Flex Serv Manuf J (2013) 25:74–93
DOI 10.1007/s10696-012-9141-8

Project scheduling with resource capacities and requests varying with time: a case study

Sönke Hartmann

Published online: 11 February 2012

Abstract This paper discusses an extension of the classical resource-constrained project scheduling problem in which the resource availability as well as the resource request of the activities may change from period to period. While the applicability of this extension should be obvious, we provide a case study in order to emphasize the need for the extension. A real-world medical research project is presented which has a structure that is typical for many other medical and pharmacological research projects that consist of experiments. Subsequently, we provide a mathematical model and analyze some properties of the extended problem setting. We also introduce a new priority rule heuristic that is based on a randomized activity selection mechanism which makes use of so-called tournaments. Finally, we report our computational results for the original data of the medical research project as well as for a set of systematically generated test instances.

Keywords Project management · Project scheduling · Temporal constraints · Resource constraints · Priority rule · Tournament heuristic · Computational results · Case study

1 Introduction

The classical resource-constrained project scheduling problem (RCPSP) can be summarized as follows: A project consists of J activities labeled $j = 1, \ldots, J$. Usually two additional activities $j = 0$ and $j = J + 1$ represent the start and the end

Sönke Hartmann–Supported by the HSBA Foundation.

S. Hartmann (✉)
HSBA Hamburg School of Business Administration, Alter Wall 38, 20457 Hamburg, Germany
e-mail: soenke.hartmann@hsba.de
URL: www.hsba.de

of the project, respectively. Each activity j is associated with a processing time (or duration) p_j during which no interruption is allowed. An j activity may start once its predecessors which are given by the set P_j are finished. The set of successors of activity j is denoted as S_j. The resulting precedence network is assumed to be acyclic. Resources are needed to carry out the activities. K resources are given. Activity j requires r_{jk} units of resource k in each period of its processing time. In each period of the planning horizon, R_k units of resource k are available. The goal is to determine a start and a finish time for each activity such that the makespan of the project is minimized.

The origins of the RCPSP can be traced back to the late 1960s (Pritsker et al. 1969). Since then it has become a popular standard problem in operations research. Several exact (Demeulemeester and Herroelen 1992; Mingozzi et al. 1998; Sprecher 2000) and many heuristic methods (Kolisch 1999; Kolisch and Hartmann 2006) have been developed for the RCPSP. In addition, various extensions of the basic RCPSP have been developed (for overviews see Brucker et al. 1999 and Hartmann and Briskorn 2010). Among the most popular extensions are multiple modes for the activities (Talbot 1982; Hartmann and Drexl 1998; Hartmann 2001), generalized precedence constraints (Demeulemeester and Herroelen 1996; Dorndorf et al. 2000; Neumann et al. 2002, 2003; Bianco and Caramia 2011) and alternative objectives (Kimms 2001; Möhring et al. 2003; Neumann and Zimmermann 2000).

In this paper, we discuss an extension of the RCPSP that has not yet received the attention that it deserves. Whereas the standard RCPSP assumes that resource capacities and requests are constant over time, we consider resource capacities and requests that may change from period to period. Sprecher (1994) and Hartmann (1999) mention time-varying resource parameters but do not provide dedicated solution methods. Bartusch et al. (1988) remark that time-dependent resource availabilities and requests can be captured by an RCPSP with generalized precedence constraints (cf. also de Reyck et al. 1999). According to Bartusch et al. (1988), time-dependent resource availabilities and requests can be transformed into constant ones by defining dummy activities which then have to be stitched together using minimal and maximal time lags.

Several other researchers discuss approaches for generalized resource constraints. A few papers (Klein 2000; Sprecher and Drexl 1998) consider time-dependent resource capacities but constant demand. Poder et al. (2004) propose a model that has some similarity with the RCPSP. The activity durations are not fixed, and their resource requests are given as a continuous function over time. Solution methods are not discussed. Some related research has been carried out for project scheduling with labor constraints: Cavalcante et al. (2001) consider a problem similar to the RCPSP in which a single resource (labor) with constant availability but time-dependent requests is given. Drezet and Billaut (2008) discuss time-dependent requests for labor resources, but their model contains several specific constraints (e.g., legal requirements concerning the working time of employees) such that it is considerably different from the classical RCPSP. The calender concept of Franck et al. (2001) takes changes in the resource availability into account. There, activities are interrupted during periods in which a resource is unavailable; afterwards, the

activities are resumed. This increases the activity duration in terms of elapsed time between start and finish.

Our purpose in this paper is to lay a foundation for research on the RCPSP with time-dependent resource capacities and requests (we will refer to the extended model as RCPSP/t). To underscore the practical relevance of this problem setting, we first describe a real-world medical research project where the activities correspond to experiments and the resources are laboratory staff and equipment. This project is typical for other projects in medical and pharmacological research, and it can be fully captured using the concepts of the RCPSP/t. We then present a mathematical model for the RCPSP/t, examine its properties and sketch out a simple randomized heuristic based on priority rules. After briefly giving our computational results for the case study, we provide a pragmatic approach to generate test instances for the RCPSP/t and summarize our computational results based on test sets with different characteristics.

2 Problem setting

2.1 Case study: a medical research project

We consider a medical research project that was carried out at the medical faculty of the University of Kiel (Germany). A detailed description of this project as well as the results can be found in Löser et al. (1997). The goal of the project was to examine the relationship between polyamine synthesis and cancer. In what follows, we give a brief summary of those aspects of the project that are relevant for scheduling. Further details can be found in Hartmann (1999).

The project consists of a set of experiments. An experiment is essentially the analysis of a particular drug combination over a certain time. Each experiment has a fixed duration, the durations of the experiments range between 2 and 8 days. Once an experiment has been started, it may not be interrupted. Moreover, each experiment is repeated several times in order to allow a statistical evaluation. The number of required repetitions varies from experiment to experiment.

The temporal arrangement of the repetitions of one experiment is restricted. On one hand, several repetitions should be carried out in parallel, that is, they should start (and thus also finish) on the same day. The main advantage of this is that it allows the researcher to dose the medication more accurately. On the other hand, performing too many repetitions of an experiment in a parallel block may cause systematic errors. Especially the last day of an experiment (i.e., the day on which the analysis takes place) is assumed to be critical in this sense. Therefore, the repetitions of one experiment should finish on a certain number of different days which is given for each number of repetitions. Moreover, the repetitions of an experiment should be distributed over the repetition blocks as evenly as possible. These requirements can be captured as follows: Each experiment corresponds to a number of activities, and each such activity represents a number of repetitions which must start on the same day. Additional temporal constraints must be

considered to ensure that the activities related to one experiment finish on different days.

