$ at jump point τ indicates the set of real activities that must at least be in progress at time τ, no matter at which start times the activities will be scheduled. Fig. 3 illustrates the resource profile $\tilde{r}_1(\cdot)$ for the problem instance depicted in Fig. 1, which has three jump points $\tau \in \{0, 3, 4\}$. Since the resource requirements remain constant while activities are in progress, the condition $P_{kt} = P_{k\tau} := \sum_{i\in\tilde{\mathcal{A}}(\tau)} r_{ik}$ will be satisfied for $t \in \{\tau, \ldots, \tau' - 1\}$, where τ, τ' are successive jump points, or for $t \in \{\tau, \ldots, \bar{d} - 1\}$, where τ is the final jump point.

For a real activity i, the interval $[LS_i, EC_i[$ will be non-empty whenever $0 < LS_i - ES_i < p_i$. The lower bounds P_{kt} will therefore be strongly dependent upon the prescribed maximum project duration $\bar{d}$ of the underlying project, i.e. tight project deadlines will yield better lower bounds.

The values of $H_{kt}, k \in \mathcal{R}, t \in \{0, \ldots, \bar{d} - 1\}$, may be computed in inverse manners. For every activity i, we regard the interval $[ES_i, LC_i[$ as its execution interval, i.e. we assume that i starts at its earliest start time and requires a processing time of $p_i := LC_i - ES_i$ for completion. Since $[S_i, S_i + p_i[\subseteq [ES_i, LC_i[$ holds for any feasible start time of activity i, the corresponding resource profiles, $\hat{r}_k(\cdot)$, provide upper bounds H_{kt} on the resource requirements for resource k at time t. Fig. 4 shows the resource profile $\hat{r}_1(\cdot)$ for the sample project instance having five jump points $\upsilon \in \{0, 1, 2, 3, 5\}$. The upper bounds for every resource k and each point in time $t \in \{\upsilon, \ldots, \upsilon' - 1\}$ may then be determined from $H_{kt} = H_{k\upsilon} := \sum_{i\in\hat{\mathcal{A}}(\upsilon)} r_{ik}$, where υ, υ' are pairs of successive jump points, or $t \in \{\upsilon, \ldots \bar{d} - 1\}$, in the event that υ is the final jump point.

Until now we have dealt with the earliest and latest start and completion times for the activities. However, we are also able to determine time-orientated precedences among project activities, based on computations of temporal scheduling. If the length l_{ij} of the longest path from node i to node j on project network N exceeds or equals the duration p_i of activity i, then both activities cannot be executed simultaneously. In order to determine upper bounds $H_{kt}, k \in \mathcal{R}, t \in \{0, \ldots, \bar{d} - 1\}$, that will be as tight as

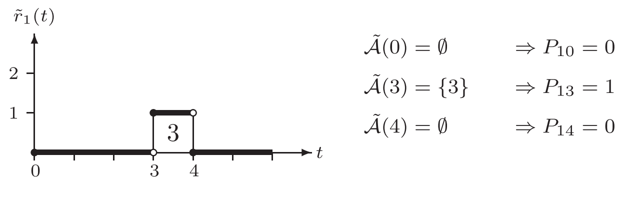
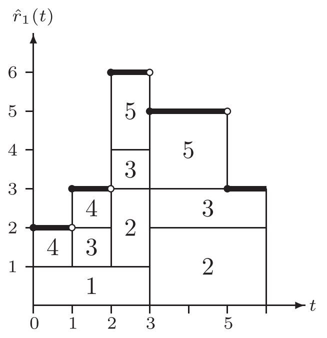


Fig. 3. Resource profile $\tilde{r}_1(\cdot)$ and lower bounds on resource requirements.



$\hat{\mathcal{A}}(0) = \{1,4\} \quad \Rightarrow H_{10} = 2$
$\hat{\mathcal{A}}(1) = \{1,3,4\} \quad \Rightarrow H_{11} = 3$
$\hat{\mathcal{A}}(2) = \{1,2,3,5\} \quad \Rightarrow H_{12} = 6$
$\hat{\mathcal{A}}(3) = \{2,3,5\} \quad \Rightarrow H_{13} = 5$
$\hat{\mathcal{A}}(5) = \{2,3\} \quad \Rightarrow H_{15} = 3$

Fig. 4. Resource profile $\hat{r}_1(\cdot)$ and upper bounds on resource requirements.

possible, we make use of the known values of the components of the distance matrix $(l_{ij})_{i,j\in V}$ and the concept of so-called antichains (cf. Möhring, 1984; Schwindt, 2005).

Definition 1. Let O be a strict order within activity set V. A set $U \subseteq V$, such that neither $(i, j) \in O$ nor $(j, i) \in O$ holds for any two activities $i, j \in U$, is termed an *antichain* in O.

An antichain is a set of activities, such that any two distinct activities, i and j, may overlap under a feasible schedule. For an antichain U, we define a weight $w_k(U) := \sum_{i\in U} r_{ik}$ for every resource $k \in \mathcal{R}$. A *maximum-weight antichain* $U_k^{max} \subseteq V$ for resource k is thus an antichain that produces the maximum weight $H_k \geqslant w_k(U)$ among all antichains U in O.

