



HSBA HAMBURG SCHOOL OF BUSINESS ADMINISTRATION

University of Applied Sciences

Working Paper No.: 01/2011

Sönke Hartmann

Project Scheduling with Resource Capacities and Requests Varying with Time

HSBA Hamburg School of
Business Administration
Adolphsplatz 1
20457 Hamburg · Germany
Tel. ++49 (0) 40-36 13 8-711
Fax ++49 (0) 40-36 13 8-751
www.hsba.de

WORKING PAPER SERIES

Project Scheduling with Resource Capacities and Requests Varying with Time

Sönke Hartmann*

HSBA Hamburg School of Business Administration
Alter Wall 38, D-20457 Hamburg, Germany
e-mail: soenke.hartmann@hsba.de
www.hsba.de

* supported by the HSBA Foundation

Abstract

This paper discusses an extension of the classical resource-constrained project scheduling problem (RCPSP) 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 discuss how priority rule based heuristics for the RCPSP can be applied to the extended problem. In addition to the priority rules themselves, we outline a framework for randomized priority rule methods. In order to provide a basis for experiments, we propose an adaptation of standard RCPSP test instances to the extended version of the problem. Finally, we report the computational results of the priority rule methods.

Keywords. Project Management and Scheduling, Temporal Constraints, Resource Constraints, Priority Rules, 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 $j = 1, \dots, J$. Usually two additional activities $j = 0$ and $j = J + 1$ represent the start and the end 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 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. [29]). Since then it has become a popular standard problem in operations research. Several exact (Demeulemeester and Herroelen [6], Mingozzi et al. [25], Sprecher [30]) and many heuristic methods (Kolisch and Hartmann [19, 20]) have been developed for the RCPSP. In addition, various extensions of the basic RCPSP have been developed (for overviews see Brucker et al. [3], Hartmann and Briskorn [14]). Among the most popular extensions are multiple modes for the activities (Talbot [34], Hartmann and Drexl [15], Hartmann [12]), generalized precedence constraints (Demeulemeester and Herroelen [7], Dorndorf et al. [8], Neumann et al. [28]) and alternative objectives (Kimms [16], Möhring et al. [26], Neumann and Zimmermann [27]).

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 varying over time. Bartusch et al. [2], Sprecher [31] and Hartmann [13] mention this problem setting but do not provide dedicated solution methods. A few papers (Klein [17], Sprecher and Drexl [32]) consider time-dependent resource capacities but constant demand. Some related research has been carried out for project scheduling with labor constraints: Cavalcante et al. [4] 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 [9] 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 calendar concept of Franck et al. [10] 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. Note that this increases the activity duration in terms of elapsed time between start and finish. Therefore, that approach is different from ours in which activity durations are fixed.

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 simple heuristics based on priority rules. Subsequently, 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. [23]. 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 [13].

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 as evenly distributed over the repetition blocks 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 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. An activity needs the laboratory equipment only on the last day of its duration (again, 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 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 Section 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 durations p_j , predecessor sets P_j and successor sets S_j . The 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 $k = 1, \dots, K$ in period $t = 1, \dots, T$ is denoted as R_{kt} , where T is the planning horizon. Each activity j has a non-preemptable processing time p_j and requires r_{jkt} units of resource k in the t -th period of its processing time, $t = 1, \dots, p_j$. 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. [29], we define binary decision variables x_{jt} for each activity $j = 0, \dots, J + 1$ and each period $t = 0, \dots, T$ 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. [29].

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

subject to

$$\sum_{t=0}^T x_{jt} = 1 \quad j = 0, \dots, J + 1 \quad (2)$$

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

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

$$x_{jt} \in \{0, 1\} \quad j = 0, \dots, J + 1, \quad t = 0, \dots, T \quad (5)$$

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 Section 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 $i = 1, \dots, a$ 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):

$$\sum_{j \in A_i} x_{jt} \leq 1 \quad i = 1, \dots, a, \quad t = 1, \dots, T \quad (6)$$

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 $t = 1, \dots, T$. The time-dependent request of each activity $j \in A_i$ for this new resource k would be defined as

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

Consequently, the RCPSP/t as given by (1)–(5) fully covers medical research projects as described in Section 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 $j = 1, \dots, J + 1$ 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 $j = J, \dots, 0$ 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 $t =$

$ES_j^*, \dots, ES_j^* + p_j - 1$ 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. 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 [13]).