Two types of resources have to be taken into account. The first one is the researcher who carries out all the experiments of the project. The researcher specifies the days he or she is in the laboratory, typically Monday through Friday plus certain weekends. When he is in the laboratory, the number of repetitions he can handle at the same time is limited. Some experiments (and thus the related activities) require the presence of the researcher on every day, others only on certain days. The number of resource units required by an activity corresponds to the number of repetitions. The second resource is the laboratory equipment. It is available for this project only on certain predetermined days. In this particular project, an activity needs the laboratory equipment only on the last day of its duration (again, the number of units required corresponds to the number of repetitions).

The researcher is responsible for determining a project schedule which observes the restrictions given above. His objective is an early project completion. This also leads to free laboratory capacities for further research projects.

To illustrate the project and how it can be captured using RCPSP concepts, we consider the following example: The researcher can handle, say, 20 repetitions at the same time, so he corresponds to a resource with a capacity of 20 on the days he is in the laboratory and 0 otherwise. The laboratory equipment allows to carry out the analysis of at most 6 repetitions per day, hence it is reflected by a resource with a capacity of 6 when it is available for this project and 0 on other days. An experiment with 8 repetitions is transformed into 3 activities, with 2 activities corresponding to 3 repetitions each and one corresponding to 2 repetitions (this way the distribution of repetitions among the activities is as even as possible). These 3 activities must not finish on the same day. If the experiment has a duration of 7 days, then the duration of the three resulting activities is 7 as well. The activity that corresponds to the 2 repetitions requires 2 units of the researcher resource on days 1, 5, 6 and 7 and 0 units on days 2, 3 and 4. Moreover, this activity needs 2 units of the laboratory equipment on day 7 and 0 units on days 1 through 6.

This description shows that the standard RCPSP concepts such as non-preemptable activities with fixed durations, resources and makespan minimization can be applied here. There are, however, two aspects that cannot be reflected within the classical RCPSP: The resource capacities and requests in this project are not constant over time, and certain activities are not allowed to finish on the same day. Also note that standard precedence constraints of the RCPSP are not used in this particular example, although they might of course be present in other cases.

2.2 Formal problem definition

In this section, we present a model that extends the standard RCPSP by two concepts, namely time-dependent resource availability and request as well as a new type of temporal constraints. The resulting model covers the medical research project described in Sect. 2.1. Beyond that, it can be considered a very general model with many potential applications.

In a first step, we extend the standard RCPSP by generalizing the resource constraints. As in the standard RCPSP, we consider J activities with non-preemptable processing time (or duration) p_j, predecessor set P_j and successor set S_j for each activity j. For notational convenience, we also need the transitive closure of the successor relation. $\overline{S}_j$ represents the set of immediate and non-immediate successors of activity j. Two additional activities $j = 0$ and $j = J + 1$ are dummy activities with $p_0 = p_{J+1} = 0$ and mark the start and the end of the project, respectively.

K resources are given. The capacity of resource $k = 1, \ldots, K$ in period $t = 1, \ldots, T$ is denoted as R_{kt}, where T is the planning horizon. Each activity j requires r_{jkt} units of resource k in the tth period of its processing time, $t = 1, \ldots, p_j$. We assume the parameters (durations, capacities, resource requests) to be nonnegative and integer valued. The objective is to determine a schedule (i.e., a start time for each activity) with minimal makespan such that both the temporal and the resource constraints are fulfilled.

Following Pritsker et al. (1969), we define binary decision variables x_{jt} for each activity $j = 0, \ldots, J + 1$ and each period $t = 0, \ldots, T$ by

$$x_{jt} = \begin{cases} 1, & \text{if activity } j \text{is finished at the end of period } t \\ 0, & \text{otherwise.} \end{cases}$$

We obtain the following mathematical model for the resource-constrained project scheduling problem with time-dependent resource availabilities and requests (RCPSP/t). It is a rather straightforward extension of the model first presented by Pritsker et al. (1969).

$$\text{Minimize} \quad \sum_{t=0}^{T} t \cdot x_{J+1,t} \tag{1}$$

subject to

$$\sum_{t=0}^{T} x_{jt} = 1 \qquad j = 0, \ldots, J+1 \tag{2}$$

$$\sum_{t=0}^{T} t \cdot x_{ht} \leq \sum_{t=0}^{T} (t - p_j) \cdot x_{jt} \qquad j = 0, \ldots, J+1, \quad h \in P_j \tag{3}$$

$$\sum_{j=1}^{J} \sum_{q=t}^{t+p_j-1} r_{j,k,t+p_j-q} \cdot x_{jq} \leq R_{kt} \qquad k = 1, \ldots, K, \quad t = 1, \ldots, T \tag{4}$$

$$x_{jt} \in \{0, 1\} \qquad j = 0, \ldots, J+1, \quad t = 0, \ldots, T \tag{5}$$

Objective (1) minimizes the finish time of the dummy sink activity and, therefore, the project's makespan. Constraints (2) secure that each activity is executed exactly once, while constraints (3) take care of the standard precedence relations. Constraints (4) reflect the time-dependent resource restrictions. Finally, constraints (5) define the binary decision variables.

In a second step we take a look at the new temporal constraints sketched out in Sect. 2.1. There, a group of activities may not finish at the same time. This can be captured as follows: Let a be the number of activity groups which are related to this type of constraint. Each set A_i with $i = 1, \ldots, a$ represents such an activity group, that is, any two activities $j, h \in A_i$ must be assigned different finish times. This can be reflected by adding the following constraints to the RCPSP/t model (1)–(5):

$$\sum_{j \in A_i} x_{jt} \leq 1 \qquad i = 1, \ldots, a, \quad t = 1, \ldots, T \tag{6}$$

Note, however, that constraints (6) are special case of the time-dependent resource constraints (4). In fact, each set A_i can be modeled as an additional resource k with a constant availability $R_{kt} = 1$ for all $t = 1, \ldots, T$. The time-dependent request of each activity $j \in A_i$ for this new resource k would be defined as

$$r_{jkt} = \begin{cases} 0 & \text{for } t = 1, \ldots, p_j - 1 \\ 1 & \text{for } t = p_j. \end{cases}$$

Consequently, the RCPSP/t as given by (1)–(5) fully covers medical research projects as described in Sect. 2.1. It should be mentioned that also other variants of the temporal constraints introduced here can be included in a similar way, especially the restriction that activities are not allowed to start at the same time or are not allowed to overlap at all. Such temporal constraints [which, unlike the standard precedence constraints (3), do not impose a partial order on the activities] are special cases of the resource constraints (4) as well.