The weight $H_k, k \in \mathcal{R}$, of a maximum-weight antichain $U_k^{max} \subseteq V$ equals the total number of units of resource k required for project execution, in the most unfavorable case. Weight H_k may be efficiently determined by computing a minimum origin–destination flow on a flow network $G_k(\widetilde{V},\widetilde{A})$. Vertex set $\widetilde{V} = W \cup W' \cup \{\alpha,\omega\}$ consists of nodes $W = V\backslash\{0, n + 1\}$ for (usually) real activities, nodes W' representing copies of activities, as well as an origin node α and a destination node ω. We identify the copy of activity i with node $i + n$. In doing so, we are able to transform the node weights r_{ik} of the underlying activity-on-node network N into arc capacities of flow network G_k. Set $\widetilde{A}$ will contain arcs $\langle\alpha, i\rangle$, and $\langle i, i + n\rangle$, $i \in W$; $\langle j, \omega\rangle$, $j \in W'$, as well as arcs $\langle i + n, j\rangle$, $i, j \in W$, $i \neq j$, iff the length l_{ij} of a longest path from node i to j in activity-on-node network N exceeds or equals the duration of activity i. Two non-negative weights, λ_{ij} and κ_{ij}, are associated with each arc $\langle i,j\rangle \in \widetilde{A}$ representing the lower and upper arc capacities, respectively. We set λ_{ij} = $\kappa_{ij} := r_{ik}$ for arcs $\langle i, i + n\rangle$, $i \in W$, as well as $\lambda_{ij} := 0$ and $\kappa_{ij} := \infty$ for all other arcs in G_k.

A minimum origin–destination flow on network $G_k, k \in \mathcal{R}$, may be determined within $\mathcal{O}(n^3)$ by two successive applications of the FIFO preflow-push algorithm for the maximum-flow problem involving upper arc capacities (see e.g. Ahuja et al., 1993, Sect. 7.7). The intensity of the solution flow equals the sought maximum resource requirement. The calculation of weight H_k allows imposing more stringent restrictions on the upper bounds, i.e. we are able to set $H_{kt} := \min\{H_{kt}, H_k\}$, $k \in \mathcal{R}, t \in \{0,\ldots,\bar{d}-1\}$.

To illustrate the concept of antichains, we consider flow network $G_1(\widetilde{V},\widetilde{A})$ for the first resource (k = 1) of our sample instance depicted in Fig. 5. Every arc $\langle i,j\rangle \in \widetilde{A}$ is associated with a parameter triad composed of the lower and upper arc capacities and the number of flow units ϕ_{ij} traversing the arc in an optimal solution. A minimum origin–destination flow on the network has the intensity H_1 = 5.

The drawback of the general concept of antichains is that it involves pretending that all activities can be in progress at all times. However, at time t only activities $i \in V$, for which $t \in [ES_i, LC_i]$, could be processed. This fact warrants the combination of the two aforementioned concepts (start time windows and antichains) for generating upper bounds H_{kt}. We therefore determine the weight of a maximum-weight antichain, $U_{k\upsilon}^{max}$, at every jump point $\upsilon \in \{0,\ldots,\bar{d}-1\}$ in resource profiles $\hat{r}_k(\cdot)$. Hence, we consider the algorithm described above in order to compute a minimum origin–destination flow on a (modified) flow network $G_{k\upsilon}(\widetilde{V},\widetilde{A})$. Networks G_k and $G_{k\upsilon}$ differ solely in their definitions of arc capacities. We set $\lambda_{i,i+n} := 0$ and $\kappa_{i,i+n} := \infty$ for all activities $i \in W \setminus \widehat{\mathcal{A}}(\upsilon)$. The intensity of the solution flow equals the maximum resource requirement $H_{k\upsilon}$ at time υ. Table 1 summarizes the resultant minimum and maximum resource requirements P_{kt}, H_{kt}, k = 1, 2, t = 0, . . . , 5 for our sample instance. The values of the parameters H_{11} and H_{21} have been reduced by max{r_{31}, r_{41}} = 1 and max{r_{32}, r_{42}} = 1, since activities 3 and 4 cannot be executed simultaneously.

5.2. Elementary linearization techniques

In order to make use of fast exact algorithms provided by commercial MIP-solvers, we study linearization techniques for objective functions (RL1a) and (RL2a), where we focuse on exact linearization methods, since we do not want to "lose" any opportunities for finding an optimal solution.

Until now objective function (RL1a) is based on the auxiliary variables $0 \leqslant z_{kt} \leqslant H_{kt}$, $k \in \mathcal{R}, t \in \{0,\ldots,\bar{d}-1\}$. In order to linearize (RL1a), we divide the interval of resource requirements $[0,\ldots,H_{kt}]$, into H_{kt} equal parts [0, 1], [1, 2], . . . , [H_{kt} − 1, H_{kt}]. In addition, a continuous auxiliary variable, y_{kth}, is introduced for each part, where

$$0 \leqslant y_{kth} \leqslant 1 \quad k \in \mathcal{R}, t \in T \setminus \{\bar{d}\}, h \in \{1,\ldots,H_{kt}\}. \tag{15}$$

The auxiliary variables z_{kt} in model (3)–(6), and the auxiliary variables y_{kth} may then be linked using the relation

$$z_{kt} = \sum_{h=1}^{H_{kt}} y_{kth} \quad k \in \mathcal{R}, T \setminus \{\bar{d}\}. \tag{16}$$

The assumption that the resource requirements r_{ik} are non-negative, along with the fact that the objective function is convex will force the variables y_{kth} to take on values of either 0 or 1, which is why we merely need to demand that the condition $y_{kth} \in [0, 1]$, rather than the condition $y_{kth} \in \{0, 1\}$, be satisfied.