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 heuristic approach; it can be done by, e.g., a priority rule or a genetic representation.

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 [20]). 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. [33]). 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. [33]).

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 schedules a selected 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 Figure 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 Figure 1 (b) and (c), respectively. Obviously, both are active, but none of them is optimal.

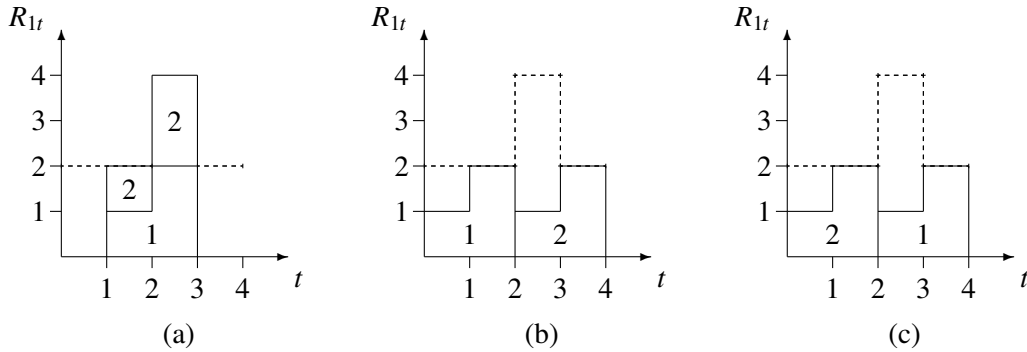


Figure 1: Schedules for the counterexample instance

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 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 computational results of Kolisch and Hartmann [19] 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 seems to be appropriate.

3.2 Priority Rules

Basic priority rule methods typically determine in each step of the SGS the eligible activities and select one of them according to a priority rule. The selected activity is then scheduled at the earliest feasible time. Various priority rules have been proposed for the RCPSP. In this paper, we consider some of the most popular ones and examine their performance when applied to instances of the RCPSP/t using the serial SGS. The following rules are tested:

- RND: The random rule picks a random activity from the eligible set. All eligible activities have the same probability to be picked. Clearly, this rule does not make use of any relevant information; it is included here merely as a benchmark.
- SPT: The shortest processing time rule (Alvarez-Valdes and Tamarit [1]) chooses the activity j with the shortest duration p_j . One might argue that this rule is not too promising for the standard RCPSP and even less so for the RCPSP/t as short activities might block segments in the resource profile which might otherwise contain sufficient resources for longer activities. Our goal is to evaluate this effect.
- LPT: The longest processing time rule selects the activity j with the longest duration p_j . While not particularly favorable for the standard RCPSP, it might be better suited for the RCPSP/t because it could be beneficial to fill long activities first into the fractioned resource profile.
- MSLK: The minimum slack rule (Davis and Patterson [5]) 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. This rule makes use of more project information than the previous ones.
- LFT: The latest finish time rule (Davis and Patterson [5]) selects the activity j with the smallest latest finish time LF_j . It was among the best performing rules for the standard RCPSP in the analysis of Kolisch [18].
- LST: The latest start time rule (Alvarez-Valdes and Tamarit [1]) chooses the activity j with the smallest latest start time $LS_j = LF_j - p_j$. Like the LFT rule, was been among the most promising rules in the study of Kolisch [18].

In all cases, the job number is used as a tie-breaker, that is, in case of a tie the activity with the smallest number is picked. All of these priority rules can be applied to the RCPSP/t without any modification. In addition, some of these rules can be adapted to take the time-dependency of the resources into account:

- LPT/t: Rather than counting all periods of an activity, this rule considers only those periods with a resource request > 0 . It selects the activity with the maximum number of periods with a nonzero demand for at least one resource. The idea is that it might be more difficult to place an activity into the resource profile if it requires resources over many periods.
- MSLK/t: This rule selects the activity j with the smallest slack time $LF_j^* - e_j$, where LF_j^* is the extended latest finish time of Section 2.3 and e_j is the earliest precedence and resource feasible start time in the current partial schedule.
- LFT/t: The extended latest finish time rule chooses the activity j with the smallest LF_j^* .