Finally, we should point to another property of the RCPSP/t. Due to the variation over time in the resource constraints, there might not be a feasible solution within the given planning horizon, even if the planning horizon is long (e.g., the sum of all activity durations). This distinguishes the RCPSP/t from the classic RCPSP, for which a feasible solution exists (provided that $r_{jk} \leq R_k$ for each activity j and each resource k).

2.3 Time windows and lower bound

For the standard RCPSP, a simple procedure allows to derive time windows for the activities. We assume that the activities are labelled in a way that each activity has a higher number than any of its predecessors. The earliest start time of the start activity is defined as $ES_0 = 0$. Now the earliest start time of activity $j = 1, \ldots, J + 1$ is the maximum of the earliest finish times of its predecessors, that is, $ES_j = \max\{ES_i + p_i \mid i \in P_j\}$. The latest finish time of the end activity is equal to the planning horizon, so we have $LF_{J+1} = T$. Then the latest finish time of activity $j = J, \ldots, 0$ is the minimum of the latest start times of its successors, that is, $LF_j = \min\{LF_i - p_i \mid i \in S_j\}$. Any activity j must be processed within the time window $[ES_j, LF_j]$, otherwise the project would not be finished within the planning horizon.

These time windows can be applied to the RCPSP/t without modification, but it is also possible to extend the procedure above by taking the time-dependent resources into account. ES_j^* then is the earliest feasible start time with regard to the precedence constraints and the resource request of activity j. Here, resource feasibility means

that we have $r_{j,k,t-ES_j^*+1} \leq R_{kt}$ for each time $t = ES_j^*, \ldots, ES_j^* + p_j - 1$ and each resource k. Precedence feasibility implies that $ES_j^* \geq \max\{ES_i^* + p_i \mid i \in P_j\}$ holds. Likewise, LF_j^* is the latest feasible finish time with regard to the precedence relations and the resource demand of activity j.

With this extended (but still simple) procedure we exploit the time-dependent availabilities. By excluding start and finish times with insufficient resources, the time windows become tighter. Note that for the standard RCPSP (and for the RCPSP/t with sufficient capacities over time for each single activity) we have $ES_j^* = ES_j$ and $LF_j^* = LF_j$. Also observe that $LF_j^* - ES_j^* < p_j$ may occur if the resources are scarce. In such a case there is no feasible solution.

There are several applications of these time windows which will be used in the remainder of this paper. First, the latest finish times and the latest start times are often used in priority rule heuristics. Second, the earliest finish time of the end activity is a lower bound on the project's makespan which can be used to evaluate heuristics if the optimal makespan is not known. To avoid ambiguity, we refer to the lower bound according to the standard procedure as LB while the extended lower bound is denoted as LB/t. Third, the time windows allow to reduce the number of variables in the mathematical programming formulation (cf., e.g., Hartmann 1999).

3 Priority rule heuristics

3.1 Schedule generation scheme

Many heuristics for project scheduling problems are based on a so-called schedule generation scheme (SGS). An SGS schedules one activity in each step until a complete schedule is constructed. In this process, it controls the calculation of the set of those activities that are eligible for scheduling, and it guides the start time computation for a selected activity. The activity selection itself, however, is based on the underlying heuristic approach; it can be done by, e.g., a priority rule or a genetic representation. Before we turn to a heuristic for the RCPSP/t, we take a brief look at the properties of SGS when applied to the RCPSP/t.

Two SGS are available for the standard RCPSP, namely the parallel SGS which is based on time-incrementation and the serial SGS which is based on activity-incrementation (for detailed descriptions see Kolisch and Hartmann 1999). The parallel SGS operates on the set of non-delay schedules whereas the serial SGS constructs active schedules (an active schedule is a schedule in which no activity can be left-shifted, for non-delay schedules cf. Sprecher et al. 1995). For the RCPSP, the search space of the parallel SGS might not contain any optimal solution, whereas the search space of the serial SGS always contains an optimal solution (cf. Sprecher et al. 1995).

The two SGS are applicable to the RCPSP/t as well. We now examine the behavior of the serial SGS in the presence of time-dependent resource availabilities and requests. Roughly speaking, the serial SGS works as follows: In each step, it

selects an eligible activity and schedules it at the earliest feasible time (an activity is called eligible if all its predecessors have already been scheduled). The findings for the serial SGS are summarized in the following remark which highlights some structural similarities and differences between the RCPSP and the RCPSP/t.

Remark 1 For the RCPSP/t the following propositions hold:

(i) All schedules constructed by the serial SGS are active ones.
(ii) For some instances there are active schedules which cannot be generated by the serial SGS.
(iii) For some instances the serial SGS might not find an existing optimal solution.

Since the serial SGS schedules all activities at the earliest feasible time, all schedules are active ones, which is of course the same for the RCPSP and the RCPSP/t. Unlike for the RCPSP, however, the serial SGS does not produce *all* active schedules for the RCPSP/t and might thus miss an optimal solution. This is shown by the following counterexample: We have $J = 2$ activities with processing times $p_1 = p_2 = 2$. There are no precedence relations. The planning horizon is $T = 4$ periods. We have a single resource with time-dependent capacity

$$R_{1t} = \begin{cases} 2, & \text{if } t \in \{1, 2, 4\} \\ 4, & \text{if } t = 3. \end{cases}$$

The time-varying resource requirements of the two activities are given by

$$r_{11t} = r_{21t} = \begin{cases} 1, & \text{if } t = 1 \\ 2, & \text{if } t = 2. \end{cases}$$

Now consider the schedules shown in Fig. 1. Schedule (a) is active as no left-shift is possible, and it is optimal with a makespan of 3 periods. The serial SGS will schedule either activity 1 first and then 2, or 2 first and then 1. Since the serial SGS schedules an activity as early as possible, the two solutions that can be found by the serial SGS are those of Fig. 1b and c, respectively. Obviously, both are active, but none of them is optimal.