Moreover, we were readily able to compute a direction factor, $2h - 1$, for every variable y_{kth}, which equals the difference between h^2 and $(h-1)^2$. Combined with auxiliary variables $y_{kth} \in [0, 1]$, we obtain an exact approximation for objective function (RL1a) that depends on $z_{kt} \in \mathbb{N}_0$. The stepwise-linear objective function may now be written as

$$f(x) := \sum_{k\in\mathcal{R}} c_k \sum_{t=0}^{\bar{d}-1} \sum_{h=1}^{H_{kt}} (2h-1)\, y_{kth}. \tag{RL1b}$$

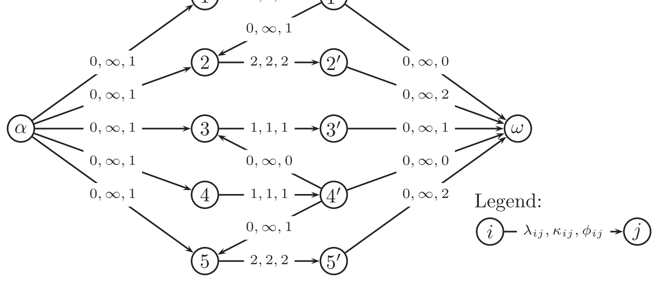


Fig. 5. Flow network $G_1(\widetilde{V},\widetilde{A})$ and minimum origin–destination flow.

Table 1
Minimum and maximum resource requirements for the sample instance.

t	0	1	2	3	4	5	t	0	1	2	3	4	5
P_{1t}	0	0	0	1	0	0	P_{2t}	0	0	0	2	0	0
H_{1t}	2	2	5	5	5	3	H_{2t}	3	4	5	4	4	3

Another possibility to linearize objective function (RL1a) involves considering additional binary variables $g_{kth}, k \in \mathcal{R}, t \in \{0, \ldots, \bar{d}-1\}, h \in \{0, \ldots, H_{kt}\}$, designating the resource requirement of resource k at time t. We then obtain

$$g_{kth} := \begin{cases} 1, & \text{if } h \text{ units of resource } k \text{ are required at time } t \\ 0, & \text{otherwise.} \end{cases}$$

The introduction of constraints (17) guarantees that just a single resource utilization level h for a resource k will exist at time t.

$$\sum_{h=0}^{H_{kt}} g_{kth} = 1 \quad k \in \mathcal{R}, T \setminus \{\bar{d}\} \tag{17}$$

The resource requirements for resource k at time t may then be determined from the inequalities

$$z_{kt} \geqslant \sum_{h=0}^{H_{kt}} h\, g_{kth} \quad k \in \mathcal{R}, T \setminus \{\bar{d}\} \tag{18}$$

and the classical resource leveling objective function will finally be given by

$$f(x) := \sum_{k \in \mathcal{R}} c_k \sum_{t=0}^{\bar{d}-1} \sum_{h=0}^{H_{kt}} h^2\, g_{kth}. \tag{RL1c}$$

The total overload cost function (RL2a) contains the term $|z_{kt} - Y_k|$ which may be replaced by $\max\{z_{kt} - Y_k, Y_k - z_{kt}\}$. In order to linearize the resulting max-function, we introduced auxiliary variables $v_{kt} \geqslant 0, k \in \mathcal{R}, t \in \{0, \ldots, \bar{d}-1\}$, that represent the deviations of total resource requirements from the resource rate. Then, the resource leveling function could be replaced by the linear function

$$f(x) := \sum_{k \in \mathcal{R}} c_k \sum_{t=0}^{\bar{d}-1} v_{kt}. \tag{RL2c}$$

However, inequalities (7) must also be substituted by

$$v_{kt} \geqslant \sum_{i \in V} r_{ik} \sum_{\tau=\max\{ES_i, t-p_i+1\}}^{\min\{t, LS_i\}} x_{i\tau} - Y_k \quad k \in \mathcal{R}, T \setminus \{\bar{d}\}. \tag{19}$$

5.3. Additional constraints

Christofides et al. (1987) proposed a set of inequalities to allow formulating precedence constraints. Those inequalities may be extended to the case of problems involving general temporal constraints. The following formulation ensures that the minimum and maximum time lags between activities will be observed.

$$\sum_{\tau=t}^{LS_i} x_{i\tau} + \sum_{\tau=ES_j}^{\min\{LS_j, t+\delta_{ij}-1\}} x_{j\tau} \leqslant 1 \quad \langle i,j \rangle \in A, t \in T \tag{20}$$

Compared to inequalities (4), which comprise at most $|V|^2$ constraints, inequalities (20) require $|V|^2|T|$ constraints.

Furthermore, the reformulation of inequalities (3) according to

$$x_{it} + \sum_{\substack{\tau=ES_i \\ \tau \neq t}}^{LS_i} x_{i\tau} = 1 \quad i \in V, \; ES_i \leqslant t \leqslant LS_i \tag{21}$$

enables the solver to generate special order sets of Type 1 within the branch-and-bound tree. This formulation includes $\sum_{i \in V}(LS_i - ES_i + 1)$ more constraints than inequalities (3).

6. Computational results

This section covers the results of computations undertaken in order to investigate the performance of the various resource leveling formulations, where we distinguish between

- Model M$_1$: (RL1b), (3)–(7), (15), (16) and
- Model M$_2$: (RL1c), (3)–(7), (17), (18), for the case of the classical resource leveling problem as well as between
- Model M$_3$: (RL2b), (9)–(14) and
- Model M$_4$: (RL2c), (3)–(6), (19) for the case of the overload problem.

