

Karnebogen, Mareike; Zimmermann, Jürgen

**Article — Published Version**

## Generation schemes for the resource-constrained project scheduling problem with partially renewable resources and generalized precedence constraints

Annals of Operations Research

**Provided in Cooperation with:**

Springer Nature

*Suggested Citation:* Karnebogen, Mareike; Zimmermann, Jürgen (2024) : Generation schemes for the resource-constrained project scheduling problem with partially renewable resources and generalized precedence constraints, Annals of Operations Research, ISSN 1572-9338, Springer US, New York, NY, Vol. 338, Iss. 1, pp. 173-192,  
<https://doi.org/10.1007/s10479-023-05788-3>

This Version is available at:

<https://hdl.handle.net/10419/315292>

**Standard-Nutzungsbedingungen:**

Die Dokumente auf EconStor dürfen zu eigenen wissenschaftlichen Zwecken und zum Privatgebrauch gespeichert und kopiert werden.

Sie dürfen die Dokumente nicht für öffentliche oder kommerzielle Zwecke vervielfältigen, öffentlich ausstellen, öffentlich zugänglich machen, vertreiben oder anderweitig nutzen.

Sofern die Verfasser die Dokumente unter Open-Content-Lizenzen (insbesondere CC-Lizenzen) zur Verfügung gestellt haben sollten, gelten abweichend von diesen Nutzungsbedingungen die in der dort genannten Lizenz gewährten Nutzungsrechte.

**Terms of use:**

*Documents in EconStor may be saved and copied for your personal and scholarly purposes.*

*You are not to copy documents for public or commercial purposes, to exhibit the documents publicly, to make them publicly available on the internet, or to distribute or otherwise use the documents in public.*

*If the documents have been made available under an Open Content Licence (especially Creative Commons Licences), you may exercise further usage rights as specified in the indicated licence.*



<http://creativecommons.org/licenses/by/4.0/>



# Generation schemes for the resource-constrained project scheduling problem with partially renewable resources and generalized precedence constraints

Mareike Karnebogen<sup>1</sup> · Jürgen Zimmermann<sup>1</sup>

Received: 7 December 2022 / Accepted: 11 December 2023 / Published online: 17 January 2024  
© The Author(s) 2024

## Abstract

In recent years, new resource types have been established in project scheduling. These include so-called partially renewable resources, whose total capacity applies only to a subset of periods in the planning horizon. In this paper, we consider the extension of the resource-constrained project scheduling problem with those partially renewable resources as well as generalized precedence constraints with the objective of minimizing the project duration (RCPSP/max- $\pi$ ). For this problem it is known that already the determination of a feasible solution is NP-hard in the strong sense. Hence, we present two different generation schemes that are able to find good feasible solutions in short time for most tested instances. The first one is a construction-based heuristic wherein the activities of the project are scheduled iteratively time- and resource-feasibly. The second one is a relaxation-based generation scheme, in which—starting from the schedule consisting of the earliest start times—resource conflicts are identified and resolved by inserting additional resource constraints.

**Keywords** Project scheduling · Generalized precedence constraints · RCPSP/max · Partially renewable resources · Generation schemes

## 1 Introduction

Due to new requirements in the field of project planning, such as the increasingly necessary flexibilization of production or higher expectations of employees on their working times, renewable resources are often coming up to their limits. Therefore, more and more resource types have been established in the literature in recent years. One of them are so-called partially renewable resources, which are a generalization of renewable and non-renewable resources and also include discrete cumulative resources (Watermeyer, 2021). Partially renewable

---

✉ Mareike Karnebogen  
mareike.karnebogen@tu-clausthal.de

Jürgen Zimmermann  
juergen.zimmermann@tu-clausthal.de

<sup>1</sup> Institute of Management and Economics, Clausthal University of Technology, Julius-Albert-Str. 2, D-38678 Clausthal-Zellerfeld, Germany

resources are characterized by the fact that their total capacity is limited for a subset of periods in the planning horizon. However, for the complementary set of periods, the resource is assumed to be available in sufficient quantity. Partially renewable resources thus open up the possibility of modeling new types of conditions in the field of project scheduling, e.g., special working hour models such as particular weekend arrangements or certain production constraints like planning dependent downtimes or setup times.

In this paper, the resource-constrained project scheduling problem with partially renewable resources and generalized precedence constraints (RCPSP/max- $\pi$ ) is considered, which was first addressed by Watermeyer and Zimmermann (2020). Even though the problem is based on the well studied RCPSP/max, the solution procedures cannot be simply transferred - due to the differently behaving resources and the associated changed problem characteristics. Since real-world projects often involve many activities as well as resources that have to be considered, exact solution methods for realistic instance sizes are usually not able to determine exact solutions to the problem within an acceptable amount of time. Therefore, this paper is devoted to the development of problem-specific heuristics for the RCPSP/max- $\pi$ . We present two different generation schemes that generate feasible solutions for the considered problem. In Sect. 2 the problem is described and a mixed-integer linear formulation (MILP) is introduced, followed by an overview of the relevant literature on partially renewable resources in the context of project scheduling in Sect. 3. Then, in Sect. 4 first a construction-based generation scheme and afterwards a relaxation-based generation scheme is presented. Finally, Sect. 5 provides the results of a performance analysis comparing the results of both generation schemes to the results of the presented MILP model solved by IBM ILOG CPLEX and the best performing branch-and-bound procedure of Watermeyer (2021).

## 2 Problem description

The RCPSP/max- $\pi$  is based on the well-known project scheduling problem with generalized precedence constraints (PSP/max). Thereby, a set of activities  $\mathcal{V} = \{0, 1, \dots, n+1\}$  is given, consisting of the fictitious project start 0,  $n$  real activities and the fictitious project completion  $n+1$ . Each activity  $i \in \mathcal{V}$  has a deterministic processing time  $p_i \in \mathbb{Z}_{\geq 0}$  during which the activity cannot be interrupted. For both fictitious activities,  $p_0 = p_{n+1} = 0$  applies. Between the activities of the project generalized precedence constraints have to be observed, which can be both minimal and maximal timelags. Due to deterministic activity durations, all precedence constraints can be converted into minimal start-start-timelags. A time constraint  $\delta_{ij} \geq 0$  indicates that more than  $\delta_{ij}$  periods must have elapsed between the start of activity  $i \in \mathcal{V}$  and the start of activity  $j \in \mathcal{V} \setminus \{i\}$ . A time constraint  $\delta_{ij} < 0$  states that the start of activity  $i \in \mathcal{V}$  must not be more than  $-\delta_{ij}$  periods after the start of activity  $j \in \mathcal{V} \setminus \{i\}$ . Furthermore, a maximal project duration  $\bar{d}$  is given in which the project must be completed modeled as a maximal timelag between project start and end.

The PSP/max as introduced before can be visualized by an activity-on-node network. The nodes  $\mathcal{V} = \{0, 1, \dots, n+1\}$  correspond to the activities of the project whereas the arcs  $\langle i, j \rangle \in E$  weighted with  $\delta_{ij} \in \mathbb{Z}$  represent the temporal constraints. The maximal project duration can be displayed by an arc from  $n+1$  to 0 weighted with  $-\bar{d}$ . As common practice, let  $d_{ij}$  be the length of a longest path from node  $i \in \mathcal{V}$  to node  $j \in \mathcal{V}$  in the project network, which can be determined using the Floyd-Warshall Algorithm by Floyd (1962). Then,  $d_{0i}$  corresponds to the earliest start time  $ES_i$  and  $-d_{i0}$  to the latest start time  $LS_i$  of activity  $i \in \mathcal{V}$ . Based on that, for each activity  $i \in \mathcal{V}$  a set  $\bar{W}_i = \{ES_i, ES_i + 1, \dots, LS_i\}$  containing all time-feasible integer start times between its earliest and latest start time can be initialized.

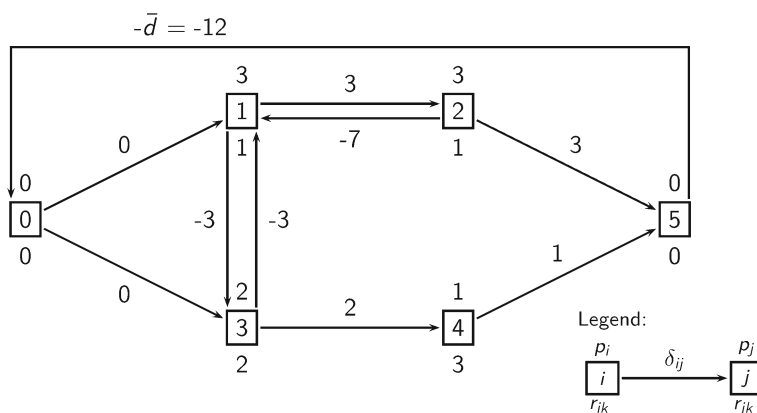


Fig. 1 Project network

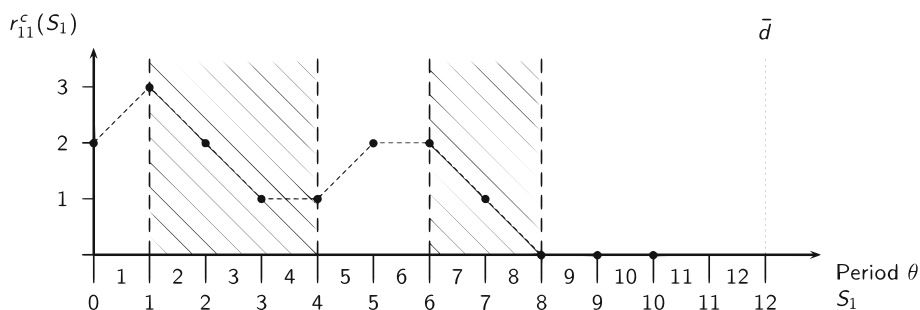


Fig. 2 Relevant cumulative resource consumption of activity 1

The PSP/max can be extended to the RCPSP/max- $\pi$  by additionally considering a set of partially renewable resources  $\mathcal{R} = \{1, \dots, m\}$ . While an activity  $i \in \mathcal{V}$  is in execution, it consumes  $r_{ik} \in \mathbb{Z}_{\geq 0}$  units of resource  $k \in \mathcal{R}$  per period. The resource requirements of the fictitious activities are assumed to be zero. For each partially renewable resource  $k \in \mathcal{R}$  a resource capacity  $R_k$  is given, which is only related to a subset  $\Pi_k \subseteq \{1, 2, \dots, \bar{d}\}$  of not necessarily connected time periods of the planning horizon. All activities executed in this restricted periods of resource  $k$  are not allowed to in total consume more than  $R_k$  units within. However, for all periods not contained in  $\Pi_k$ , it is assumed that the resource availability is unlimited. In the activity-on-node network for each resource  $k \in \mathcal{R}$  the resource demand per period  $r_{ik}$  of an activity  $i \in \mathcal{V}$  is noted below the corresponding node. An exemplary network for a project with  $n = 4$  real activities and  $m = 1$  partially renewable resource is shown in Fig. 1.