Nevertheless, we will use the serial SGS in our priority rule heuristics for the RCPSP/t. It is a simple and effective method, and in fact it seems reasonable to

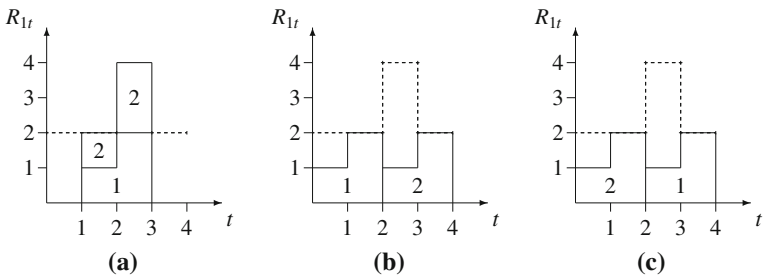

Fig. 1 Schedules for the counterexample instance

schedule activities as early as possible given that the goal is to minimize the makespan. While certainly not desirable, reducing the search space and thus excluding all optimal schedules needs not be a severe issue: A smaller search space may in fact focus on schedules of good average quality, even if it might not contain an optimal solution. In other words, the risk of eventually excluding the optimal schedules can be more than compensated for by omitting many inferior schedules. This is supported by the the computational results of Kolisch and Hartmann (2006) for large instances of the standard RCPSP, where the parallel SGS clearly outperforms the serial one although it might exclude all optimal schedules. After all, we consider priority rule methods here, and such heuristics are employed for finding good solutions very quickly (and not for fine-tuned long-term optimization). With this in mind, the choice of the serial SGS for the RCPSP/t seems to be appropriate. Thus, the serial SGS will be employed in the heuristic introduced below.

3.2 A multi-pass heuristic based on tournaments

Being simple and fast, priority rule based heuristics are popular methods to find acceptable solutions quickly. They are applied either to find schedules or, if the schedule quality is of particular importance, to find starting solutions for metaheuristics.

Many different priority rule based heuristics have been developed for the standard RCPSP, see Kolisch and Hartmann (1999). Essentially, such heuristics follow an SGS and construct a schedule step by step. In each step a priority rule decides which activity is scheduled. Of particular importance are multi-pass priority rule based methods which apply the SGS multiple times and thus lead to several schedules from which the best can be picked. A common way to obtain a different schedule in each pass is to randomize the selection of the next activity to be scheduled. There are several concepts for randomizing the activity selection. Among the most popular ones is the calculation of selection probabilities based on the values of the priority rule. The higher the priority of an activity, the higher its probability to be selected. Quite a few ways to transform the activity priorities into selection probabilities have been proposed, see Kolisch and Hartmann (1999).

In what follows, we introduce a multi-pass priority rule based heuristic based on the serial SGS. It makes use of a new, simple and intuitive concept for randomizing the activity selection. Unlike many methods in the literature, this concept is not based on the explicit computation of selection probabilities. Instead, it employs the idea of tournament selection which originally is an approach to select individuals from the population of a genetic algorithm (Goldberg and Deb 1991; Michalewicz 1995).

Generally, the idea of a tournament is as follows: Z individuals are picked from a population of n individuals (either with or without replacement). Thereby, all individuals have the same probability to be picked. Then the fittest of these Z individuals has "won the tournament" and is selected. This way, higher fitness values imply higher probabilities to be selected. The tournament size implicitly influences the selection probability: The larger the tournament size Z, the stronger

the selective pressure, that is, the more likely the selection of an individual with a very high fitness.

This idea can be used for randomized activity selection within an SGS. The individuals correspond to the eligible activities and the individual's fitness corresponds to the activity's value derived from a given priority rule. Hence, in each step of the SGS we determine the next activity to be scheduled by picking Z activities from the eligible set E at random (i.e., each activity has the same probability $\frac{1}{|E|}$ to be picked) and then select the one with the best priority value.

Note, however, that the selective pressure reflected by Z must be seen in relation to the number of eligible activities. A fixed value for Z will be very selective for a small eligible set and not very selective for a large eligible set. The size of the eligible set changes during the steps of the SGS (and also depends on the project size and the precedence relations in the first place). Thus, it would not be beneficial to fix Z as this would not allow to control the selective pressure effectively. To solve this issue, we introduce a factor $\varphi \in [0, 1]$ that determines what fraction of the eligible set should be selected for a tournament. In addition, we stipulate that the tournament size should be at least two (this enforces a minimum selective pressure). Denoting the current eligible set as E, we obtain $Z = \max\{\varphi \cdot |E|, 2\}$ (values for Z are rounded). For example, with $|E| = 8$ eligible activities and a tournament factor of $\varphi = 0.4$ we obtain $Z = 0.4 \cdot 8 = 3.2$, which is rounded to $Z = 3$ eligible activities to be picked for the tournament.

The tournament heuristic based on the serial SGS can be summarized as displayed in Fig. 2. The approach is completed by the definition of a priority rule (see Sect. 3.3), a tournament factor φ, and a stopping criterion. The stopping criterion could be a maximum number of passes, a maximum number of consecutive passes without improvement of the best current solution, or a time limit. Throughout this paper, we use a maximum number of passes (i.e., number of schedules allowed to be generated) as stopping criterion.

3.3 Priority rules

To complete the tournament heuristic proposed in the previous subsection, we have to define a priority rule. While most rules developed for the standard RCPSP could be used, we outline a priority rule that takes the specific characteristics of the RCPSP/t into account. We will refer to the rule as CPRU, which stands for *critical path and resource utilization*.

The CPRU priority rule consists of two parts. The first part focuses on the precedence relations and activity durations. For a given activity j, we calculate the associated critical path length $CP_j = T - LS_j^*$, that is, the planning horizon minus the latest start time according to Sect. 2.3. Clearly, it makes sense to give priority to an activity with a long critical path, that is, a large CP_j value.

The second part puts the emphasis more on the time-dependent resource availabilities and demands. For a given activity j, we first determine for each resource the cumulated demand during the activity's duration divided by the cumulated availability during the activity's time window. Then we calculate the

set best current makespan $M^* := \infty$
while stopping criterion is not met **do**
  let the set of scheduled activities be empty, $SA := \emptyset$
  for $i := 0$ to $J+1$ **do**
    determine eligible set $E := \{j \in \{0,1,...,J+1\} \setminus SA \mid P_j \subseteq SA\}$
    set $Z := \max\{\varphi \cdot |E|, 2\}$
    set $E' := \emptyset$
    for $l := 1$ to Z **do**
      randomly pick an activity $j \in E$ with $p(h) = \frac{1}{|E|}\ \forall h \in E$
      set $E' := E' \cup \{j\}$
    end
    select the activity $j \in E'$ with the best priority value
    schedule j at the earliest precedence and resource feasible time s_j
    set $SA := SA \cup \{j\}$
  end
  if $s_{J+1} + p_{J+1} < M^*$ **then**
    set $M^* := s_{J+1} + p_{J+1}$
    save current best schedule $s_0, s_1, \ldots, s_{J+1}$
  end
end

Fig. 2 Tournament heuristic based on the serial SGS

average over all resources. The resulting value reflects the degree to which the activity requires the resources within its time window. Summing up, the resource utilization of an activity j is defined as

$$RU_j = \frac{1}{K} \cdot \sum_{k=1}^{K} \frac{\sum_{t=1}^{p_j} r_{jkt}}{\sum_{t=ES_j^*}^{LF_j^*} R_{kt}}.$$