- LST/t: Analogously to LFT/t, this rule picks the activity with the smallest extended latest start time $LS_j^* = LF_j^* - p_j$.

3.3 A Multi-Pass Heuristic Based on Tournaments

Priority rules like those of Section 3.2 can be employed in two ways. First, they can be embedded into a deterministic method. A single pass of an SGS is executed, and in each step the eligible activity with the best priority value is scheduled. Second, priority rules can be applied within a randomized method. Multiple passes of the SGS are applied, in each pass one schedule is constructed. There are several concepts for randomizing the selection of the activity to be scheduled next. Among the most popular ones is the calculation of selection probabilities based on the values of the priority rule. Quite a few ways to transform the activity priorities into selection probabilities have been proposed, see Kolisch and Hartmann [20]. Since randomized priority rule methods generate more schedules from which the best one can be selected, they usually achieve a better solution quality than the deterministic approach.

In this paper, we consider the deterministic single-pass method as well as a new randomized multi-pass approach. Unlike many methods in the literature, the latter is not based on the computation of selection probabilities. Instead, it employs the concept of tournament selection which is an approach to select individuals from the population of a genetic algorithm (Goldberg and Deb [11], Michalewicz [24]). Generally, the idea of a tournament is as follows: k 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 k individuals has “won the tournament” and is selected. The larger the tournament size k , 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 fitness corresponds to the value derived from a given priority rule. Hence, in each step of the SGS we pick k eligible activities at random and then select the one with the best priority value.

Note, however, that the selective pressure reflected by k must be seen in relation to the number of eligible activities. A fixed value for k 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 k 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 $k = \max\{\varphi \cdot |E|, 2\}$ (values for k are rounded).

The tournament-based priority rule heuristic is defined by the number of passes (i.e., number of schedules to be generated), the priority rule to be applied, and the tournament factor φ . Note that the number of passes could also be replaced by other stopping criteria such as a time limit or a maximum number of consecutive passes without improvement.

4 Generation of Test Instances

4.1 Instance Generation for the RCPSP/t

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 [14]. ProGen of Kolisch et al. [22] 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 [21].

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 $r_{jk1} \dots r_{jkp_j}$. 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 $R_{k1} \dots R_{kT}$.

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 $t = 1, \dots, T$ of the planning horizon, the availability R_{kt} of resource $k = 1, \dots, K$ 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 $j = 1, \dots, J$ and each period $t = 1, \dots, p_j$, the request r_{jkt} for resource $k = 1, \dots, K$ 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 throughout this paper.

4.2 Test Sets

For the computational experiments, 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 [21]). 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 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.

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

Table 1: Parameter settings for generation of test sets

5 Computational Results

5.1 Lower Bounds

In what follows, we summarize the outcome of the computational experiments. Since optimal solutions for the new test instances of Section 4 are not (or not yet) known, deviations of the heuristic makespan from the optimal one cannot be given. Therefore, the results of the priority rules and 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. P^R and P^r are the probabilities that the resource capacity and the request are reduced in a period, respectively. When 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 longer duration). Thus, the deviation of LP/t from LB increases. The factors F^R and F^r play a similar role, where F^R is more important here. A factor of $F^R = 0$ implies a reduction to an availability of 0 resource units. This leads to more infeasible start times than for $F^R = 0.5$ and thus to a higher deviation of LP/t from 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 2: Deviation of new lower bound LB/t from classical lower bound LB

5.2 Priority Rules

Each of the priority rules of Section 3.2 was tested within a deterministic single-pass method on each of the data sets described in Section 4.2. The results are given in terms of average deviation

from the extended lower bound LB/t. Recall that for some instances a feasible solution might not be found, therefore the average was calculated only based on those instances for which a feasible solution was found by the respective rule. Tables 3 and 4 provide the results for the set with $J = 30$ and the set with $J = 120$, respectively. For both project sizes, the results are given for each of the six subsets separately as well as for the overall set.