We start off by describing the composition and generation of the problem instances used for testing the model formulations (cf. Section 6.1). In an experimental performance analysis, we call CPLEX 12.1 to solve medium-scale problem instances to optimality. The solutions of the branch-and-cut procedures used by CPLEX are compared to those for tree-based branch-and-bound methods involving a sophisticated constructive lower bound (the best exact methods known from the literature) (cf. Gather et al., 2011). Our new methods usually outperform the state-of-the-art tree-based algorithms. Instances with up to 50 activities and tight project deadlines are solved to optimality for the first time (cf. Section 6.2).

6.1. Benchmark instances

The computational tests have been performed on two test sets, T_1 and T_2, generated by the ProGen/max problem instance generator (cf. Schwindt, 1998). The design of both test sets highlights several control parameters that affect the behavior of scheduling algorithms. In particular, parameters for the network structure, the activities and the resources are considered. We identified the first test set with the instances provided by Kolisch et al. (1999). Since this benchmark covers exclusively instances involving as many as 30 activities, where each activity requires just one resource, even if $|\mathcal{R}| > 1$ resources are available, we decided to generate a second benchmark set. This second test set, T_2, incorporates instances involving as many as 50 activities, where every activity may require more than a single resource for its execution, if $|\mathcal{R}| > 1$.

The control parameters used by ProGen/max in order to specify the networks to be constructed may be described as follows. Parameters for the network structure are the number n of (usually real) activities, and the restrictiveness of Thesen (RT) that measures the degree to which precedence constraints restrict the total number of feasible activity sequences. We obtain $RT = 0$ for a parallel network and $RT = 1$ for a series network. The parameters for any activity $i \in V$ are the resource utilization $r_{ik}, k \in \mathcal{R}$, and the processing time p_i. The resource parameters involved are the number $|\mathcal{R}|$ of differing renewable resources, and the resource factor (RF), which denotes the average fraction of the $|\mathcal{R}|$ resources used per activity.

A series project network (i.e. $RT > 0.5$) usually contains more indirect temporal constraints (given by the transitive closure) than a parallel one. Incorporating more constraints substantially reduces the search space by introducing more restrictions that must be satisfied. Moreover, if a large number of resource requirements are identically zero (i.e. $RF < 0.5$), then fewer activities will be competing for the available renewable resources. In order to generate some rather difficult and practically-relevant instances (cf. the application of resource leveling problems in Section 3), we choose resource factors exceeding 0.5 and set the restrictiveness of Thesen to be either 0.3 or 0.6 within test set T_2. Table 2 lists the parameter values for both test sets, T_1 and T_2. Test set T_1 consists of 810 problem instances (270 for each n), and T_2 contains 600 instances (120 for each n).

6.2. Performance study

In a comprehensive performance analysis, we studied the classical resource leveling problem and the overload problem by considering the four different model formulations, $M_1, \ldots, M_4$. Each instance is solved using CPLEX 12.1 and ILOG's Concert interface for communications with the solver. Those tests are performed on an Intel 4-core processor with 2,66 GHz, 6 GB RAM under Windows 7. CPLEX uses a branch-and-cut approach, and we include problem-specific preprocessing techniques and cutting planes (cf. Sections 5.1, 5.2, 5.3). The features of CPLEX allow generating general cuts during optimization. Cuts are therefore added at the root node, and at other nodes, whenever warranted by conditions. We identified GUB-cover, clique, and cover cuts as being particularly suitable for our purposes, since such cuts arise from considering inequalities consisting of binary variables and integral coefficients (covering, e.g. both temporal and resource constraints). Moreover, effective zero-half cuts may be generated by taking account of resource constraints (7), (11), and (12), and effective mixed-integer rounding cuts may be generated by taking account of resource constraints (19). Models and cutting planes are implemented in an object-orientated manner using C++ and compiled with MS Visual Studio.NET 2008. The total elimination of CPLEX–cuts would yield average run times of around 100 s for instances involving 20 activities and tight project deadlines, which can normally be solved within 3 s.

Since the resource leveling problems considered are $\mathcal{NP}$-hard optimization problems, 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 three hours, after which the best solution found up to that point is returned. In order to determine the impact of expanding time-windows $[ES_i, LS_i]$, for activities $i \in V$, we tested each instance using the shortest possible deadline, $\bar{d} := ES_{n+1}$. We subsequently extended the prescribed maximum project duration to $\bar{d} := \alpha ES_{n+1}$, where $\alpha \in \{1.1, 1.5\}$, by overwriting the prescribed deadlines specified in the test instances. Moreover, to improve the bounding accuracy and to speed up the solver, particularly for large instances, an upper bound on the objective function is supplied. Thereby, the Ballestin et al. (2007) procedure was used for generating upper bounds for those resource leveling problems under consideration.

Under preliminary tests, we have investigated the effectiveness of constraints (20) and (21), where we considered 120 medium-scale instances with 20 activities, three renewable resources, and different project deadlines, $\bar{d} := \alpha ES_{21}$ (instances may be explicitly identified by their names n-$|\mathcal{R}|$-α). Table 3 lists the results of that procedure using model M_1. t_{cpu} designates the average computation times, and $\text{Inst}_{<3h}$ the numbers of instances solved to optimality within the time limit. Column "# Opt" shows the total numbers of optimally solved instances.

Table 2
Control parameters of test sets T_1 and T_2.