When selecting the next activity to be scheduled, we should take into account that not only an activity itself but also its successors might have large resource requirements. If an activity has many successors with large resource utilization level, it might be scheduled with a higher priority. Therefore, we calculate the extended resource utilization RU'_j of activity j that also takes the cumulated resource utilization of its successors into account. Thereby, ω_1 and ω_2 are the weights related to the activity and to its successors, respectively. We obtain

$$RU'_j = \omega_1 \cdot |\overline{S}_j| \cdot RU_j \ + \ \omega_2 \cdot \sum_{i \in \overline{S}_j} RU_i.$$

Finally, we multiply the critical path value and the extended resource utilization value, which yields the priority value of the CPRU rule. Thus, an activity has a high

priority if it is associated with a long critical path and substantial resource requirements. We have

$$CPRU_j = CP_j \cdot RU'_j.$$

4 Computational results

4.1 Results for original data of the real project

In what follows, we consider again the original data of the real-world medical research project of Sect. 2.1. This project consists of 25 experiments. The number of repetitions per experiment is between 2 and 9. Experiments with up to 4 repetitions were transformed 2 activities, while those with 5 or more were transformed into 3 activities. In total, 62 activities were defined (excluding start and end activity). The durations of the experiments and hence those of the activities range from 2 to 8 days.

Each experiment is associated with a constraint that the related jobs do not finish on the same day. This requirement is transformed into 25 resources with a constant per-period availability of one unit. The associated resource demand is time-dependent with a non-zero demand only in the last period (cf. Sect. 2.2).

Two further resources cover the laboratory equipment and the researcher. The availabilities of these two resources are varying over time, taking into account the laboratory conditions and the personal schedule of the researcher. For both resources, the availability corresponds to the number of repetitions that can be handled at the same time. The activities' requests for these two resources are varying over time, they are either 0 or correspond to the number of repetitions. The laboratory equipment resource is needed only in the last period of an activity (i.e., on the last day of an experiment). The request for the researcher resource over time depends on the actual experiment.

In all, this case study leads to a moderately-sized project with $J = 62$ non-dummy activities and $K = 27$ resources. While there are no precedence relations, many of the resource availabilities and requests are varying with time, which is the key feature of the RCPSP/t. Also additional temporal constraints which stipulate that certain activities may not finish on the same day are captured with this model.

The classical lower bound LB is only 8 days (recall that no precedence relations are given, hence LB corresponds to the longest activity duration). The new lower bound LB/t is 31 days. A detailed analysis of this particular instance has shown that a tight lower bound is 67 days (cf. Hartmann 1999). The proposed tournament heuristic usually leads to a makespan of 67 days when at least 100 passes are allowed (note that it is a non-deterministic method, hence several test runs were carried out with a limit of 100 passes, and almost 100% of these runs led to a project duration of 67 days). Taking the lower bound into account, we can conclude that 67 is the optimal solution. With 0.014 CPU seconds for 100 passes, the calculation times of the tournament heuristic were very moderate. Obviously, it was not a difficult task for this simple method to find an optimal schedule. It is a rather small

and easy project instance, and its main use here is the motivation of the problem setting and not an evaluation of the method.

Next, we have tested the capability of the mathematical model of Sect. 2.2 to find an optimal solution for the case study instance. In order to reduce the number of binary variables, we proceeded as follows. First, we applied the tournament heuristic to find a good upper bound on the makespan. Then, based on this upper bound, we calculated the adapted time windows $[EF_j^*, LF_j^*]$. Now only variables x_{jt} with $j = 0, \ldots, J+1$ and $t = EF_j^*, \ldots, LF_j^*$ instead of $t = 1, \ldots, T$ are needed. We applied the GNU LP solver GLPK to find an optimal schedule, but, as expected, this was impossible. After more than 18 h we stopped the calculation without an optimal solution. This is, of course, in line with the experience from the standard RCPSP where projects of this size cannot be tackled with IP solvers. This also underscores the need for heuristics.

The real project schedule (which was made by hand by the researcher) had a makespan of 75 days. The schedule found by the tournament heuristic is 8 days or 10.7% shorter. However, this result must be interpreted with care because it was not possible to retrieve all details that led to the original project duration of 75 days.

4.2 Generation of test instances for the RCPSP/t

An in-depth analysis of the properties of the RCPSP/t and the behavior of a scheduling method is hardly possible based on just a single project instance. Therefore, this subsection is devoted to a pragmatic approach to generate test instances for the RCPSP/t. These test sets will then be used to evaluate the tournament heuristic in more detail.

Several instance generators for project scheduling have been proposed in the literature. In addition to the standard RCPSP, a few extensions such as multiple modes and maximal time lags have been considered; for a brief survey we refer to Hartmann and Briskorn (2010). ProGen of Kolisch et al. (1995) is probably the most widely used generator. The test sets generated by ProGen have become more or less a standard in the scientific community. They are available in the online project scheduling problem library PSPLIB, cf. Kolisch and Sprecher (1996).

To the best of our knowledge, time-dependent resource parameters are not considered by any of the generators. Therefore, we propose a method to generate test instances for the RCPSP/t. Rather than a whole new generator, however, we suggest to extend standard RCPSP instances by varying the originally constant resource availability and request. Our generator can be viewed as an add-on to ProGen: It reads a set of RCPSP instances in the ProGen format, adds variation to the resources and writes the resulting RCPSP/t instances in an extended ProGen format. The only changes of the format are the following:

- Each resource request r_{jk} is replaced by a list $r_{jk1}, \ldots, r_{jkp_j}$. Note that a dummy activity with a duration of $p_j = 0$ periods cannot request a resource by definition, hence the related lists are empty, that is, the output does not contain requests for dummy activities.

- Each resource capacity R_k is replaced by a list $R_{k1}, \ldots, R_{kT}$, with $T = \sum_j p_j$ being the sum of all durations.

The following parameters are employed to control the variation of the resource availabilities and requests. Probabilities P^R and P^r control whether or not a reduction is applied to the availability and the request, respectively. Factors F^R and F^r determine the strength of the reduction for the availability and the request, respectively.

Having read a standard RCPSP instance, the generator proceeds as follows. In each period $t = 1, \ldots, T$ of the planning horizon, the availability R_{kt} of resource $k = 1, \ldots, K$ is set to $R_k \cdot F^R$ with probability P^R and to R_k with probability $1 - P^R$, where R_k is the constant availability from the original ProGen file. The requests are defined analogously. For each activity $j = 1, \ldots, J$ and each period $t = 1, \ldots, p_j$, the request r_{jkt} for resource $k = 1, \ldots, K$ is set to $r_{jk} \cdot F^r$ with probability P^r and to r_{jk} with probability $1 - P^r$. Here, r_{jk} is the constant request from the original ProGen file.