According to Watermeyer and Zimmermann (2020) the relevant cumulative resource consumption of an activity  $i \in \mathcal{V}$  of a resource  $k \in \mathcal{R}$  is based on its start time  $S_i$  and can be calculated as  $r_{ik}^c(S_i) = |\{S_i + 1, S_i + 2, \dots, S_i + p_i\} \cap \Pi_k| \cdot r_{ik}$ . Assuming one resource  $k = 1$  with  $\Pi_1 = \{2, 3, 4, 7, 8\}$  for the project visualised in Fig. 1, for activity  $i = 1$  the function for the cumulative resource consumption as shown in Fig. 2 results.

The objective of the problem is to find a time- and resource-feasible schedule  $S = (S_0, S_1, \dots, S_{n+1})$  minimizing the project duration  $S_{n+1}$ . A schedule is called time-feasible, if  $S_j - S_i \geq \delta_{ij}$  applies to all  $(i, j) \in E$ , meaning that all temporal constraints are observed.

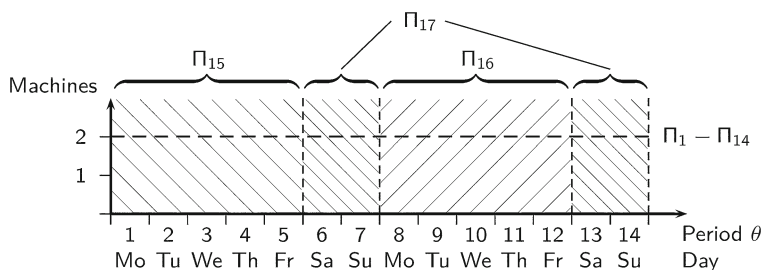
A schedule is resource-feasible, if the accumulated resource consumption within all capacitated periods not exceeds the given capacity  $R_k$  for any partially renewable resource  $k \in \mathcal{R}$ . To ensure this, for each resource  $k \in \mathcal{R}$  the start time dependent cumulative resource consumption  $r_{ik}^c(S_i)$  is summed up over all activities  $i \in \mathcal{V}$  and limited by  $R_k$ . Formally, the RCPSP/max- $\pi$  described above can be formulated as follows:

$$\begin{aligned} & \text{Minimize} && f(S) = S_{n+1} \\ & \text{subject to} && S_j - S_i \geq \delta_{ij} && ((i, j) \in E) \\ & && S_0 = 0 \\ & && \sum_{i \in \mathcal{V}} r_{ik}^c(S_i) \leq R_k && (k \in \mathcal{R}) \\ & && S_i \in \mathbb{Z}_{\geq 0} && (i \in \mathcal{V}) \end{aligned}$$

As is common practice in resource-constrained project scheduling, the RCPSP/max- $\pi$  can also be formulated with binary time-indexed variables  $x_{it}$  for each activity  $i \in \mathcal{V}$  and each start time point  $t \in \overline{W}_i$ . If activity  $i$  starts at  $t$  ( $S_i = t$ ) or at an earlier point in time, the corresponding binary variable  $x_{it}$  takes the value 1, otherwise  $x_{it}$  has the value 0. This type of binary decision variables are also called step variables, because the variables  $x_{it}$  jump from value 0 to value 1, i.e. one step up, exactly once over time for each activity  $i \in \mathcal{V}$ . Hence, the decision variable  $S_i$  can be replaced by the term  $\sum_{t \in \overline{W}_i \setminus \{0\}} (x_{it} - x_{i,t-1})$ . If we define sets  $Q_{it} := \{t - p_i + 1, \dots, t\}$  containing all points in time an activity  $i \in \mathcal{V}$  could start so that it would be in execution at point in time  $t$ , the cumulative resource consumption can be expressed dependent on the binary decision variables  $x_{it}$ . For this, for each time-feasible start time  $t \in \overline{W}_i$  of an activity  $i \in \mathcal{V}$ , the number of capacitated periods in which  $i$  would be active if started at  $t$  is determined. The discrete-time formulation based on step start variables can then be given according to Watermeyer and Zimmermann (2020) as follows:

$$\left. \begin{aligned} & \text{Min.} && \sum_{t \in \overline{W}_{n+1}} t \cdot (x_{n+1,t} - x_{n+1,t-1}) \\ & \text{s.t.} && \sum_{t \in \overline{W}_j \setminus \{0\}} t \cdot (x_{jt} - x_{j,t-1}) \\ & && - \sum_{t \in \overline{W}_i \setminus \{0\}} t \cdot (x_{it} - x_{i,t-1}) \geq \delta_{ij} && ((i, j) \in E) \\ & && \sum_{i \in \mathcal{V}} r_{ik} \sum_{\tau \in \Pi_k} \sum_{t \in Q_{i,(\tau-1)} \cap \overline{W}_i} (x_{it} - x_{i,t-p_i}) \leq R_k && (k \in \mathcal{R}) \\ & && x_{i,t-1} \leq x_{it} && (i \in \mathcal{V}, t \in \overline{W}_i \setminus \{0\}) \\ & && x_{it} = 0 && (i \in \mathcal{V}, t \in \{0, \dots, ES_i - 1\}) \\ & && x_{it} = 1 && (i \in \mathcal{V}, t \in \{LS_i, \dots, \overline{d}\}) \\ & && x_{it} \in \{0, 1\} && (i \in \mathcal{V}, t \in \overline{W}_i) \end{aligned} \right\} \begin{array}{l} \text{(RCPSP/} \\ \text{max-}\pi) \end{array}$$

In order to motivate the relevance of partially renewable resources, an example is given where partially renewable resources are utilized to model all existing resource constraints. Suppose a company plans a two-week project on a daily basis. For this, two identical machines are available. Normally, the daily capacity of the machines is modeled by renewable resources, but partially renewable resources can be used instead. Therefore, for each period of the planning horizon a new partially renewable resource with a capacity of two is defined, i.e.



**Fig. 3** Example for partially renewable resources

$\Pi_k = \{k\}$  and  $R_k = 2$  for  $k \in \{1, \dots, 14\}$ . In addition, each 5-day workweek one of the two machines has to be cleaned and maintained, leading to a downtime of a whole day. A restriction like this can be modeled with renewable resources and dummy activities or - quite easy - with partially renewable resources. For that, a resource  $k = 15$  is defined restricted to the periods 1 to 5 ( $\Pi_{15} = \{1, \dots, 5\}$ ) to a capacity of  $2 \cdot 5 - 1 = 9$  ( $R_{15} = 9$ ). Similarly, for the second week a resource  $k = 16$  with  $\Pi_{16} = \{8, \dots, 12\}$  and  $R_{16} = 9$  is introduced. As a further limitation, each machine may not run for more than two days on both weekends due to higher operating costs on Saturday and Sunday. To map this restriction, another partially renewable resource  $k = 17$  is defined with  $\Pi_{17} = \{6, 7, 13, 14\}$  and  $R_{17} = 4$ .

The resource restrictions as introduced before are shown in a resource profile in Fig. 3. The dashed line represents the 14 partial renewable resources limiting the machine capacity per day. The maintenance restrictions are depicted by broad striped areas, while the weekend restriction is displayed by narrow striped areas.

Resource constraints such as in the given example show the relevance of the concept of partially renewable resources, because they enable to model new restrictions as flexible working time models or special weekend arrangements.

### 3 Literature review

Partially renewable resources embedded in the context of project scheduling were first mentioned by Drexel et al. (1993) as well as Drexel and Salewski (1997) for modeling an academic course scheduling problem. Over the next few years, several papers were published dealing with the problem of RCPSP/ $\pi$  and problem specific solution methods. Böttcher et al. (1999) adapt a branch-and-bound procedure of Talbot and Patterson (1978), so that the RCPSP/ $\pi$  can be solved to optimality. In addition, they developed a serial scheduling scheme to construct feasible solutions for the RCPSP/ $\pi$ . Schirmer (1999) also present construction methods for project scheduling with partially-renewable resources. Furthermore, he developed local search procedures, e.g. a tabu search heuristic. Alvarez-Valdés et al. (2006) present a scatter search heuristic for the RCPSP/ $\pi$ . Two years later, they also introduced a GRASP (Alvarez-Valdés et al., 2008). A summary and overview of the literature can be found in Alvarez-Valdés et al. (2015).