Parameter	Test set T_1	Test set T_2
n	10, 20, 30	10, 15, 20, 30, 50
RT	0.25, 0.5, 0.75	0.3, 0.6
r_{ik}	$1, \ldots, 5$	$1, \ldots, 5$
p_i	$1, \ldots, 10$	$1, \ldots, 10$
$[\lvert\mathcal{R}\rvert, RF]$	[1, 1.0], [3, 0.33], [5, 0.2]	[1, 1.0], [3, 0.7], [5, 0.6]

Regarding the total number of instances solved within the time limit, model M_1 (with no additional constraints) produces the best results. A comparison of run times shows that all model formulations yield similar run times for instances having a tight deadline $\bar{d}$. Inequalities (20) work well for instances having longer deadlines. Although considering more constraints than in the case of the standard formulation, M_1, obviously helps the solver to terminate the enumeration quickly for instances that are, in general, easy to solve, it hinders the solution process for more difficult instances. In the remaining portion of this paper, we attach importance to the number of instances optimally solved and exclude inequalities (20) and (21) from the model formulations.

We begin our analyses with test set T_1 provided by Kolisch et al. (1999). The solutions obtained from branch-and-cut procedures are compared to those obtained from the tree-based branch-and-bound methods of Gather et al. (2011). In order to arrive at fair comparisons, the tree-based methods are terminated after six hours (the program uses just one processor core). We report both the average computation times, t_{cpu}, and the numbers of instances, ($\text{Inst}_{<\beta h}$), that are optimally solved within a time limit of β hours.

Table 4 summarizes the results for the classical resource leveling problem and instances with $n = 10, 20, 30$ activities as well as differing project deadlines, $\bar{d} := \alpha ES_{n+1}$, $\alpha \in \{1.0, 1.1, 1.5\}$. The instances are denoted by the same descriptors "*rlp_jn*" used in the literature. We also introduced a suffix "$-\alpha$" indicating the lengths of time-windows, $[ES_i, LS_i]$, for the various activities $i \in V$.

In the case of small instances with ten activities, all solution methods find optimal solutions within four seconds, but it is apparent that the state-of-the-art tree-based algorithm outperforms the branch-and-cut procedures used by CPLEX. However, the tree-based method is suitable for solving instances with 20 or more activities to a limited extent. Fewer than 80% of instances contained in *rlp_j*20–1.0 were optimally solved, and the average computation time exceeded 25 min. We therefore skipped pursuing further details for the larger instances listed in the table. In contrast, the branch-and-cut procedures are, indeed, able to cope with many activities and differing project deadlines. All test instances contained in *rlp_j*20 and *rlp_j*30 having a tight deadline, ($\bar{d} := ES_{n+1}$), are solved to optimality for the first time. Moreover, extending the maximum computation time to 18 h provides optimal solutions for eight of the eleven unsolved instances, employing model M_1, while, for the three remaining instances, the average deviations of the feasible solutions from the associated lower bound amount to 2.7%.

Table 5 summarizes the results for the overload problem. Model M_4 works particularly well. All instances with up to 30 activities and tight, or moderate, project deadlines, ($\bar{d} := \alpha ES_{n+1}, \alpha = 1.0, 1.1$), are solved to optimality for the first time. The tree-based method performs well in case of small instances involving ten activities, but is unsuitable for dealing with practically-relevant applications with more than ten activities. Furthermore, model M_3 is found to be of little value as an alternative method to M_4.

We continue our analyses using the new test set T_2. In order to investigate the increases in computation time occasioned by increasing the number n of activities or the number $|\mathcal{R}|$ of resources, we subdivided test set T_2 into blocks of 40 instances (having the same n and the same $|\mathcal{R}|$). Table 6 lists the performance results for the classical resource leveling problem based on models M_1 and M_2 and the corresponding tree-based branch-and-bound method. In addition to the parameters n, $\mathcal{R}$, and α, we considered extensions of the resource requirements of the activities (i.e. $r_{ik} := 10r_{ik}, i \in V, k \in \mathcal{R}$), which lead to large increases in the upper bounds $H_{kt}, k \in \mathcal{R}, T \setminus \{\bar{d}\}$, where instance names obtained an additional suffix "-10".

As expected, the tree-based algorithm performs very well for instances involving 10 activities. The average run times are

Table 3
Computation times and the numbers of instances solved (classical RLP).

Instances	20-3-1.0		20-3-1.1		20-3-1.5		# Opt.
	t_{cpu}	$\text{Inst}_{<3h}$	t_{cpu}	$\text{Inst}_{<3h}$	t_{cpu}	$\text{Inst}_{<3h}$	
M_1	2.410	40	260.844	40	2669.788	30	110
$M_1 \cup (20)$	3.713	40	20.345	39	2240.655	27	106
$M_1 \setminus (4) \cup (20)$	2.913	40	168.497	40	2031.113	28	108
$M_1 \cup (21)$	2.073	40	232.258	40	2644.545	28	108
$M_1 \setminus (3) \cup (21)$	2.380	40	168.350	39	2633.396	30	109

Table 4
Computation times and the numbers of instances solved (test set T_1, classical RLP)

Instances	M_1		M_2		Tree-based B&B	
	t_{cpu}	$\text{Inst}_{<3h}$	t_{cpu}	$\text{Inst}_{<3h}$	t_{cpu}	$\text{Inst}_{<6h}$
*rlp_j*10-1.0	0.038	270	0.025	270	0.015	270
*rlp_j*10-1.1	0.120	270	0.061	270	0.065	270
*rlp_j*10-1.5	2.854	270	3.359	270	0.219	270
*rlp_j*20-1.0	0.575	270	0.981	270	1605.740	215
*rlp_j*20-1.1	9.281	270	14.566	270	–	–
*rlp_j*20-1.5	405.195	263	771.905	234	–	–
*rlp_j*30-1.0	59.195	270	28.391	269	–	–
*rlp_j*30-1.1	356.071	266	425.693	259	–	–
# Opt.		2149		2112		1025

Table 5
Computation times and the numbers of instances solved (test set T_1, overload problem).