$P^R = P^r$	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
RND	24.9%	21.3%	13.7%	31.9%	33.4%	34.9%	26.9%
SPT	31.3%	26.0%	14.6%	37.7%	41.7%	40.7%	32.4%
LPT	23.8%	18.1%	11.4%	29.6%	30.6%	30.8%	24.3%
LPT/t	23.9%	18.8%	11.1%	29.6%	30.6%	30.8%	24.4%
MSLK	21.3%	16.1%	11.0%	25.7%	27.8%	27.2%	21.7%
MSLK/t	20.7%	16.1%	10.6%	24.8%	26.7%	27.0%	21.2%
LFT	18.4%	14.2%	8.6%	21.7%	24.2%	23.3%	18.5%
LFT/t	18.3%	13.4%	9.6%	21.4%	23.7%	23.4%	18.4%
LST	16.4%	12.8%	8.4%	20.4%	22.3%	21.3%	17.1%
LST/t	16.8%	12.9%	9.1%	20.0%	21.6%	21.6%	17.1%

Table 3: Average deviation from lower bound LB/t — deterministic, $J = 30$

$P^R = P^r$	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
RND	57.0%	47.8%	33.8%	72.7%	75.0%	80.2%	61.1%
SPT	67.6%	60.2%	44.1%	85.1%	90.9%	98.9%	74.5%
LPT	54.6%	45.4%	32.5%	70.1%	73.2%	77.6%	58.9%
LPT/t	54.2%	45.6%	32.2%	70.1%	73.2%	77.6%	58.8%
MSLK	49.2%	41.9%	29.9%	64.6%	68.2%	74.4%	54.7%
MSLK/t	48.5%	39.5%	25.9%	64.3%	67.1%	72.7%	53.0%
LFT	37.5%	31.1%	22.1%	51.3%	54.0%	58.5%	42.4%
LFT/t	37.6%	31.0%	21.3%	51.4%	53.7%	57.4%	42.1%
LST	35.2%	28.4%	19.8%	48.9%	50.7%	54.7%	39.6%
LST/t	35.5%	29.1%	19.7%	49.1%	50.3%	53.8%	39.6%

Table 4: Average deviation from lower bound LB/t — deterministic, $J = 120$

The results show that the choice of the priority rule has a substantial impact on the solution quality. A random choice of the activity to be scheduled (i.e. the RND rule) gives bad results. That approach is used as a benchmark here, but the tables indicate that the shortest processing time rule (SPT) performs even worse. As expected, it is counterproductive to prefer short activities because they may fill gaps in the resource profile, which might make it more difficult to find early start times for longer activities. The longest processing time (LPT) rule makes more sense, but the results are only marginally better than those of the RND benchmark. Also the minimum slack (MSLK) rule is rather disappointing. The latest finish and latest start time rules (LFT and LST) perform best, and LST is a bit better than LFT. The four rule adaptations to the RCPSP/t yield only slightly better results than their standard RCPSP counterparts, but there are no noticeable

differences between LST/t and LST.

Next, Table 5 shows the percentage of instances for which a feasible solution was found for the subsets of the data set with $J = 30$. Not surprisingly, both more frequent ($P^R = 0.2$) and more drastic ($F^R = 0$) reductions of the resource availabilities make it harder for the simple single-pass priority rule methods to find feasible schedules within the horizon. Generally, rules that achieve a lower deviation from the lower bound are also more successful in finding feasible solutions, and LST/t performs best (although the differences between the rules are not big). Note that we do not consider the data set with $J = 120$ here because for that set almost all rules find a feasible solution for each of the instances.