The resource-constrained project scheduling problem with partially renewable resources and generalized precedence constraints was first considered by Watermeyer and Zimmermann (2020). They present a branch-and-bound procedure which is based on the resource relaxation. As part of their further research they also developed a partition-based branch-and-bound procedure (Watermeyer & Zimmermann, 2022) as well as a constructive branch-and-bound

procedure (Watermeyer & Zimmermann, 2023). To the best knowledge of the authors, apart from the schedule-generation scheme of Watermeyer and Zimmermann (2023) there exist no appropriate heuristic solution methods for the RCPSP/max- $\pi$  so far.

## 4 Generation schemes

Due to the lack of heuristic solution methods for the RCPSP/max- $\pi$ , in the following we present two different generation schemes to get feasible solutions. The first one is a construction-based heuristic presented at the 17th International Conference on Project Management and Scheduling (Karnebogen & Zimmermann, 2021) and will be described in Sect. 4.1. The other one is a relaxation-based heuristic first introduced at the 18th International Conference on Project Management and Scheduling (Karnebogen & Zimmermann, 2022) that will be explained in Sect. 4.2.

### 4.1 Construction-based generation scheme

The construction-based generation scheme is characterized by iteratively scheduling the activities of a project until a time- and resource-feasible schedule is built. The general concept of the heuristic is based on the well-known generation scheme for the RCPSP/max developed by Franck et al. (2001). However, due to the partially renewable resources, it is - in contrast to the RCPSP/max - not sufficient to restrict the construction to the set of active schedules, i.e. generally select the earliest time- and resource-feasible start time in the course of the construction phase. Accordingly, the unscheduling phase must also be changed, since a rightward shift as in Franck et al. (2001) is not appropriate. Moreover, the reasons for the necessity of an unscheduling step are more diverse. In addition to binding time restrictions that prevent an activity from being scheduled, one or more resources may also no longer be available in sufficient quantity due to previous scheduling steps.

Algorithm 1 shows the procedure of our construction-based generation scheme. In the beginning, an initialization process is done (line 1–7), in which the fictitious project start 0 is scheduled at point in time zero ( $S_0 = 0$ ) and included in the set  $\mathcal{C}$  of activities that have already been scheduled. Furthermore, a counter  $u = 0$  is initialized to count the number of unscheduling steps done in the recurring main step of the generation scheme as well as an empty tabu-list  $\Theta_i$  for each activity  $i \in \mathcal{V}$  to manage tabu-set start times. Since the total resource requirement  $r_{ik}^c(S_i)$  of an activity  $i \in \mathcal{V} \setminus \{0\}$  in the restricted periods of a partially renewable resource  $k \in \mathcal{R}$  only depends on its own start time  $S_i$  and not on those of other activities, it can be calculated in advance for all possible start times  $t \in \overline{W}_i = \{ES_i, ES_i + 1, \dots, LS_i\}$  between its earliest and latest start time.

In the main step of the generation scheme, which is iterated until all activities are feasibly scheduled, for each partially renewable resource  $k \in \mathcal{R}$  and each activity  $i \in \mathcal{V} \setminus \{\mathcal{C}\}$  the minimal cumulative demand  $r_{ik}^{min}$  and the maximal cumulative demand  $r_{ik}^{max}$  are determined. Since an activity  $i \in \mathcal{V}$  cannot have less than its minimal cumulative resource demand  $r_{ik}^{min}$  for each resource  $k \in \mathcal{R}$  in any feasible schedule, this demand is already deducted from the total capacity  $RC_k$  as well as the demand of already scheduled activities. In the next step, the eligible set  $\mathcal{E}$  is established containing all activities  $i \in \mathcal{V} \setminus \mathcal{C}$  whose immediate predecessors regarding the distance order  $<_D$  are already scheduled. An activity  $h \in \mathcal{V} \setminus \{i\}$  is a predecessor of activity  $i$  regarding the distance order  $<_D$ , if either  $d_{hi} > 0$  or  $d_{hi} = 0$  and  $d_{ih} < 0$  applies (Neumann et al., 2003). Then, based on a predefined priority rule, for all

eligible activities  $i \in \mathcal{E}$  priority values are calculated and the activity  $j^* \in \mathcal{E}$  with the highest priority is selected. For activity  $j^*$  a set  $\mathcal{Z}_{j^*}$  containing all points in time  $t \in [ES_{j^*}, LS_{j^*}]$ , which are resource-feasible and dominant, is determined. A point in time  $t \in [ES_i, LS_i]$  is called dominant for an activity  $i \in \mathcal{V}$ , if the resource demand  $r_{ik}^c(t)$  is smaller for at least one resource  $k \in \mathcal{R}$  compared to all points in time  $\tau \in \mathcal{Z}_{j^*} \mid \tau < t$ . The restriction to dominant points in time is motivated by the fact that, due to the aim of minimizing the project duration, an activity should generally be scheduled as early as possible and therefore a later start time is only accepted if it leads to a saving for at least one resource compared to all previous dominant as well as time- and resource-feasible points in time. This is not a limitation for the generation scheme, because dominated points in time can become dominant due to the unscheduling step in the further course of the heuristic, and thus are not necessarily unselectable. If  $\mathcal{Z}_{j^*}$  is empty, i.e. no time- and resource-feasible start time exists for activity  $j^*$ , an unscheduling step is performed and the counter  $u$  is increased by one. Otherwise, based on priority values calculated according to a given priority rule, the earliest point in time  $t^* \in \mathcal{Z}_{j^*}$  with highest priority is selected and set as start time of  $j^*$  ( $S_{j^*} = t^*$ ). As this causes the activity to be scheduled,  $j^*$  is added to  $\mathcal{C}$ . If all activities are scheduled, the procedure terminates. Otherwise, since the scheduling of the activity may have reduced the time windows of other activities, the earliest and latest start times of all activities not scheduled so far are updated and the procedure is repeated.

Algorithm 2 shows the unscheduling step which is performed if no time- and resource-feasible point in time for the start of activity  $j^*$  can be found. In case  $u$  is higher than a maximal number of unscheduling steps  $\hat{u}$  the algorithm aborts and no feasible schedule can

---

### Algorithm 1 Construction-based generation scheme

---

**Input:** RCPSP/max- $\pi$  instance

- 1: Determine longest path lengths  $d_{ij}$  for all  $i, j \in \mathcal{V}$
- 2: Determine  $Pred^{<D}(i)$  for all  $i \in \mathcal{V}$
- 3: Set  $ES_i = d_{0i}$ ,  $LS_i = -d_{i0}$  for all  $i \in \mathcal{V}$
- 4: Set  $\mathcal{C} := \{0\}$ ,  $S_0 := 0$ ,  $u := 0$ ,  $\Theta_i := \emptyset$  for all  $i \in \mathcal{V}$
- 5: Determine  $r_{ik}^c(t)$  for all  $i \in \mathcal{V} \setminus \{0\}$  and  $k \in \mathcal{R}$  and  $t \in \overline{W}_i$
- 6: **for** all  $i \in \mathcal{V} \setminus \{0\}$  **do**
- 7:   **if**  $ES_i = LS_i$  **then**  $S_i = ES_i$ ,  $\mathcal{C} := \mathcal{C} \cup \{i\}$
- 8: **while**  $\mathcal{C} \neq \mathcal{V}$  **do**
- 9:    $r_{ik}^{min} := \min_{t \in \overline{W}_i} r_{ik}^c(t)$  for all  $i \in \mathcal{V} \setminus \mathcal{C}$  and  $k \in \mathcal{R}$
- 10:    $r_{ik}^{max} := \max_{t \in \overline{W}_i} r_{ik}^c(t)$  for all  $i \in \mathcal{V} \setminus \mathcal{C}$  and  $k \in \mathcal{R}$
- 11:    $RC_k := R_k - \sum_{i \in \mathcal{V} \setminus \mathcal{C}} r_{ik}^{min} - \sum_{i \in \mathcal{C}} r_{ik}^c(S_i)$  for all  $k \in \mathcal{R}$
- 12:    $\mathcal{E} := \{i \in \mathcal{V} \setminus \mathcal{C} \mid Pred^{<D}(i) \subseteq \mathcal{C}\}$
- 13:   priority based choice of activity  $j^* \in \mathcal{E}$  to be scheduled next
- 14:    $\mathcal{Z}_{j^*} := \{t \in \overline{W}_{j^*} \setminus \Theta_{j^*} \mid r_{j^*k}^c(t) - r_{j^*k}^{min} \leq RC_k \text{ for all } k \in \mathcal{R} \text{ and } \min_{k \in \mathcal{R}} \{r_{j^*k}^c(t) - r_{j^*k}^{min}\} < 0 \text{ for all } \tau \in \overline{W}_{j^*} \setminus \Theta_{j^*} \mid \tau < t\}$
- 15:   **if**  $\mathcal{Z}_{j^*} = \emptyset$  **then**  $u := u + 1$  and **Unschedule**
- 16:   **else**
- 17:     priority based choice of point in time  $t^* \in \mathcal{Z}_{j^*}$  as start time of  $j^*$
- 18:      $S_{j^*} := t^*$ ,  $\mathcal{C} := \mathcal{C} \cup \{j^*\}$
- 19:     **for** all  $h \in \mathcal{V} \setminus \mathcal{C}$  **do**
- 20:        $ES_h := \max(ES_h, S_{j^*} + d_{jh})$
- 21:        $LS_h := \min(LS_h, S_{j^*} - d_{hj})$
- 22:        $\overline{W}_h := \{ES_h, \dots, LS_h\}$
- 23:       **if**  $ES_h = LS_h$  **then**  $S_h = ES_h$ ,  $\mathcal{C} := \mathcal{C} \cup \{h\}$
- 24: **return**  $S$