Instances	M_3		M_4		Tree-based B&B	
	t_{cpu}	$\text{Inst}_{<3h}$	t_{cpu}	$\text{Inst}_{<3h}$	t_{cpu}	$\text{Inst}_{<6h}$
*rlp_j*10-1.0	0.589	270	0.009	270	0.009	270
*rlp_j*10-1.1	48.253	270	0.017	270	0.036	270
*rlp_j*10-1.5	470.901	248	0.258	270	0.142	270
*rlp_j*20-1.0	269.142	259	0.531	270	1680.259	226
*rlp_j*20-1.1	–	–	3.898	270	–	–
*rlp_j*20-1.5	–	–	127.705	268	–	–
*rlp_j*30-1.0	–	–	43.288	270	–	–
*rlp_j*30-1.1	–	–	60.262	270	–	–
# Opt.		1047		2158		1036

Table 6
Computation times and the numbers of instances solved (test set T_2, classical RLP).

Instances	M_1		M_2		Tree-based B&B	
	t_{cpu}	$\text{Inst}_{<3h}$	t_{cpu}	$\text{Inst}_{<3h}$	t_{cpu}	$\text{Inst}_{<6h}$
10-1-1.0	0.029	40	0.029	40	0.028	40
10-3-1.0	0.153	40	0.108	40	0.033	40
10-5-1.0	0.199	40	0.206	40	0.049	40
10-1-1.1	0.071	40	0.067	40	0.097	40
10-3-1.1	0.278	40	0.301	40	0.155	40
10-5-1.1	0.627	40	0.570	40	0.384	40
10-1-1.5	0.992	40	2.306	40	0.513	40
10-3-1.5	4.172	40	7.236	40	0.895	40
10-5-1.5	20.292	40	64.975	40	2.305	40
10-1-1.5–10	277.590	36	19.517	40	1.188	40
10-3-1.5–10	744.338	33	85.417	40	2.154	40
10-5-1.5–10	1181.030	26	391.993	40	5.029	40
15-1-1.0	0.154	40	0.154	40	0.146	40
15-3-1.0	0.291	40	0.356	40	25.722	40
15-5-1.0	0.513	40	1.194	40	29.397	40
15-1-1.1	1.164	40	1.762	40	136.953	40
15-3-1.1	5.665	40	11.713	40	382.971	40
15-5-1.1	7.586	40	8.275	40	688.150	40
15-1-1.5	151.005	40	277.711	39	775.556	40
15-3-1.5	262.679	38	1206.530	34	2813.005	37
15-5-1.5	824.416	40	2656.601	30	3930.287	37
20-1-1.0	0.336	40	0.691	40	1792.086	38
20-3-1.0	2.410	40	10.994	40	1540.256	35
20-5-1.0	2.410	40	41.639	40	3392.972	31
# Opt.		933		943		938

invariably less than five seconds. Extensions of the time-windows has a crucial impact on the performances of the MIP–approaches, since both the number of binary decision variables, x_{it}, $i \in V$, $t \in T$, and the number of constraints increased with the increasing length of the time horizon. In contrast, the tree-based method turns out to be much more robust in relation to the deadlines for project completion. Considerations involving extended resource requirements also lead to drastic increases in the numbers of auxiliary variables, y_{kth}, and binary variables, $g_{kth}, k \in \mathcal{R}, t \in T \setminus \{\bar{d}\}, h \in \{1, \ldots, H_{kt}\}$. In particular, the procedure using model M_1 proved incapable of terminating the enumerations for all of the respective instances. Compared thereto, the procedure using model M_2, which incorporates exclusively binary variables, come off well. Based on the practically-relevant problem dimension of 15 activities, the performance of model M_1 must be regarded as acceptable. The computation times of the tree-based method are highly dependent upon the network structure, the activities and the resources involved. Increasing numbers of activities leads to the existence of many feasible order networks (i.e. order networks with no cycle of positive length) that must be considered during enumeration. Furthermore, the long running times are a consequence of the parallelism of project networks (with $RT < 0.5$) and the existence of a large slack factor $SF \in \{0.5, 1.0\}$. A positive slack factor avoids that activities are firmly tied by temporal constraints. M_2 yields the worst performance. Since the number of instances solved to optimality is relatively low for the tree-based method, and for model M_2, we investigated larger instances only with tight deadlines.

Table 7 summarizes the results for the overload problem in the case of models M_3 and M_4, and the respective tree-based branch-and-bound method. For small instances with ten activities no superiority of the tree-based method over the procedure using model M_4 can be ascertained. Extensions of the time-windows has no relevant impacts on computation times. Moreover, the run times for instances having long project durations, as well as instances having long project durations and extended resource requirements are similar, since the number of auxiliary variables, $v_{kt}, k \in \mathcal{R}, t \in T \setminus \{\bar{d}\}$, depends on the number of resources and the time horizon. The procedure using model M_3 fails to terminate the enumerations within the time limit in some cases. Treating binary decision variables, x_{iq}, $i \in V$, $q \in \{1, \ldots, TF_i\}$, is obviously not the best choice. For larger instances involving 15 or more activities, the procedure using model M_4 yields the best results. All of the problem instances under consideration are optimally solved within 1.5 min, or less.