$P^R = P^r$	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
RND	100.0%	99.0%	86.5%	100.0%	99.0%	94.4%	96.5%
SPT	100.0%	97.5%	81.5%	100.0%	98.5%	92.3%	95.0%
LPT	100.0%	99.4%	86.9%	100.0%	99.6%	95.4%	96.9%
LPT/t	100.0%	99.4%	86.3%	100.0%	99.6%	95.4%	96.8%
MSLK	100.0%	99.4%	85.8%	100.0%	99.6%	94.6%	96.6%
MSLK/t	100.0%	99.8%	87.7%	100.0%	99.4%	95.8%	97.1%
LFT	100.0%	99.6%	88.8%	100.0%	99.8%	95.2%	97.2%
LFT/t	100.0%	99.8%	89.0%	100.0%	100.0%	96.3%	97.5%
LST	100.0%	99.8%	89.4%	100.0%	100.0%	95.8%	97.5%
LST/t	100.0%	99.6%	89.8%	100.0%	99.8%	96.5%	97.6%

Table 5: Feasible solution found — deterministic, $J = 30$

5.3 Tournament Heuristic

Next, we examine the behavior of the tournament heuristic. To keep the analysis short, we consider only the LST/t priority rule. LST/t was, together with LST, the best rule in our single pass experiments. In contrast to LST, LST/t takes also the resources into account. This might be beneficial if no precedence relations are given (which was the case in the case study of Section 2.1). Therefore, we preferred LST/t over LST. As a benchmark, we used again the RND rule.

Table 6 reports the average percentage deviation from the lower bound LB/t. Since the impact of the parameters for the generation of the instances has been examined above (and since the observations also hold for the multi-pass tournament method), we do not look at the subsets and give only aggregated results for the two instance sets with $J = 30$ and $J = 120$, respectively. Two stopping criteria were used: The heuristic was stopped after 100 and 1000 schedules were calculated for an instance. The tournament factor ϕ was varied between 0.1 and 0.9. Note that the tournament factor is irrelevant for the pure random (i.e., RND) method.

Table 6 shows that, as expected, calculating more schedules per instance leads to better average results. Also, the LST/t priority rule clearly outperforms the pure random method. The results are best for medium values of the tournament factor; generally, $\phi = 0.3$ gives good results. This is no surprise as a small tournament factor decreases the probability that an eligible job with a good priority value is selected. On the other hand, a large tournament factor implies that the eligible job with the highest priority value is selected very often, which might restrict the method to a too narrow area of the search space. Generally, the impact of ϕ seems to be rather small as long as no

	passes (schedules)	RND	LST/t				
			$\varphi = 0.1$	$\varphi = 0.3$	$\varphi = 0.5$	$\varphi = 0.7$	$\varphi = 0.9$
$J = 30$	100	13.9%	12.8%	12.7%	12.7%	12.8%	13.0%
	1000	12.5%	11.8%	11.7%	11.8%	11.9%	12.2%
$J = 120$	100	44.3%	37.9%	35.5%	35.7%	35.9%	36.1%
	1000	40.8%	35.5%	33.7%	34.0%	34.3%	34.6%

Table 6: Average deviation from lower bound LB/t — tournament heuristic

	random	$\varphi = 0.1$	$\varphi = 0.3$	$\varphi = 0.5$	$\varphi = 0.7$	$\varphi = 0.9$
$J = 30$	0.020	0.020	0.020	0.021	0.022	0.023
$J = 120$	0.076	0.082	0.101	0.116	0.128	0.141

Table 7: Average computation time (sec) — tournament heuristic, 1000 schedules

extreme settings are used, which indicates that the tournament method is robust in this regard.

For the instance set with $J = 120$, already a stopping criterion of 100 schedules is sufficient to find a feasible solution for each instance. For the $J = 30$ set, however, feasible solutions are found for only 98.2% of the instances after 100 schedules and for 98.3% after 1000 schedules. Recall that it is not yet known for how many of the instances with $J = 30$ a feasible solution exists.

Finally, we take a brief look at the computation times. Table 7 shows the average CPU-seconds per instance when the heuristic stops after 1000 schedules. The calculation times are very moderate. They slightly increase when the tournament factor goes up, which is plausible since more activities have to be evaluated in each iteration if the tournament gets larger. The computer used for the experiments contains an Intel Core2 Duo CPU with 2.66 GHz and runs under Windows Vista (32 Bit).