---



**Algorithm 2** Unschedule

---

**Input:**  $\mathcal{C}$ ,  $S_i$  and  $\Theta_i$  for all  $i \in \mathcal{V}$ ,  $j^*$ ,  $r_{ik}^c(S_i)$  for all  $i \in \mathcal{V}$  and  $k \in \mathcal{R}$

- 1: **if**  $u \geq \hat{u}$  **then** terminate
- 2: **if**  $ES_{j^*} \neq d_{0j^*}$  **then**  $\mathcal{U} := \{i \in \mathcal{C} \mid ES_{j^*} = S_i + d_{ij^*}\}$
- 3: **if**  $LS_{j^*} \neq -d_{j^*0}$  **then**  $\mathcal{U} := \mathcal{U} \cup \{i \in \mathcal{C} \mid LS_{j^*} = S_i - d_{j^*i}\}$
- 4: **if**  $\mathcal{U} := \emptyset$  **then**  $\mathcal{U} := \{i \in \mathcal{C} \mid \min\{r_{ik}(S_i), r_{j^*k}\} > 0 \text{ for at least one } k \in \mathcal{R}\}$
- 5: **for all**  $i \in \mathcal{U}$  **do**
- 6:    $\mathcal{C} := \mathcal{C} \setminus \{i\}$
- 7:    $\Theta_i = \Theta_i \cup \{S_i\}$
- 8:    $\Theta_{j^*} := \emptyset$
- 9: **for all**  $i \in \mathcal{C}$  with  $S_i > \min_{h \in \mathcal{U}} S_h$  **do**
- 10:    $\mathcal{C} := \mathcal{C} \setminus \{i\}$
- 11: **for all**  $h \in \mathcal{V} \setminus \mathcal{C}$  **do**
- 12:    $ES_h := d_{0h}$
- 13:    $LS_h := -d_{h0}$
- 14:   **for all**  $i \in \mathcal{C}$  **do**
- 15:      $ES_h := \max(ES_h, S_i + d_{ih})$
- 16:      $LS_h := \min(LS_h, S_i - d_{hi})$
- 17: **return**  $\mathcal{C}$ ,  $ES_i$  and  $LS_i$  for all  $i \in \mathcal{V} \setminus \mathcal{C}$ ,  $\Theta_i$  for all  $i \in \mathcal{V}$

---

be returned. Otherwise, all activities  $i \in \mathcal{C}$  that cause  $j^*$  to be unfeasibly scheduled are identified and stored in set  $\mathcal{U}$ . For this, we first examine if one or more activities  $i \in \mathcal{C}$  restrict the time window of the chosen activity  $j^*$  i.e. increases  $ES_{j^*}$  or decreases  $LS_{j^*}$ . If this is not the case for any of the scheduled activities, we determine all activities  $i \in \mathcal{C}$  that require one or more resource  $k \in \mathcal{R}$  that activity  $j^*$  also needs to be executed, but whose capacity is insufficient. Because at least one activity  $i \in \mathcal{U}$  has to be rescheduled in order to obtain a feasible schedule, all activities  $i \in \mathcal{U}$  are unscheduled and removed from the set  $\mathcal{C}$ . Moreover, the current start point  $S_i$  is forbidden by storing it in the tabu-list  $\Theta_i$  whereas  $\Theta_{j^*}$  is cleared. Points in time  $t \in \Theta_i$  can no longer be chosen as start time of activity  $i$  in the scheduling phase of the generation scheme until they are removed from the tabu-list. In addition, all activities  $i \in \mathcal{C}$  with  $S_i > \min_{h \in \mathcal{U}} S_h$  are also unscheduled, because they could start earlier in time due to the unscheduling of activities and potentially increasing remaining resource availabilities. Finally, for all activities  $i \in \mathcal{V} \setminus \mathcal{C}$  the earliest and latest start times  $ES_i$  and  $LS_i$  are recalculated.

In addition to the approach presented above, only scarce resources are considered to further improve the heuristic. If the remaining capacity for a resource  $k \in \mathcal{R}$  is higher than the sum of the maximal resource demands  $r_{ik}^{max}$  of all activities that remain to be scheduled, the resource is no longer critical and consequently can be removed from the set of resources  $\mathcal{R}$  to be observed. Since this can change again later in the procedure due to deallocation, resources that are used by unscheduled activities but have been removed from  $\mathcal{R}$  are checked whether a resumption is necessary. Moreover, to improve the generation scheme and to prevent it to getting stuck, we extended the activity selection to consider strong components of the network. A strong component of a network denotes a maximum set of activities for which each activity is reachable from every other activity (without using  $\langle n+1, 0 \rangle$ ) (Neumann et al., 2003). Once an activity of a strong component has been scheduled, the remaining activities of this component are prioritized during the activity selection in the following iterations.

For the activity selection, we used the priority rules LSTd ("latest start time first - dynamic") and TFd ("smallest total float first - dynamic"), since resource-based priority rules have shown worse performance in pre-tests. For the start time selection the priority

rules  $T_{min}$  ("earliest start time"), RD ("minimal total resource demand"), and RL ("resource leveling") are applied.

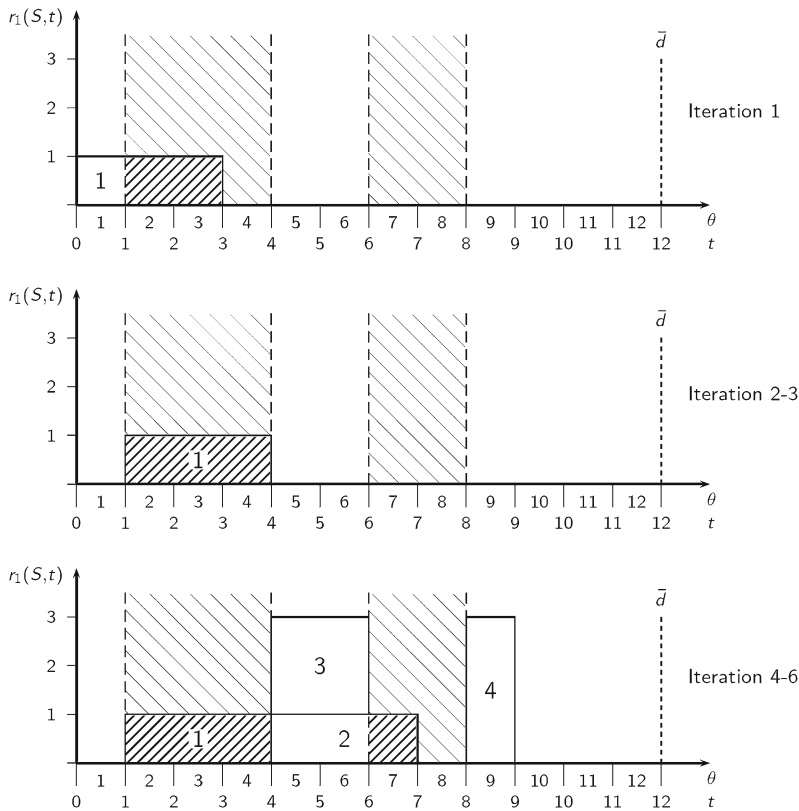
If a stochastic priority rule is used for the activity selection, all possible scheduling sequences regarding the distance order can be achieved. Furthermore, due to the tabooing and de-tabooing of possible start times, all time-feasible points in time can be selected as start time for each activity in the course of the procedure, if a stochastic priority rule is used for selecting the start time. Thus, the generation scheme is complete, i.e. there is a sequence of scheduling and unscheduling steps to determine an optimal solution for the RCPSP/max- $\pi$ . For this, the restriction to dominant points in time is not a limitation. Assume that a schedule was constructed with an arbitrary scheduling sequence in which at least one operation  $j \in \mathcal{V}$  does not start at a dominant point in time. In this case,  $j$  could be moved forward in time to a permissible point in time  $t < S_j$  that dominates the current start time  $S_j$ . Such a left shift would result in a non-increasing resource requirement of activity  $j$  for each resource  $k$ . Since the project duration is non-increasing in the start times of the activities with constant resource requirements, the resulting schedule is not worse than the original schedule with respect to the pursued objective function. It can be concluded that the set of optimal solutions must contain at least one schedule in which all operations start at a dominant point in time. Consequently, the procedure is complete, despite the restriction to the scheduling-dependent feasible and dominant points in time.

The generation scheme can be easily converted into a multistart approach by using the priority rules stochastically instead of deterministically. This can be realized, for example, by a roulette wheel selection based on selection probabilities calculated according to the chosen priority rule. To improve the multistart generation scheme, each run the maximal project duration  $\bar{d}$  is set to  $S_{n+1} - 1$  of the best known schedule and sets  $\bar{W}_i$  are updated accordingly.

The construction-based generation scheme has a time complexity of  $\mathcal{O}(\max(|\mathcal{V}|^3, |\mathcal{V}| |\mathcal{R}| \bar{d}^2))$ , if there does not occur any unscheduling step. The unscheduling step has a time complexity of  $\mathcal{O}(\max(|\mathcal{V}|^2, |\mathcal{V}| |\mathcal{R}|))$ . Assuming, we have scheduled  $n$  activities until we perform an unscheduling step and that the maximal number of unscheduling steps equals  $|\mathcal{V}|$ , we in total obtain a time complexity of  $\mathcal{O}(|\mathcal{V}|^2 \max(|\mathcal{V}|^3, |\mathcal{V}| |\mathcal{R}| \bar{d}^2))$ .