The computational studies that we have conducted thus far, demonstrate the dominance of the branch-and-cut procedures provided by CPLEX in combination with suitable model formulations. However, since the procedure using model M_1 should be considered in order to solve instances of the classical resource leveling problem, and the procedure using model M_4 should be considered in order to solve instances of the overload problem, our further

Table 7
Computation times and the numbers of instances solved (test set T_2, overload problem).

Instances	M_3		M_4		Tree-based B&B	
	t_{cpu}	Inst<3h	t_{cpu}	Inst<3h	t_{cpu}	Inst<6h
10-1-1.0	0.031	40	0.014	40	0.022	40
10-3-1.0	0.140	40	0.027	40	0.023	40
10-5-1.0	0.298	40	0.051	40	0.038	40
10-1-1.1	0.060	40	0.028	40	0.050	40
10-3-1.1	7.344	40	0.087	40	0.300	40
10-5-1.1	0.570	40	0.161	40	0.384	40
10-1-1.5	8.063	40	0.102	40	0.120	40
10-3-1.5	184.346	39	0.466	40	0.414	40
10-5-1.5	1078.888	36	2.141	40	1.393	40
10-1-1.5–10	3.012	40	0.049	40	0.128	40
10-3-1.5–10	163.805	39	0.388	40	0.438	40
10-5-1.5–10	777.612	35	2.677	40	1.494	40
15-1-1.0	0.309	40	0.096	40	9.733	40
15-3-1.0	0.975	40	0.071	40	20.043	40
15-5-1.0	14.390	40	0.112	40	23.640	40
15-1-1.1	1.843	40	0.272	40	47.848	40
15-3-1.1	105.933	39	0.875	40	262.949	40
15-5-1.1	85.256	38	2.127	40	518.701	40
15-1-1.5	36.413	39	3.116	40	42.996	40
15-3-1.5	320.092	30	80.794	40	1441.203	39
15-5-1.5	1268.314	18	88.531	40	1912.634	38
20-1-1.0	1.236	40	0.191	40	1566.697	38
20-3-1.0	16.086	39	1.172	40	2283.954	37
20-5-1.0	48.214	40	1.523	40	2751.409	31
# Opt.		912		960		943

analyses involve these methods. Table 8 shows that most instances with up to 50 activities and tight project deadlines are solved to optimality. Note that no algorithm known from the literature has found optimal schedules for similar problem sizes or real-world instances within reasonable time.

Table 9 lists the average numbers of cuts added during optimization, where we consider both test sets, T_1, T_2, and the effective models, M_1 and M_4, and introduce GUB-cuts, clique, cover, and zero-half cuts.

The numbers of cuts depend on the number n of activities involved, the number $|\mathcal{R}|$ of resources, and the specified project deadlines $\bar{d}$. In particular, for model M_1, GUB-cuts and cover cuts, and for model M_4, cover and zero-half cuts, are created in the subproblems of the branch-and-bound tree. The total number of cuts generated for model M_1 significantly exceeded that for model M_4. Furthermore, fewer cuts are generated for test set T_1 in comparison to test set T_2, a result that might have been expected, since the solver finds optimal solutions faster for former instances than for later ones.

Table 8
Computation times and the numbers of instances solved (test set T_1, classical RLP and overload problem).

Instances	M_1		M_4	
	t_{cpu}	Inst<3h	t_{cpu}	Inst<3h
20-1-1.1	4.266	40	2.530	40
20-3-1.1	260.844	40	35.129	40
20-5-1.1	85.700	40	52.288	40
20-1-1.5	1245.811	36	86.157	39
20-3-1.5	2669.789	30	340.235	39
20-5-1.5	3074.423	20	1395.448	34
30-1-1.0	9.983	40	12.529	40
30-3-1.0	18.389	40	14.155	40
30-5-1.0	61.198	40	101.538	40
30-1-1.1	387.526	36	196.596	36
30-3-1.1	691.950	38	608.064	38
30-5-1.1	1808.033	33	112.820	36
50-1-1.0	530.358	38	334.413	37
50-3-1.0	675.812	36	926.266	36
50-5-1.0	969.605	32	288.888	28
# Opt.		539		563

Table 9
Average numbers of cuts for test sets T_1 and T_2.

	M_1				M_4			
Instances	GUB	Clique	Cover	Zero-half	GUB	Clique	Cover	Zero-half
T_1								
$n = 10$	159.2	57.2	82.2	10.6	4.9	7.7	6.8	34.8
$n = 20$	418.6	100.4	234.5	29.2	8.4	32.3	31.8	99.4
$n = 30$	487.3	149.8	372.0	45.0	36.0	66.5	100.2	140.9
T_2								
$n = 10$	360.2	42.0	136.2	17.6	6.0	13.3	9.7	71.4
$n = 15$	436.4	96.9	368.0	29.8	21.6	47.7	70.9	118.5
$n = 20$	578.4	126.8	588.4	34.7	47.7	90.6	227.2	151.4
$n = 30$	625.8	169.6	1008.5	49.1	52.2	125.6	343.2	209.1
$n = 50$	789.8	130.5	1320.5	57.5	53.6	98.5	615.7	263.2

7. Conclusion

In this paper, we have devised new mixed-integer linear model formulations for the classical resource leveling problem and the overload problem. The models are based on smart discrete-time formulations. Small-scale and medium-scale problem instances could efficiently be solved using the standard solver CPLEX. In order to facilitate the solution process, we have added problem-specific preprocessing techniques and cutting planes. Within this context, we have restricted the domains of auxiliary variables and considered analytical techniques for linearizing the various objective functions. We have also identified GUB-cuts and clique, cover, and zero-half cuts as particularly suitable for our purposes.