The generator can operate in two modes. In the first mode, reductions are applied to periods (either of the horizon or of an activity duration) as a whole. That is, if it is decided that the capacity or demand is reduced in a period, this reduction is applied to all resources. In the second mode, it is decided for each resource separately whether or not the capacity or demand is reduced in a period. Preliminary experiments have shown that the second mode may lead to only a very few feasible start times for activities with long durations, which reduces the degrees of freedom for scheduling. Therefore, we have used only the first mode for the experiments reported in the remainder of this paper.

Several sets of test instances have been generated. As a basis, we used the J30 and the J120 RCPSP test sets from the PSPLIB (Kolisch and Sprecher 1999). For each of these two, six test sets have been created. They are denoted as J30t1,. . .,J30t6 and J120t1,. . .,J120t6, respectively, where "t" indicates the time dependency and the number refers to the parameter setting for the calculation. Since there are 480 instances in the J30 set and 600 in the J120 set, we obtained $6 \cdot 480 = 2880$ RCPSP/t instances with $J = 30$ and $6 \cdot 600 = 3600$ with $J = 120$. The reduction probabilities have been varied between 0.05 and 0.2. Note that the probabilities are the same for availability and request, that is, $P^R = P^r$. The strength of the reduction is either to half of the original capacity or down to 0. The factors are the same for capacity and demand, hence we have $F^R = F^r$. The design of the test sets is displayed in Table 1.

Table 1 Parameter settings for generation of test sets

Set no.	1	2	3	4	5	6
P^R	0.05	0.1	0.2	0.05	0.1	0.2
P^r	0.05	0.1	0.2	0.05	0.1	0.2
F^R	0	0	0	0.5	0.5	0.5
F^r	0	0	0	0.5	0.5	0.5

4.3 Analysis of lower bounds

In what follows, we summarize the outcome of the computational experiments. Since optimal solutions for the new test instances of Sect. 4.2 are not yet known, deviations of the heuristic makespan from the optimal one cannot be given. Therefore, the results of the tournament heuristic will be given in terms of deviations from a lower bound.

Table 2 shows that the new lower bound LB/t is substantially larger than the classical lower bound LB. Hence, if the gap between upper and lower bound is examined, it clearly pays to make use of LB/t instead of LB. When the reduction probability P^R increases, there are more periods with a reduced resource availability. Then it becomes more difficult to find a feasible start time for an activity (especially if the latter has a long duration). Thus, the deviation of LP/t from LB increases. The reduction factors F^R and F^r play a similar role, where F^R appears to have a stronger impact. A factor of $F^R = 0$ implies a reduction to an availability of 0 resource units. This leads to more infeasible start times than $F^R = 0.5$ and thus to a higher deviation of LP/t from LB.

4.4 Results for generated test sets

In this section, we analyze the performance of the proposed tournament heuristic using the test sets introduced in the previous subsection. Two stopping criteria were used: The heuristic was stopped after 100 and 1000 schedules were calculated for an instance, respectively.

In a first step, we have a look at the tournament factor φ, which was varied between 0.1 and 0.9. The CPRU rule with weights $\omega_1 = 0.4$ and $\omega_2 = 0.6$ was selected. Table 3 shows that the results are best for medium values of the tournament factor; generally, $\varphi = 0.5$ gives good results. This is no surprise as a small tournament factor decreases the probability that an eligible job with a good priority value Ist selected, whereas a large tournament factor implies that the eligible job with the highest priority value is selected very often, which might restrict the search to a too narrow area of the search space. In other words, a medium selective pressure keeps the balance between focus on promising activities and wider exploration of the search space. Generally, however, the impact of φ seems to be rather small as long as no extreme settings are used, which indicates that the tournament method is robust in this regard. Based on these observations, $\varphi = 0.5$ was used in all further experiments.

Table 2 Deviation of new lower bound LB/t from classical lower bound LB

$P^R = P^r$	0.05	0.1	0.2	0.05	0.1	0.2
$F^R = F^r$	0	0	0	0.5	0.5	0.5
$J = 30$	23.2%	50.7%	114.6%	5.5%	12.5%	30.7%
$J = 120$	23.4%	48.9%	122.9%	2.5%	5.7%	13.6%

Table 3 Impact of selectiveness (average deviation from lower bound LB/t)

Test set	Passes	$\varphi = 0.1$	$\varphi = 0.3$	$\varphi = 0.5$	$\varphi = 0.7$	$\varphi = 0.9$
J = 30	100	12.6%	12.6%	12.4%	12.5%	12.7%
	1,000	11.6%	11.6%	11.6%	11.7%	11.9%
J = 120	100	37.9%	34.3%	33.6%	33.6%	33.8%
	1,000	35.3%	32.4%	32.0%	32.1%	32.4%

Next, we take a look at the impact of the choice of the priority rule within the tournament heuristic. We compare the proposed CPRU rule to several classic priority rules for the standard RCPSP. Unlike CPRU, the classic rules do not exploit the time dependency of the resources. The following rules are included: The random rule (RND) picks a random activity from the eligible set (thereby all eligible activities have the same probability to be picked). It is used as a benchmark. The minimum slack rule (MSLK, Davis and Patterson 1975) picks the activity j with the smallest slack time $LF_j - e_j$, where LF_j is the precedence-based latest finish time and e_j is the earliest precedence and resource feasible start time in the current partial schedule. The latest start time rule (LST, Alvarez-Valdes and Tamarit 1989) chooses the activity j with the smallest latest start time $LS_j = LF_j - p_j$. LST was among the best performing RCPSP rules in the study of Kolisch (1996). The very similar latest finish time rule (LFT, Davis and Patterson 1975) is not considered here since it yielded consistently worse results than LST. Table 4 summarizes the results for the set with $J = 30$ while Table 5 does so for the set with $J = 120$. Again, the two stopping criteria of 100 and 1000 schedules have been employed.