In order to illustrate the procedure of our first generation scheme, it is conducted exemplary for the project from Fig. 1 with a capacity of  $R_k = 4$  and  $\Pi_k = \{2, 3, 4, 7, 8\}$  for the partially-renewable resource  $k$ . For the sake of simplicity, no selection probabilities are calculated, instead the priority rules are applied deterministically. For the choice of activity  $j^*$  the priority rule "smallest total float first - dynamic" (TFd) is used, meaning the activity  $i \in \mathcal{E}$  with the currently smallest total float  $TF_i := LS_i - ES_i$  is scheduled next, whereas for the start time  $t^*$  the earliest feasible start time  $t \in \mathcal{Z}_{j^*}$  is chosen, which corresponds to the priority rule  $T_{min}$ .

In the course of the initialization step, the project start is scheduled at point in time zero. Furthermore, for each activity  $i \in \mathcal{V} \setminus \{0\}$  the start time dependent resource consumption is calculated for each possible start time  $t \in \bar{W}_i$  just like shown in Fig. 2 for activity 1. Afterwards, the construction phase of the generation scheme starts. In the first iteration, activity 1 and 2 can be scheduled, i.e.  $\mathcal{E} = \{1, 2\}$  applies. Because activity 1 has a smaller slack time ( $LS_1 - ES_1 = 6$ ) than activity 2 ( $LS_2 - ES_2 = 9$ ),  $j^* = 1$  is selected. Since no real activity has been scheduled so far, the tabu list of activity 1 is empty. The minimal resource requirement of the remaining activities equals one, so each point in time  $t$  between  $ES_1$  and  $LS_1$  is resource-feasible. However, within this time window only the points in time  $t = 0$  and  $t = 3$  are dominant, i.e.  $\mathcal{Z}_1 = \{0, 3\}$ . According to the given rule, we select the earliest possible start time  $S_1 = 0$ , add activity 1 to set  $C$  and perform an  $ES$ - $LS$ -update for all activities  $i \in \mathcal{V} \setminus C$ . This results in changes for the following values:  $LS_2 = 7$  and



**Fig. 4** Construction based generation scheme

$LS_3 = 3$ . Accordingly, the sets of time-feasible start times  $\bar{W}_i$  for  $i = 2$  and  $i = 3$  are updated to  $\bar{W}_2 = \{3, \dots, 7\}$  and  $\bar{W}_3 = \{0, \dots, 3\}$ . The current resource profile is shown in the upper part of Fig. 4. In the next iteration,  $\mathcal{E} = \{2, 3\}$  applies. Due to its smaller slack time, activity  $j^* = 3$  is selected. However, since activity 2 requires at least one unit of the resource ( $r_{21}^{min} = 1$ ) and two units are already taken by activity 1, there exists no time- and resource-feasible start time for activity 3. Therefore, an unscheduling step is performed. The result of this step is that activity 1 is unscheduled and its previous start time  $S_1 = 0$  is added to its tabu-list ( $\Theta_1 = \{0\}$ ). Furthermore, the latest start times of activity 2 and 3 are reset to nine. Afterwards, in iteration three, activity  $j^* = 1$  is scheduled to the earliest dominant time- and resource-feasible point in time not being tabued, i.e.  $S_1 = 1$ . The *ES-LS*-update leads to  $ES_2 = 4$ ,  $LS_2 = 8$ ,  $LS_3 = 4$  and  $ES_5 = 7$ . Consequently, the sets of time-feasible start times  $\bar{W}_i$  are updated as well. Now, the resource profile as shown in the middle of Fig. 4 results. In the fourth iteration, activity  $j^* = 3$  can be scheduled feasibly at  $S_3 = 4$  and, in the fifth iteration, the start time of activity  $j^* = 2$  is set to point in time four too. In the next iteration only activity 4 can be scheduled. Because of  $S_3 = 4$  the value  $ES_4$  was updated to 6. Since all units of the partially renewable resource are already in use, activity 4 can be feasibly scheduled earliest at point in time 8, thus results in  $S_4 = 8$  and consequently  $ES_5 = 9$ . The procedure ends with the scheduling of project end  $i = 5$  at  $t = 9$ . The result of the procedure is the time- and resource-feasible schedule  $S = (0, 1, 4, 4, 8, 9)$  as shown in the last resource profile in Fig. 4.

## 4.2 Relaxation-based generation scheme

In this section a relaxation-based generation scheme is presented, which is based on the branch-and-bound procedure of Watermeyer and Zimmermann (2020). The generation scheme starts with an optimal solution of the resource relaxation of the RCPSP/max- $\pi$ , e.g. the  $ES$ -schedule. If this schedule is not resource-feasible, we gradually reduce the resource consumption of conflict resources until we obtain a time- and resource-feasible schedule. This is achieved by iteratively adding resource constraints for the activities of a project, which are then converted into start time restrictions and added to the resource relaxation.

According to Watermeyer and Zimmermann (2020), the resource relaxation of the RCPSP/max- $\pi$  can be formulated as follows:

$$\left. \begin{array}{ll} \text{Minimize} & f(S) = S_{n+1} \\ \text{subject to} & S_j - S_i \geq \delta_{ij} \quad ((i, j) \in E) \\ & S_i \in W_i \quad (i \in \mathcal{V}) \end{array} \right\} (\text{RCPSP/max-}\pi^{\text{relax}})$$

Vector  $W = (W_i)_{i \in \mathcal{V}}$  is further on called start time restrictions of the activities  $i \in \mathcal{V}$  with  $W_i \subseteq \{0, 1, \dots, \bar{d}\}$  for all  $i \in \mathcal{V} \setminus 0$  and  $W_0 = \{0\}$ . If  $W_i := \{ES_i, \dots, LS_i\}$  for all  $i \in \mathcal{V}$  applies, the resource relaxation is equal to the PSP/max. In the following,  $S_T(W)$  describes the feasible area of problem RCPSP/max- $\pi^{\text{relax}}$ , i.e. the  $n+1$ -dimensional space containing all schedules that observe the given start time restrictions. The unique minimal point of this space, which exists unless  $S_T(W) = \emptyset$ , is called  $ES(W)$  and can be determined by an adapted Label Correcting Algorithm in  $\mathcal{O}(|\mathcal{V}| |E| (B+1))$  with  $B$  as number of start time breaks in  $W$  (Watermeyer & Zimmermann, 2020).

In the following, the detailed procedure of our relaxation-based generation scheme is presented, which is also shown in Algorithm 3. In the initialization process, analogous to the construction-based generation scheme, for each activity  $i \in \mathcal{V}$ , the cumulative resource consumption  $r_{ik}^c(t)$  for each point in time  $t \in \bar{W}_i$  and each partially renewable resource  $k \in \mathcal{R}$  is calculated as well as the minimal possible resource demand  $r_{ik}^{\min}$  over time. Afterwards, for each activity  $i \in \mathcal{V}$  a set  $W_i$  of all potential start times is established containing all points in time  $t \in \bar{W}_i$  excluding those who are certainly known to be resource-infeasible. Based on  $W_i$ , the maximal cumulative resource demand  $r_{ik}^{\max}$  is computed for each activity  $i \in \mathcal{V}$  and resource  $k \in \mathcal{R}$ . Furthermore, a lower bound  $r_{ik}^{LB} := r_{ik}^{\min}$  and an upper bound  $r_{ik}^{UB} := r_{ik}^{\max}$  for the resource consumption allowed for activity  $i \in \mathcal{V}$  are initialized, which at first correspond to the minimal and maximal possible resource usage, respectively. The initialization step concludes with the set up of a counter  $u$ , which is set to zero, and an empty tabu list  $\Theta_k$  for each resource  $k \in \mathcal{R}$ .

In the main step, which is only performed if a time-feasible schedule exists, i.e.  $S_T \neq \emptyset$ , and if the minimum possible resource requirement  $r_{ik}^{\min}$  of all activities  $i \in \mathcal{V}$  in total does not exceed the given capacity  $R_k$  for any resource  $k \in \mathcal{R}$ , the unique minimal point  $S = ES(W)$  of the feasible area of the RCPSP/max- $\pi^{\text{relax}}$  is determined. For the resulting schedule all resources for which the total resource consumption of all activities exceeds the given capacity in the restricted periods are identified and stored in set  $\mathcal{R}^{\text{conflict}}$ . If this is not the case for any partially renewable resource, i.e. the set  $\mathcal{R}^{\text{conflict}}$  is empty, schedule  $S$  is time- and resource-feasible and the generation scheme terminates. In this case, due to the pursued objective of minimizing the project duration, the minimum point  $ES(W)$  is optimal. Otherwise, based on a given priority rule, a resource  $k^* \in \mathcal{R}^{\text{conflict}}$  is selected. In the following, for at least one activity using resource  $k^*$  in the current schedule in the capacitated periods, the resource usage in those periods should be decreased. For this, a set  $\mathcal{V}^{\text{pot}}$  of all activities  $i \in \mathcal{V}$ , whose