5.4 Results for Original Data of the Real Project

The priority rule methods were also applied to the original data of the real-world medical research project of Section 2.1. This project consists of 25 experiments which were transformed into 62 activities (excluding start and end activity) and 25 resources reflecting the finish time constraints. Two further resources cover the laboratory equipment and the researcher.

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. When applied in a deterministic single-pass framework, the priority rules yielded a project makespan between 81 days (SPT rule) and 68 days (LST/t rule). The tournament heuristic (with $\varphi = 0.3$) gave the following results (since it is a non-deterministic method, several test runs were carried out for each schedule limit): After 100 passes, most test runs resulted in a makespan of 68 days. After 1000 passes, about half of the test runs led to 68 days and the other half to 67 days. After 10,000 passes, a makespan of 67 days was usually achieved. Since 67 is an instance-specific lower bound (cf. Hartmann [13]), 67 is also the optimum makespan. The calculation times were also very moderate (0.15 CPU seconds for 1000 passes). Obviously, it was not a difficult task for this simple method to find the optimal makespan—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 methods.

6 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. It has also been shown that new types of temporal constraints which can be important in such projects can be captured by the RCPSP/t. Subsequently, several concepts for the standard RCPSP have been analyzed with regard to applicability to the RCPSP/t and, where appropriate, extended. This included schedule generation schemes and priority rules.

Also 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 [19]), priority rule heuristics usually cannot compete with other approaches such as metaheuristics. Hence the application of this new method will most likely be the calculation of initial solutions of good quality 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 elaborate heuristic methods are promising areas of future research on this problem setting.

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

- [1] R. Alvarez-Valdes and J. M. Tamarit. Heuristic algorithms for resource-constrained project scheduling: A review and an empirical analysis. In R. Słowiński and J. Węglarz, editors, *Advances in project scheduling*, pages 113–134. Elsevier, Amsterdam, the Netherlands, 1989.
- [2] M. Bartusch, R. H. Möhring, and F. J. Radermacher. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research*, 16:201–240, 1988.
- [3] P. Brucker, A. Drexl, R. Möhring, K. Neumann, and E. Pesch. Resource-constrained project scheduling: Notation, classification, models, and methods. *European Journal of Operational Research*, 112: 3–41, 1999.
- [4] C. C. B. Cavalcante, C. C. de Souza, M. W. P. Savelsbergh, Y. Wang, and L. A. Wolsey. Scheduling projects with labor constraints. *Discrete Applied Mathematics*, 112(1-3):27–52, 2001.
- [5] E. W. Davis and J. H. Patterson. A comparison of heuristic and optimum solutions in resource-constrained project scheduling. *Management Science*, 21:944–955, 1975.
- [6] E. L. Demeulemeester and W. S. Herroelen. A branch-and-bound procedure for the multiple resource-constrained project scheduling problem. *Management Science*, 38:1803–1818, 1992.
- [7] E. L. Demeulemeester and W. S. Herroelen. Modelling setup times, process batches and transfer batches using activity network logic. *European Journal of Operational Research*, 89:355–365, 1996.