The results are reported in terms of average deviation from the new lower bound LB/t. For both the $J = 30$ and $J = 120$ set, the results are given separately for each subset and for the entire set. We observe that, as expected, the RND rule leads to the worst results since it does not add any intelligence to the schedule generation

Table 4 Results of tournament heuristic for $J = 30$

$P^R = P^r$		0.05	0.1	0.2	0.05	0.1	0.2	All		
$F^R = F^r$		0	0	0	0.5	0.5	0.5	All		
Passes	Rule	Average deviation from lower bound LB/t							Feasible	CPU-s
100	RND	13.8%	10.2%	6.5%	17.0%	18.3%	17.4%	13.9%	98.2%	0.002
	MSLK	12.9%	9.8%	6.2%	16.0%	17.0%	16.3%	13.1%	98.2%	0.003
	LST	12.3%	9.1%	5.9%	15.2%	15.9%	15.3%	12.4%	98.2%	0.002
	CPRU	12.3%	9.1%	6.0%	15.2%	16.1%	15.3%	12.4%	98.2%	0.002
1000	RND	12.2%	9.2%	6.1%	15.4%	16.2%	15.3%	12.5%	98.3%	0.018
	MSLK	11.9%	9.0%	5.8%	14.8%	15.6%	14.7%	12.0%	98.2%	0.027
	LST	11.5%	8.7%	5.7%	14.4%	15.0%	13.9%	11.6%	98.2%	0.019
	CPRU	11.5%	8.7%	5.7%	14.2%	15.0%	14.0%	11.6%	98.3%	0.019

Table 5 Results of tournament heuristic for $J = 120$

$P^R = P^r$		0.05	0.1	0.2	0.05	0.1	0.2	All	All	All
$F^R = F^r$		0	0	0	0.5	0.5	0.5	All	All	All
Passes	Rule	Average deviation from lower bound LB/t							Feasible	CPU-s
100	RND	40.5%	32.0%	20.3%	55.6%	57.4%	60.1%	44.3%	100.0%	0.009
	MSLK	37.4%	29.8%	18.7%	51.3%	53.7%	56.7%	41.3%	100.0%	0.021
	LST	30.4%	23.5%	15.2%	43.6%	44.7%	47.2%	34.1%	100.0%	0.012
	CPRU	30.5%	23.5%	14.2%	43.2%	44.0%	46.1%	33.6%	100.0%	0.013
				***	**	*	***	***		
1000	RND	37.5%	29.3%	17.2%	51.8%	53.6%	56.2%	40.9%	100.0%	0.086
	MSLK	35.2%	27.7%	16.8%	48.8%	50.5%	53.3%	38.7%	100.0%	0.188
	LST	29.2%	22.4%	14.0%	41.9%	43.1%	45.2%	32.6%	100.0%	0.114
	CPRU	29.1%	21.9%	13.2%	41.4%	42.2%	44.2%	32.0%	100.0%	0.118
			***	***	***	***	***	***		

Significant difference between CPRU and LST with * $p < 0.05$, ** $p < 0.01$, *** $p < 0.001$.

scheme. MSLK performs better than RND, but the difference is not that big. LST is much better than MSLK, and the new CPRU rule is slightly better than LST. Note that the critical path part of the CPRU rule is closely related to the LST rule. Since CPRU performs slightly better than LST, we can conclude that it pays to add resource information to the critical path information.

Note, however, that for the $J = 30$ set the differences between the rules are small, and there are hardly any differences between CPRU and LST. For the set with $J = 120$, the differences are bigger, but those between CRPU and LST remain rather small. Consequently, we have carried out statistical tests to analyze the differences between CPRU and LST for this set. The Wilcoxon signed ranks test was applied (see Bowerman and O'Connell 2003). Table 5 shows that the difference between CPRU and LST is in fact clearly significant for the entire $J = 120$ test set as well as for most subsets, in many cases with $p < 0.001$. Also observe that the difference between CPRU and LST is greatest (and highly significant) for $P^R = P^r = 0.2$. That is, the CPRU rule is most promising if frequent changes in the resource availability and demand occur.

Let us now briefly discuss the percentage of project instances for which a feasible schedule can be found. Recall that it might not be possible to find a feasible solution within the horizon $T = \sum_j p_j$ because of the variation of resource capacity and request over time. Tables 4 and 5 indicate that there are no substantial differences between the priority rules. Table 6 looks at the subsets of the two test sets, considering only the CPRU rule. For the sets with $J = 120$, a feasible schedule is found for each instance, which is not the case for the set with $J = 30$. For $J = 30$, we observe that both more frequent ($P^R = 0.2$) and more drastic ($F^R = 0$) reductions of the resource availabilities make it harder to find feasible schedules within the horizon.

Table 6 Percentage of instances with feasible solution found (CPRU rule, 1000 passes)

$P^R = P^r$	0.05	0.1	0.2	0.05	0.1	0.2	All
$F^R = F^r$	0	0	0	0.5	0.5	0.5	All
$J = 30$	100.0%	100.0%	91.3%	100.0%	100.0%	98.3%	98.3%
$J = 120$	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%

Finally, we take a quick look at the computation times. The last column of Tables 4 and 5 shows the average CPU-seconds per instance when the heuristic stops after 100 and 1000 schedules, respectively. The calculation times are very moderate. The MSLK time are the highest because the earliest start time has to be calculated for each eligible activity, which is fairly time-consuming. The computer used for the experiments contains an Intel Core2 Duo CPU with 2.66 GHz and runs under Windows Vista (32 Bit).

5 Conclusions

In this paper, we have tackled the resource-constrained project scheduling problem with time-dependent resource capacity and demand (RCPSP/t) which is a straightforward extension of the classical RCPSP. The practical relevance has been demonstrated by means of a real medical research project. This case study also included a new type of temporal constraints. It was shown that these temporal constraints can be captured by the RCPSP/t, which further indicates the practical relevance of this problem setting.

Moreover, a new type of randomized priority rule heuristic has been proposed. It borrows the concept of tournament selection from genetic algorithms. The goal was to develop an alternative randomized approach that allows to adjust the selective pressure by means of an intuitive parameter. As the experience shows (Kolisch and Hartmann 2006), priority rule heuristics usually cannot compete with more elaborate approaches such as metaheuristics. Hence the application of this new method will most likely be the calculation of initial solutions for metaheuristics.

In order to conduct computational experiments, a large set of test instances has been generated. The starting point were widely used RCPSP test instances from the online library PSPLIB. These standard instances were extended to the RCPSP/t by adding variation to the originally constant resource availability and demand. The proposed generator includes parameters that allow to control the frequency and strength of this variation in a systematic manner. The resulting test sets will be made available to the scientific community in the PSPLIB online library. We hope that this helps to advance further research on the RCPSP/t which deserves attention already because of its practical relevance. Exact and more advanced heuristic methods are promising areas of future research on this problem setting.

Finally, it should be mentioned that the application of the tournament heuristic is not restricted to project scheduling. In fact, it can be used for any combinatorial optimization problem for which it makes sense to build up solutions step by step and

to pick one out of several alternatives in each step based on a priority rule. Nevertheless, its benefits are the simple structure and the intuitive parameter definition, while high quality heuristic solutions can usually not be expected from this type of heuristic.