**Algorithm 3** Relaxation-based generation Scheme

---

**Input:** RCPSP/max- $\pi$  instance

- 1: Determine longest path lengths  $d_{ij}$  for all  $i, j \in \mathcal{V}$
- 2: Set  $ES_i = d_{0i}$ ,  $LS_i = -d_{i0}$  for all  $i \in \mathcal{V}$
- 3: Determine  $r_{ik}^c(t)$  for all  $i \in \mathcal{V} \setminus \{0\}$ ,  $k \in \mathcal{R}$  and  $t \in \bar{W}_i$
- 4:  $r_{ik}^{\min} := \min_{t \in \bar{W}_i} r_{ik}^c(t)$  for all  $i \in \mathcal{V}$  and  $k \in \mathcal{R}$
- 5:  $W_i := \{t \in \bar{W}_i \mid r_{ik}^c(t) \leq R_k - \sum_{j \in \mathcal{V} \setminus \{i\}} r_{ik}^{\min} \text{ for all } k \in \mathcal{R}\}$  for all  $i \in \mathcal{V}$
- 6:  $r_{ik}^{\max} := \max_{t \in \bar{W}_i} r_{ik}^c(t)$  for all  $i \in \mathcal{V}$  and  $k \in \mathcal{R}$
- 7:  $r_{ik}^{LB} := r_{ik}^{\min}$ ,  $r_{ik}^{UB} := r_{ik}^{\max}$  for all  $k \in \mathcal{R}$  for all  $i \in \mathcal{V}$
- 8:  $u := 0$ ,  $\Theta_k = \emptyset$  for all  $k \in \mathcal{R}$ , feas=false
- 9: **if**  $S_T(W) = \emptyset$  **then** terminate
- 10: **while** feas=false **do**
- 11:    $S := ES(W)$
- 12:    $\mathcal{R}^{\text{conflict}} := \{k \in \mathcal{R} \mid \sum_{i \in \mathcal{V}} r_{ik}^c(S_i) > R_k\}$
- 13:   **if**  $\mathcal{R}^{\text{conflict}} = \emptyset$  **then** feas=true
- 14:   **else**
- 15:     priority based choice of resource  $k^* \in \mathcal{R}^{\text{conflict}}$
- 16:      $\mathcal{V}^{\text{pot}} := \{i \in \mathcal{V} \mid r_{ik^*}^c(S_i) > 0 \wedge r_{ik^*}^c(S_i) - r_{ik^*}^{LB} > 0 \wedge i \notin \Theta_{k^*}\}$
- 17:     **if**  $\mathcal{V}^{\text{pot}} = \emptyset$  **then**  $u := u + 1$  **and** Reverse Step
- 18:     **else**
- 19:       priority based choice of activity  $i^* \in \mathcal{V}^{\text{pot}}$
- 20:        $r_{i^*k^*}^{UB} := r_{i^*k^*}^c(S_{i^*}) - r_{i^*k^*}$
- 21:       **for**  $t \in W_i$  **do**
- 22:         **if**  $r_{i^*k^*}^c(t) > r_{i^*k^*}^{UB}$  **then**  $W_i := W_i \setminus \{t\}$
- 23:       **if**  $S_T(W) = \emptyset$  **then**
- 24:          $r_{i^*k^*}^{UB} := r_{i^*k^*}^c(S_{i^*})$
- 25:          $\Theta_{k^*} := \Theta_{k^*} \cup \{i^*\}$
- 26:       **else**
- 27:         update  $W_i$ ,  $r_{ik}^{\min}$  and  $r_{ik}^{\max}$  for all  $i \in \mathcal{V}$  and  $k \in \mathcal{R}$
- 28: **return**  $S$

---

relevant cumulative resource consumption of resource  $k^*$  can be reduced, is determined. If  $\mathcal{V}^{\text{pot}}$  is empty, a so-called reverse step is done. Otherwise, based on a certain priority rule, an activity  $i^* \in \mathcal{V}^{\text{pot}}$  is selected. For the chosen activity  $i^*$  the maximal cumulative resource consumption  $r_{i^*k^*}^{\max}$  is set to  $r_{i^*k^*}^c(S_{i^*}) - r_{i^*k^*}$ . Afterwards, this new restriction for the resource consumption of activity  $i^*$  is converted into a start time restriction, i.e. all points in time  $t \in W_i$  leading in a higher resource requirement than  $r_{i^*k^*}^{\max}$  are removed from the set  $W_i$ . If this results in an empty feasible area of the corresponding RCPSP/max- $\pi^{\text{relax}}$ , the additional start time restriction is undone and activity  $i^*$  is added to the tabu list of resource  $k^*$ . Otherwise,  $W_i$ ,  $r_{ik}^{\min}$  and  $r_{ik}^{\max}$  are updated for all  $i \in \mathcal{V}$  and  $k \in \mathcal{R}$ . The main step as just described is executed until a time- and resource-feasible schedule is found or a termination criterion, e.g. a maximum number of reverse steps, is reached.

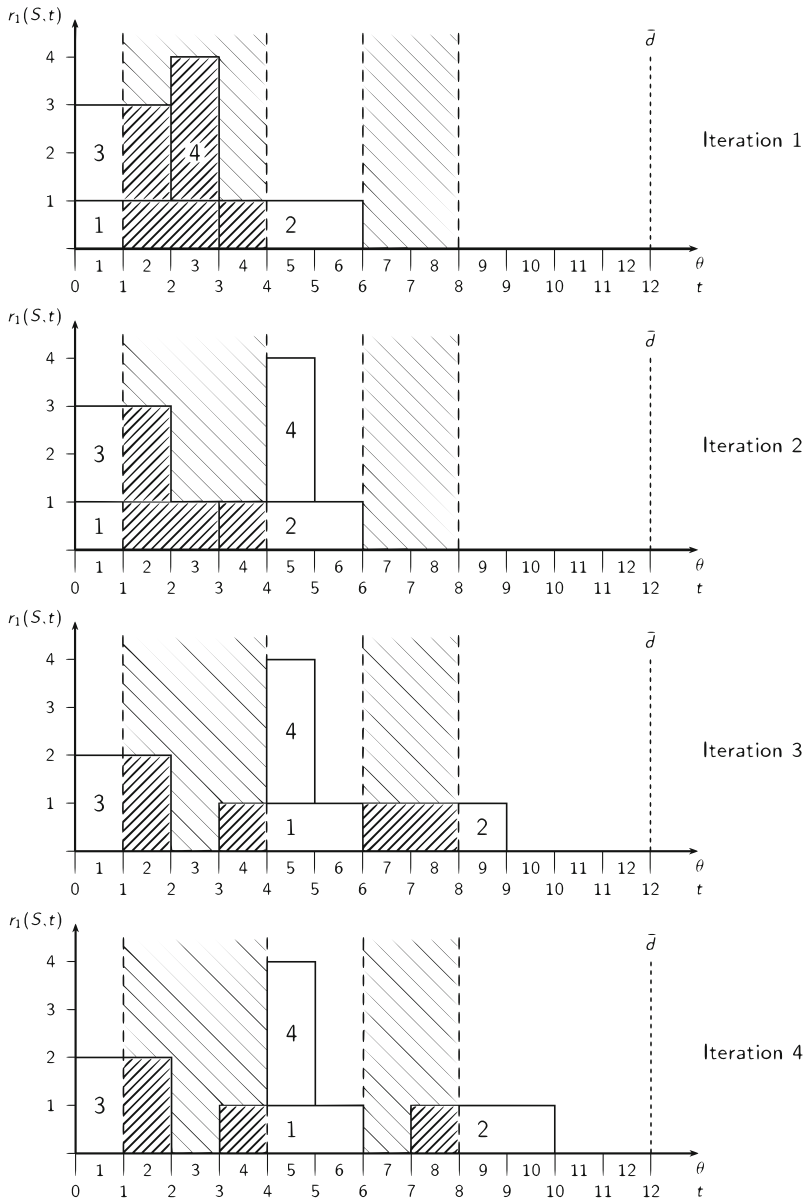
The reverse step is executed if there is no feasible possibility to reduce the resource consumption for the selected conflict resource  $k^*$ . To count the number of reverse steps already performed, the counter  $u$  is used. If  $u$  is higher than a given maximal number of reverse steps  $\hat{u}$ , the generation scheme aborts without finding a feasible schedule. Otherwise, counter  $u$  is increased by one. Aim of the reverse step is to undo start time restrictions added in previous iterations that result in conflict resources or that cause the resource requirements of currently existing conflict resources to be irreducible. Since these disrupting start time restrictions cannot be easily identified, four different reverse step strategies were investigated.

The first strategy ( $\sigma = 1$ ) simply reverses a random number of recently added start time restrictions. If the second strategy ( $\sigma = 2$ ) is chosen, we always return to the initial state ( $W_i = \{ES_i, \dots, LS_i\}$  for all  $i \in \mathcal{V}$ ). The third strategy ( $\sigma = 3$ ) resets the start time restrictions for activities  $i \in \mathcal{V}$  that actually use the conflict resource  $k^*$  whose resource consumption in the capacitated periods could not be reduced and whose demand could in general be decreased, i.e.  $r_{ik^*}^c(S_i) - r_{ik^*}^{LB} > 0$ . Finally, the fourth strategy ( $\sigma = 4$ ) tries to find out the reasons why the heuristic is stuck. For this purpose, quite similar to the unscheduling step of the construction-based generation scheme it is determined whether binding time restrictions or resource constraints are the reason that the resource consumption of  $k^*$  cannot be reduced in order to specifically reset them. Finishing one iteration, regardless of the selected reverse step strategy, the tabu list  $\Theta_{k^*}$  is cleared and finally,  $W_i$ ,  $r_{ik}^{min}$  and  $r_{ik}^{max}$  are updated for all activities  $i \in \mathcal{V}$  and resources  $k \in \mathcal{R}$ .

For the choice of resource  $k^*$  the priority rules RDd ("maximal total Resource Demand dynamic first") and ROd ("maximal resource capacity overrun first - dynamic") were tested. For the activity selection the rules TFd (in contrast to our first generation scheme, the activity with the largest total float is prioritized here), GRDd ("greatest resource demand first - dynamic") and GRDTd ("greatest resource demand per time unit first - dynamic") were used. Analogous to the first generation scheme, the relaxation-based generation scheme can be easily converted into a multistart procedure by calculating selection probabilities according to the chosen priority rules and performing roulette wheel selections for instance.