- [8] U. Dorndorf, E. Pesch, and T. Phan-Huy. A time-oriented branch-and-bound algorithm for resource-constrained project scheduling with generalised precedence constraints. *Management Science*, 46: 1365–1384, 2000.
- [9] L.-E. Drezet and J.-C. Billaut. A project scheduling problem with labour constraints and time-dependent activities requirements. *European Journal of Operational Research*, 112(1):217–225, 2008.
- [10] B. Franck, K. Neumann, and C. Schwindt. Project scheduling with calendars. *OR Spektrum*, 23: 325–334, 2001.
- [11] D. E. Goldberg and K. Deb. A comparative analysis of selection schemes used in genetic algorithms. pages 69–93. 1991.
- [12] S. Hartmann. Project scheduling with multiple modes: A genetic algorithm. *Annals of Operations Research*, 102:111–135, 2001.
- [13] S. Hartmann. *Project scheduling under limited resources: Models, methods, and applications*. Number 478 in Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, Germany, 1999.
- [14] S. Hartmann and D. Briskorn. A survey of variants and extensions of the resource-constrained project scheduling problem. *European Journal of Operational Research*, 207:1–14, 2010.
- [15] S. Hartmann and A. Drexl. Project scheduling with multiple modes: A comparison of exact algorithms. *Networks*, 32:283–297, 1998.
- [16] A. Kimms. Maximizing the net present value of a project under resource constraints using a lagrangian relaxation based heuristic with tight upper bounds. *Annals of Operations Research*, 102(1-4):221–236, 2001.
- [17] R. Klein. Project scheduling with time-varying resource constraints. *International Journal of Production Research*, 38:3937–3952, 2000.
- [18] R. Kolisch. Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation. *European Journal of Operational Research*, 90:320–333, 1996.
- [19] R. Kolisch and S. Hartmann. Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*, 174:23–37, 2006.
- [20] R. Kolisch and S. Hartmann. Heuristic algorithms for solving the resource-constrained project scheduling problem: Classification and computational analysis. In J. Węglarz, editor, *Project scheduling: Recent models, algorithms and applications*, pages 147–178. Kluwer Academic Publishers, 1999.
- [21] R. Kolisch and A. Sprecher. PSPLIB – a project scheduling problem library. *European Journal of Operational Research*, 96:205–216, 1996.
- [22] R. Kolisch, A. Sprecher, and A. Drexl. Characterization and generation of a general class of resource-constrained project scheduling problems. *Management Science*, 41:1693–1703, 1995.
- [23] C. Löser, T. Fitting, and U. R. Fölsch. Importance of intracellular s-adenosylmethionine decarboxylase activity for the regulation of camostate-induced pancreatic polyamine metabolism and growth: In vivo effect of two novel s-adenosylmethionine decarboxylase inhibitors. *Digestion*, 58:258–265, 1997.
- [24] Z. Michalewicz. Heuristic methods for evolutionary computation techniques. *Journal of Heuristics*, 1:177–206, 1995.
- [25] A. Mingozzi, V. Maniezzo, S. Ricciardelli, and L. Bianco. An exact algorithm for the resource-constrained project scheduling problem based on a new mathematical formulation. *Management Science*, 44:714–729, 1998.

- [26] R. H. Möhring, A. S. Schulz, F. Stork, and M. Uetz. Solving project scheduling problems by minimum cut computations. *Management Science*, 49:330–350, 2003.
- [27] K. Neumann and J. Zimmermann. Procedures for resource leveling and net present value problems in project scheduling with general temporal and resource constraints. *European Journal of Operational Research*, 127(2):425–443, 2000.
- [28] K. Neumann, C. Schwindt, and J. Zimmermann. Recent results on resource-constrained project scheduling with time windows: Models, solution methods, and applications. *Central European Journal of Operations Research*, 10:113–148, 2002.
- [29] A. A. B. Pritsker, L. J. Watters, and P. M. Wolfe. Multiproject scheduling with limited resources: A zero-one programming approach. *Management Science*, 16:93–107, 1969.
- [30] A. Sprecher. Scheduling resource-constrained projects competitively at modest memory requirements. *Management Science*, 46:710–723, 2000.
- [31] A. Sprecher. *Resource-constrained project scheduling: Exact methods for the multi-mode case*. Number 409 in Lecture Notes in Economics and Mathematical Systems. Springer, Berlin, Germany, 1994.
- [32] A. Sprecher and A. Drexl. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. *European Journal of Operational Research*, 107:431–450, 1998.
- [33] A. Sprecher, R. Kolisch, and A. Drexl. Semi-active, active and non-delay schedules for the resource-constrained project scheduling problem. *European Journal of Operational Research*, 80:94–102, 1995.
- [34] F. B. Talbot. Resource-constrained project scheduling with time-resource tradeoffs: The nonpreemptive case. *Management Science*, 28:1197–1210, 1982.