Acknowledgement I would like to thank Thies Fitting for providing the relevant data of the medical research project (and for noticing that he had to deal with a difficult OR problem).

References

Alvarez-Valdes R, Tamarit JM (1989) Heuristic algorithms for resource-constrained project scheduling: a review and an empirical analysis. In: Słowiński R, Węglarz J (eds) Advances in project scheduling, Elsevier, Amsterdam, pp 113–134

Bartusch M, Möhring RH, Radermacher FJ (1988) Sched-uling project networks with resource constraints and time windows. Ann Oper Res 16:201–240

Bianco L, Caramia M (2011) A new lower bound for the resource-constrained project scheduling problem with generalized precedence relations. Comput Oper Res 38:14–20

Bowerman BL, O'Connell RT (2003) Business statistics in practice. McGraw-Hill, New York

Brucker P, Drexl A, Möhring R, Neumann K, Pesch E (1999) Resource-constrained project scheduling: Notation, classification, models, and methods. Eur J Oper Res 112:3–41

Cavalcante CCB, de Souza CC, Savelsbergh MWP, Wang Y, Wolsey LA (2001) Scheduling projects with labor constraints. Discrete Appl Math 112(1–3):27–52

Davis EW, Patterson JH (1975) A comparison of heuristic and optimum solutions in resource-constrained project scheduling. Manag Sci 21:944–955

de Reyck B, Demeulemeester EL, Herroelen WS (1999) Algorithms for scheduling projects with generalized precedence relations. In Węglarz, pp 77–106

Demeulemeester EL, Herroelen WS (1992) A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. Manag Sci 38:1803–1818

Demeulemeester EL, Herroelen WS (1996) Modelling setup times, process batches and transfer batches using activity network logic. Eur J Oper Res 89:355–365

Dorndorf U, Pesch E, Phan-Huy T (2000) A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. Manag Sci 46:1365–1384

Drezet L-E, Billaut J-C (2008) A project scheduling problem with labour constraints and time-dependent activities requirements. Eur J Oper Res 112((1):217–225

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

Goldberg DE, Deb K (1991) A comparative analysis of selection schemes used in genetic algorithms. In: Rawlins GJE (eds) Foundations of genetic algorithms, Morgan Kaufmann, San Mateo, pp 69–93

Hartmann S (2001) Project scheduling with multiple modes: a genetic algorithm. Ann Oper Res 102:111–135

Hartmann S (1999) Project scheduling under limited resources: models, methods, and applications. Number 478 in necture notes in economics and mathematical systems. Springer, Berlin

Hartmann S, Briskorn D (2010) A survey of variants and extensions of the resource-constrained project scheduling problem. Eur J Oper Res 207:1–14

Hartmann S, Drexl A (1998) Project scheduling with multiple modes: a comparison of exact algorithms. Networks 32:283–297

Kimms A (2001) Maximizing the net present value of a project under resource constraints using a lagrangian relaxation based heuristic with tight upper bounds. Ann Oper Res 102(1–4): 221–236

Klein R (2000) Project scheduling with time-varying resource constraints. Int J Prod Res 38:3937–3952

Kolisch R (1996) Serial and parallel resource-constrained project scheduling methods revisited: theory and computation. Eur J Oper Res 90:320–333

Kolisch R, Hartmann S (2006) Experimental investigation of heuristics for resource-constrained project scheduling: an update. Eur J Oper Res 174:23–37

Kolisch R, Hartmann S (1999) Heuristic algorithms for solving the resource-constrained project scheduling problem: classification and computational analysis. In Węglarz, pp 147–178

Kolisch R, Sprecher A (1996) PSPLIB – a project scheduling problem library. Eur J Oper Res 96:205–216
Kolisch R, Sprecher A, Drexl A (1995) Characterization and generation of a general class of resource-constrained project scheduling problems. Manag Sci 41:1693–1703
Löser C, Fitting T, Fölsch UR (1997) Importance of intracellular s-adenosyl-methionine decarb-oxy-lase activity for the regulation of camostate-induced pancreatic polyamine metabolism and growth: in vivo effect of two novel s-adenosyl-methionine decarb-oxy-lase inhibitors. Digestion 58:258–26
Michalewicz Z (1995) Heuristic methods for evolutionary computation techniques. J Heuristics 1:177–206
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. Manag Sci 44:714–729
Möhring RH, Schulz AS, Stork F, Uetz M (2003) Solving project scheduling problems by minimum cut computations. Manag Sci 49:330–350
Neumann K, Zimmermann J (2000) Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. Eur J Oper Res 127(2):425–443
Neumann K, Schwindt C, Zimmermann J (2002) Recent results on resource-constrained project scheduling with time windows: models, solution methods, and applications. Cent Eur J Oper Res 10:113–148
Neumann K, Schwindt C, Zimmermann J (2003) Project scheduling with time windows and scarce resources. Springer, Berlin
Poder E, Beldiceanu N, Sanlaville E (2004) Computing a lower approximation of the compulsory part of a task with varying duration and varying resource consumption. Eur J Oper Res 153:239–254
Pritsker AAB, Watters LJ, Wolfe PM (1969) Multiproject scheduling with limited resources: a zero-one programming approach. Manag Sci 16:93–107
Sprecher A (2000) Scheduling resource-constrained projects competetively at modest memory requirements. Manag Sci 46:710–723
Sprecher A (1994) Resource-constrained project scheduling: exact methods for the multi-mode case. Number 409 in lecture notes in economics and mathematical systems. Springer, Berlin
Sprecher A, Drexl A (1998) Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. Eur J Oper Res 107:431–450
Sprecher A, Kolisch R, Drexl A (1995) Semi-active, active and non-delay schedules for the resource-constrained project scheduling problem. Eur J Oper Res 80:94–102
Talbot FB (1982) Resource-constrained project scheduling with time-resource tradeoffs: the nonpre-emptive case. Manag Sci 28:1197–1210
Węglarz, J (eds) (1999) Project scheduling: recent models, algorithms and applications. Kluwer, Dordrecht

Author Biography

Sönke Hartmann received a diploma in computer science and a PhD in Operations Research from the University of Kiel, Germany. Subsequently, he was a logistics consultant for several years. The main focus of his projects was on optimization and simulation of container terminals. Since 2007, Sönke is professor of operations research and logistics at the HSBA Hamburg School of Business Administration. His research interests are in optimization algorithms and their applications as well as simulation of logistics processes. He has published in several scientific journals including European Journal of Operational Research, OR Spectrum and Naval Research Logistics.