The relaxation-based generation scheme has a time complexity of  $\mathcal{O}(|\mathcal{V}| |\mathcal{R}| \bar{d} s)$ , if there does not occur any reverse step, since at most  $s = \sum_{i \in \mathcal{V}} \sum_{k \in \mathcal{R}} r_{ik}^{max} / r_{ik}$  resource constraints can be inserted. The time complexity of the reverse step depends on the chosen reverse strategy and can at worst be  $\mathcal{O}(\max(|\mathcal{V}|^2, |\mathcal{V}| |\mathcal{R}|))$ .

The relaxation-based generation scheme is demonstrated for the example shown in Fig. 1. First, the initialization process is executed including the set up of the start time restrictions  $W_i$  for each activity  $i \in \mathcal{V}$ . In this example resulting in  $W_0 = \{0\}$ ,  $W_1 = \{0, \dots, 6\}$ ,  $W_2 = \{3, \dots, 9\}$ ,  $W_3 = \{0, \dots, 9\}$ ,  $W_4 = \{2, \dots, 11\}$  and  $W_5 = \{6, \dots, 12\}$ . In the main step,  $S = ES(W)$  is calculated. The resulting schedule  $S = (0, 0, 3, 0, 2, 6)$  drawn in the first resource profile in Fig. 4 is not resource-feasible because the resource usage in the capacitated periods exceeds the capacity  $R_1$  ( $\sum_{i \in \mathcal{V}} r_{i1}^c(S_i) = 8 > R_1 = 4$ ). To reduce the resource consumption, the resource requirements of the activities  $\mathcal{V}^{pot} = \{1, 2, 3, 4\}$  can be reduced. Since activity 4 uses the most resources in the current schedule, it is selected and its maximal allowed resource usage is set to  $r_{41}^{UB} := r_{41}^c(S_4) - r_{41} = 3 - 3 = 0$ . This results in  $W_4 = \{4, 5, 8, \dots, 11\}$  whereas all other start time restrictions remain unchanged. Based on this, the second iteration results in Schedule  $S = ES(W) = (0, 0, 3, 0, 4, 6)$ , which is still not resource-feasible. Since the resource consumption of activities 1 and 2 are identical, the activity with the smallest index, i.e. activity  $j^* = 1$ , is selected here and its maximal allowed resource usage is restricted to  $r_{11}^{UB} := r_{11}^c(S_1) - r_{11} = 2 - 1 = 1$ . Based on this, the start time restrictions are updated as follows:  $W_1 = \{3, 4\}$ ,  $W_2 = \{6, 7, 8, 9\}$  and  $W_5 = \{9, \dots, 12\}$ . In the third iteration, the schedule  $S = ES(W) = (0, 3, 6, 0, 4, 9)$  is obtained. We chose activity  $j^* = 2$  from the set  $\mathcal{V}^{pot} = \{1, 2, 3\}$  and restrict its resource consumption to  $r_{21}^{UB} := 2 - 1 = 1$ . Thus,  $W_2 = \{7, 8, 9\}$  and  $W_5 = \{10, \dots, 12\}$  applies. Finally, in the forth iteration, schedule  $S = ES(W) = (0, 3, 7, 0, 4, 10)$  drawn in the last resource profile in Fig. 4 is time- and resource-feasible and the generation scheme terminates.



**Fig. 5** Relaxation based generation scheme



## 5 Results

This section provides the results of an experimental performance analysis of the two generation schemes. All combinations of different priority rules and reverse step strategies as introduced before were tested. For each combination the generation schemes were executed as multistart procedure with one deterministic and 100 stochastic runs. The maximal number of unscheduling or reverse steps per run were limited to the instance dependent number of activities  $n$ .

Within the experimental performance analysis, we used test instances for the RCPSP/max- $\pi$  generated by Watermeyer and Zimmermann (2020), which are based on the well-known benchmark UBO-instances for the RCPSP/max developed and described by Schwindt (1998). We used the test sets UBO50 $\pi$ , UBO100 $\pi$ , and UBO200 $\pi$  with  $n \in \{50, 100, 200\}$  real activities and  $m = 30$  partially renewable resources. Each of these test sets consists of 243 instances with different specifications. For better comparability, analogous to Watermeyer the maximum project duration  $\bar{d}$  is calculated by  $\bar{d} = \sum_{i \in \mathcal{V}} \max\{p_i, \max_{(i,j) \in E} \delta_{ij}\}$ . Note, that for the RCPSP/max- $\pi$  in contrast to the RCPSP/max this value does not represent an upper bound for the project duration and consequently no feasible solution may exist. All instances that are known to be either trivial or infeasible assuming  $\bar{d}$  as introduced before are excluded. Consequently, a total of 502 instances of testset UBO50 $\pi$ , 479 instances of testset UBO100 $\pi$ , and 466 instances of testset UBO200 $\pi$  remain.

In order to better benchmark the results of our generation schemes, they are compared to those of the presented MIP formulation as well as to those of the best-performing branch-and-bound procedure of Watermeyer and Zimmermann (2022). All solution methods are coded in C++. To solve the MIP we used the ILOG IBM CPLEX 12.0 solver with a runtime limit of 3600 s, whereas Watermeyer and Zimmermann (2022) specifies a time limit of 300 s for the UBO50 $\pi$  and UBO100 $\pi$  test sets and 600 s for the UBO200 $\pi$  test set, which was adopted for our generation schemes. All runs were done on an Intel Core i7-7700K CPU with 4.2 GHz and 64 GB RAM under Windows 10 on a single thread.

Since the feasibility problem of the RCPSP/max- $\pi$  is  $\mathcal{NP}$ -complete in the strong sense (cf. Watermeyer, 2021, p.34), both heuristic generation schemes do not necessarily find a feasible solution for an instance of the RCPSP/max- $\pi$  even if a feasible solution exists. Therefore, in a first step in Tabel 1 the percentage of instances for which a feasible solution could be found ( $\%_{feas}$ ) was evaluated for both generation schemes and compared to those obtained by the time indexed formulation solved by the solver ILOG IBM CPLEX (MIP) and the partitioning-based branch-and-bound procedure (B&B) of Watermeyer and Zimmermann (2022).

The results show that both generation schemes are able to find feasible solutions for a high percentage of the tested instances (at worst 91.04%), especially using the resource-based priority rules. However, for the instances with 50 or 100 real activities, the construction-based method using the priority rule RD for the start time selection outperforms the relaxations-based generation scheme for all tested combinations of priority rules and reverse step strategies. For the instances with 200 real activities, both generation schemes are able to find feasible solutions for almost all tested instances regardless of the selected priority rules and reverse step strategies (at worst 99.36%). For both generation schemes as well as for the branch-and-bound procedure, it can be observed that the proportion of feasibly solved instances rises with increasing instance size, whereas for the MIP the percentage decreases significantly. This indicates that due to the planning horizon, which increases with growing instance size, and the overall available resource capacity, there is more flexibility for planning the activities, which is exploited especially in the problem-specific solution methods.



**Table 1** Performance regarding the percentage of feasibility

			UBO50 $\pi$	UBO100 $\pi$	UBO200 $\pi$	
constr. GS	$LSTd$	$T_{min}$	96.81	98.07	99.36	
		$RD$	97.21	98.20	99.57	
		$RL$	97.21	98.07	99.36	
	$TFd$	$T_{min}$	96.41	98.07	99.36	
		$RD$	97.21	98.20	99.57	
		$RL$	97.21	98.07	99.36	
	relax. GS	$RDd$	$\sigma = 1$	91.04	95.82	99.36
			$\sigma = 2$	92.23	96.24	99.36
			$\sigma = 3$	93.23	96.45	99.36
			$\sigma = 4$	93.63	96.66	99.57
			$\sigma = 1$	91.63	96.03	99.36
			$\sigma = 2$	92.43	96.45	99.36
$\sigma = 3$			93.22	96.87	99.57	
$\sigma = 4$			94.02	97.29	99.57	
$\sigma = 1$			91.63	96.03	99.36	
$\sigma = 2$			92.63	97.08	99.57	
$\sigma = 3$			93.82	97.29	99.57	
$\sigma = 4$			94.22	97.70	99.57	
MIP	$ROd$	$\sigma = 1$	91.63	96.03	99.36	
		$\sigma = 2$	92.63	96.66	99.36	
		$\sigma = 3$	93.43	97.08	99.57	
		$\sigma = 4$	93.83	97.08	99.57	
		$\sigma = 1$	92.03	92.63	99.36	
		$\sigma = 2$	93.22	97.08	99.36	
		$\sigma = 3$	94.02	97.29	99.57	
		$\sigma = 4$	94.44	97.29	99.57	
		$\sigma = 1$	92.03	97.70	99.57	
		$\sigma = 2$	93.43	97.49	99.57	
		$\sigma = 3$	94.22	97.70	99.57	
		$\sigma = 4$	94.44	97.91	99.57	
MIP			96.81	63.26	25.11	
B&B			98.80	98.96	100.00	

Beside the percentage of feasibly solved instances, for each tested generation scheme specification, the number of instances for which it finds the best solution over both generation schemes ( $\#^{best}$ ) was counted and also the average percentage deviation ( $\varnothing^{gap}$ ) from this best solution over all feasibly solved instances was determined. The results are shown in Table 2.