Instances involving small numbers of resources and tight project deadlines need significantly less computing time in order to receive optimal schedules than instances involving large numbers of resources and an extended deadline. However, instances with 50 activities and small deadlines are solved to optimality for the first time. Run times for instances contained in test set T_1 are often shorter than those for instances contained in test set T_2, because T_2 usually involves higher values for the resource factor, *RF*. Since the solver is able to find optimal solutions within acceptable time periods, the models seem to be suitable for solving real-world applications.

An important area for future research is the study of further resource leveling objective functions, e.g. considerations of increments in resource requirements from period $t-1$ to period t. Moreover, investigating hybrid heuristics for solving resource leveling problems would appear to be a suitable future endeavor.

Acknowledgements

The benchmarks for the resource leveling problems presented herein and the results obtained (i.e. upper and lower bounds) may be downloaded from http://www.wiwi.tu-clausthal.de/abteilungen/unternehmensforschung/forschung/.

References

Ahuja, H.N., 1976. Construction Performance Control by Networks. Wiley, New York.
Ahuja, R., Magnanti, T., Orlin, J., 1993. Network Flows. Prentice Hall, Englewood Cliffs.
Artigues, C., Michelon, P., Reusser, S., 2003. Insertion techniques for static and dynamic resource-constraint project scheduling. European Journal of Operational Research 149 (2), 249–267.

Ballestin, F., Schwindt, C., Zimmermann, J., 2007. Resource leveling in make-to-order production: modeling and heuristic solution method. International Journal of Operations Research 13 (2), 76–83.

Bandelloni, M., Tucci, M., Rinaldi, R., 1994. Optimal resource leveling using non-serial dynamic programming. European Journal of Operational Research 78, 162–177.

Bartusch, M., Möhring, R., Radermacher, F., 1988. Scheduling project networks with resource constraints and time windows. Annals of Operations Research 16, 201–240.

Burgess, A., Killebrew, J., 1962. Variation in activity level on a cyclical arrow diagram. Journal of Industrial Engineering 13, 76–83.

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

Das Gupta, O., 2009. Alte Atomkraftwerke – Die Gelddruckmaschinen. Süddeutsche Zeitung 06.07.2009.

Demeulemeester, E., Herroelen, W., 2002. Project Scheduling: A Research Handbook. Kluwer, Bosten.

Easa, S.M., 1989. Resource leveling in construction by optimization. Journal of Construction Engineering and Management 115, 302–316.

Eidgenössisches Nuklearsicherheitsinspektorat, E., 2009. Revision von Kernkraftwerken. ENSI Newsletter 4 (Themenheft).

Gabow, H., Myers, E., 1978. Finding all spanning trees of directed and undirected graphs. SIAM Journal on Computing 7, 208–287.

Gather, T., 2011. Exakte Verfahren für das Ressourcennivellierungsproblem. Gabler, Wiesbaden.

Gather, T., Zimmermann, J., Bartels, J.-H., 2011. Exact methods for the resource levelling problem. Journal of Scheduling 14 (6), 557–569.

Habib, M., Morvan, M., Rampon, J.X., 1993. On the calculation of transitive reduction-closure of orders. Discrete Mathematics 111, 289–303.

Hagmayer, S., 2006. Struktur von Projektplanungsproblemen aus polyedertheoretischer Sicht. Shaker, Aachen.

Harris, R.B., 1990. Packing method for resource leveling (pack). Journal of Construction Engineering and Management 116, 39–43.

Ilg, P., 2009. Wie Atomkraftwerke in Deutschland geprüft werden. Financial Times Deutschland 20.10.2009.

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

Kelly, J.E., 1961. Critical-path planning and scheduling: mathematical basis. Operations Research 9, 296–320.

Kolisch, R., Schwindt, C., Sprecher, A., 1999. Benchmark instances for project scheduling problems. In: Węglarz, J. (Ed.), Project Scheduling: Recent Models, Algorithms, and Applications. Kluwer, Boston, pp. 197–212.

Koné, O., Artigues, C., Lopeza, P., Mongeauc, M., 2011. Event-based milp models for resource-constrained project scheduling problems. Computers and Operations Research 38, 3–13.

Möhring, R.H., 1984. Minimizing costs of resource requirements in project networks subject to a fixed completion time. Operations Research 32, 89–120.

Neumann, K., Zimmermann, J., 1999. Methods for resource-constrained project scheduling with regular and nonregular objective functions and schedule-dependent time windows. In: Węglarz, J. (Ed.), Project Scheduling: Recent Models, Algorithms, and Applications. Kluwer, Boston, pp. 261–287.

Neumann, K., Zimmermann, J., 2000. Procedures for resource levelling and net present value problems in project scheduling with general temporal and resource constraints. European Journal of Operations Research 127, 425443.

Neumann, K., Nübel, H., Schwindt, C., 2000. Active and stable project scheduling. Mathematical Methods of Operations Research 52, 441–465.

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

Nübel, H., 1999. Minimierung der Ressourcenkosten für Projekte mit planungsabhängigen Zeitfenstern. Gabler, Wiesbaden.

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.

Schreiber, H., 2007. Eine Revision der Rekorde im Kernkraftwerk. Augsburger Allgemeine Zeitung 26.10.2007.

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

Schwindt, C., 2005. Resource Allocation in Project Management. Springer, Berlin.

Selle, T., 2002. Untere Schranken für Projektplanungsprobleme. Shaker, Aachen.

Younis, M.A., Saad, B., 1996. Optimal resource leveling of multi-resource projects. Computers and Industrial Engineering 31, 1–4.