For some instances the different specifications come to the same objective function value, so the sum of  $\#^{best}$  per instance size is greater than the number of instances contained in the instance set. But for a part of the instances, there are also strong variations between the best solutions of the tested specifications of both generation methods. For the construction-based generation scheme, the quality of the solutions generated with the priority rules *LSTd* and *TFd* are quite similar. However, regarding the average gap both are outperformed by the

**Table 2** Performance of the generation schemes

			UBO50 $\pi$		UBO100 $\pi$		UBO200 $\pi$	
			# <i>best</i>	$\varnothing$ <i>gap</i>	# <i>best</i>	$\varnothing$ <i>gap</i>	# <i>best</i>	$\varnothing$ <i>gap</i>
constr. GS	<i>LSTd</i>	<i>T<sub>min</sub></i>	177	2.83	85	4.65	92	4.27
		<i>RD</i>	221	3.46	147	6.21	152	5.35
		<i>RL</i>	259	4.20	206	6.06	224	5.28
	<i>TFd</i>	<i>T<sub>min</sub></i>	141	3.04	64	5.99	78	4.53
		<i>RD</i>	202	3.76	134	7.35	143	5.42
		<i>RL</i>	242	3.59	170	7.32	179	5.33
	<i>TFd</i>	$\sigma = 1$	105	4.00	47	4.48	60	4.60
		$\sigma = 2$	120	2.62	65	2.95	73	2.83
		$\sigma = 3$	123	2.83	53	3.86	64	3.59
		$\sigma = 4$	124	2.84	59	4.05	64	4.10
	<i>RDd</i>	$\sigma = 1$	99	4.24	44	4.68	57	4.98
		$\sigma = 2$	114	2.84	62	3.11	66	3.03
		$\sigma = 3$	110	3.16	50	4.27	59	3.79
		$\sigma = 4$	109	3.22	57	4.30	62	4.37
	<i>GRDTd</i>	$\sigma = 1$	93	4.16	47	4.45	58	4.74
		$\sigma = 2$	122	2.82	69	2.92	73	2.84
		$\sigma = 3$	119	3.12	53	3.80	61	3.66
		$\sigma = 4$	116	3.19	57	4.02	63	4.22
relax. GS	<i>TFd</i>	$\sigma = 1$	103	3.96	48	4.41	60	4.44
		$\sigma = 2$	132	2.38	72	2.42	77	2.41
		$\sigma = 3$	123	2.54	54	4.23	62	3.34
		$\sigma = 4$	124	2.63	59	3.75	66	3.73
	<i>GRDd</i>	$\sigma = 1$	97	4.23	43	4.55	62	4.92
		$\sigma = 2$	110	2.80	63	3.07	71	3.00
		$\sigma = 3$	110	3.12	48	4.24	60	3.74
		$\sigma = 4$	106	3.17	58	4.25	65	2.30
	<i>ROd</i>	$\sigma = 1$	104	4.11	48	4.39	60	4.71
		$\sigma = 2$	123	2.75	69	2.87	74	2.80
		$\sigma = 3$	120	3.12	54	3.72	63	3.59
		$\sigma = 4$	120	3.13	59	4.00	66	4.22

time-based rule  $T_{min}$  for the smaller as well as the larger instances, even if they provide the best solutions over all tested specifications for a higher number of instances. This can be explained by the fact that – although they generate very good results for many instances – they perform comparatively poor for some of the instances, resulting in a higher average gap. The results also show that for the relaxation-based generation scheme, the combination of the resource selection rule  $ROd$  and reverse strategy  $\sigma = 2$  performs best.

Now, the results of the best performing combinations of priority rules and strategies for both generation schemes are compared to those obtained by the time indexed formulation solved by the solver ILOG IBM CPLEX (MIP). For all solution methods, the average percentage deviation  $\varnothing^{gap}$  from the best found solution over all feasibly solved instance and the average computation time  $\varnothing^{time}$  in seconds were examined. The results are displayed in Table 3.

**Table 3** Performance of the generation schemes in comparison

	UBO50 $\pi$		UBO100 $\pi$		UBO200 $\pi$	
	$\varnothing_{gap}$	$\varnothing_{time}$	$\varnothing_{gap}$	$\varnothing_{time}$	$\varnothing_{gap}$	$\varnothing_{time}$
constr. GS	7.43	24.48	9.45	89.94	6.48	262.23
relax. GS	6.89	59.24	8.58	172.71	5.25	332.62
MIP	0.37	2175.01	22.58	2985.09	25.31	3198.26

For the MIP, in general, it can be observed, that the larger the instance, the higher the achieved gap and the required computation time. To improve the performance of the MIP, higher time limits of up to eight hours were tested. However, it can be observed that several of the tested instances, especially the larger ones, still cannot be solved with a small gap or even optimally. Therefore, solving the instances using a solver is not expedient even with an extended solution time. For the generation schemes, the gap first rises, but then falls again for the instances with 200 real activities. For the UBO50 $\pi$  instances, the MIP outperforms the heuristics. With increasing instance size, the generation schemes are not only able to solve more instances feasibly than the MIP, but also the constructed schedules are better for UBO100 $\pi$  and significantly better for UBO200 $\pi$  compared to the MIP. Regarding the average computation time, it can be observed, that the relaxation-based generation scheme takes more time than the construction-based generation scheme. In particular, returning to the initial state as reverse step strategy causes a high time overhead compared to the other reverse step strategies. For all tested instance sizes, the average computation time solving the MIP is drastically higher than for both generation schemes. Overall, the relaxation-based generation scheme generates a marginal smaller gap than the construction-based, but also takes more time.

## 6 Conclusion

In this paper we presented two generation schemes for the resource-constrained project scheduling problem with partially renewable resources and generalized precedence constraints (RCPSP/max- $\pi$ ). In the first one, a successive scheduling of the activities is done to finally construct a feasible solution. In the second, starting from the ES-Schedule, which in general is resource-unfeasible, resource conflicts are solved until a feasible solution has been found. The results of a comprehensive experimental performance analysis show, that both generation schemes are able to generate feasible solutions for nearly all tested instances in a short time and that the relaxation-based generation schemes finds marginal better solutions while requiring slightly more time.

In order to improve the solutions obtained in this way, further research should address the development of improvement heuristics, such as a genetic algorithm or a fix-and-optimize heuristic. Also, machine learning approaches seem promising. For example, our generation schemes could be extended to include a learning component that, from run to run, gives higher weight to promising scheduling sequences or resource conflict solutions in the selection process and, in contrast, gives less preference to steps that have led to unscheduling or reverse steps. In addition, projects with renewable and partially renewable resources should be considered, since in practice both resource types usually occur together.

**Funding** Open Access funding enabled and organized by Projekt DEAL. No funding was received to assist with the preparation of this article.

## Declarations

**Conflict of interest** There are no interests to declare.

**Ethical approval** This article does not contain any studies with human participants or animals performed by any of the authors.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- Alvarez-Valdés, R., Crespo, E., Tamarit, J., & Villa, F. (2006). A scatter search algorithm for project scheduling under partially renewable resources. *Journal of Heuristics*, 12(1), 95–113.
- Alvarez-Valdés, R., Crespo, E., Tamarit, J., & Villa, F. (2008). Grasp and path relinking for project scheduling under partially renewable resources. *European Journal of Operational Research*, 189, 1153–1170.
- Alvarez-Valdés, R., Tamarit, J., & Villa, F. (2015). Partially renewable resources. In C. Schwindt & J. Zimmermann (Eds.), *Handbook on Project Management and Scheduling* (Vol. 1, pp. 203–227). Springer.
- Böttcher, J., Drexl, A., Kolisch, R., & Salewski, F. (1999). Project scheduling under partially renewable resource constraints. *Management Science*, 45(4), 543–559.
- Drexl, A., Juretzka, J., & Salewski, F. (1993). Academic course scheduling under workload and changeover constraints. *Working paper of the University of Kiel* No. 337.
- Drexl, A., & Salewski, F. (1997). Distribution requirements and compactness constraints in school timetabling. *European Journal of Operational Research*, 102(1), 193–214.
- Floyd, R. W. (1962). Algorithm 97: Shortest path. *Communications of the ACM*, 5(6), 345.
- Franck, B., Neumann, K., & Schwindt, C. (2001). Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling. *OR-Spektrum*, 23, 297–324.
- Karnebogen, M., & Zimmermann, J. (2021). A generation scheme for the resource-constrained project scheduling problem with partially renewable resources and time windows. In: *Book of Extended Abstracts of 17th International Conference on Project Management and Scheduling*, Toulouse. pp. 195–198.
- Karnebogen, M., & Zimmermann, J. (2022). A relaxation-based generation scheme for the RCPSP/max,  $\pi$ . In: *Book of Extended Abstracts of 18th International Conference on Project Management and Scheduling*, Ghent. pp. 72–75.
- Neumann, K., Schwindt, C., & Zimmermann, J. (2003). *Project scheduling with time windows and scarce resources*. Springer.
- Talbot, F. B. & J. H. Patterson. (1978). An efficient integer programming algorithm with network cuts for solving resource-constrained scheduling problems. *Management Science*, 24(11), 1163–117.
- Schirmer, A. (1999). *Project scheduling with scarce resources: Models, methods and applications*. Springer.
- Schwindt, C. (1998). Generation of resource-constrained project scheduling problems subject to temporal constraints. *Technical Report WIOR-543*, University of Karlsruhe.
- Watermeyer, K. (2021). *Projektplanung mit partiell erneuerbaren Ressourcen*. Dürren: Shaker.
- Watermeyer, K., & Zimmermann, J. (2020). A branch-and-bound procedure for the resource-constrained project scheduling problem with partially renewable resources and general temporal constraints. *OR Spektrum*, 42(2), 427–460.
- Watermeyer, K., & Zimmermann, J. (2022). A partition-based branch-and-bound algorithm for the project duration problem with partially renewable resources and general temporal constraints. *OR Spektrum*, 44(2), 575–602.
- Watermeyer, K., & Zimmermann, J. (2023). A constructive branch-and-bound algorithm for the project duration problem with partially renewable resources and general temporal constraints. *Journal of Scheduling*, 26, 95–111.

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.