A Generation Scheme for the Resource-Constrained Project Scheduling Problem with Partially Renewable Resources and Time Windows

Mareike Karnebogen and Jürgen Zimmermann

Clausthal University of Technology, Germany
mareike.karnebogen@tu-clausthal.de, juergen.zimmermann@tu-clausthal.de

Keywords: Project scheduling, Partially renewable resources, RCPSP/max,π

1 Introduction

Partially renewable resources are a generalized form of renewable and non-renewable resources. For each partially renewable resource a capacity is given, which only applies to predetermined time periods. This type of resources was first mentioned by Böttcher *et al.* (1996) in the context of projects restricted by precedence constraints (RCPSP/π). In this paper the resource-constrained project scheduling problem with partially renewable resources and time windows (RCPSP/max,π) is considered, which for example allows to take more flexible working arrangements in case of deployment planning into account. The RCPSP/max,π is a generalization of the RCPSP/π and thus hard to solve (NP-hard). Therefore, we present a generation scheme for the RCPSP/max,π to construct feasible solutions within a short period of time. Section 2 contains a general description of the problem and a mixed-integer linear formulation (MILP). In Section 3 the developed generation scheme is presented. Finally, Section 4 includes a preliminary performance analysis, in which the results of our generation scheme are compared to results obtained by *IBM CPLEX* applied on the MIP model of Watermeyer *et al.* (2018).

2 Problem description

The activities and temporal constraints of the resource-constrained project scheduling problem with time windows and partially renewable resources (RCPSP/max,π) can be represented graphically as an activity-on-node network. The nodes correspond to the activities of the project $V = \{0, 1, \ldots, \text{n+1}\}$ consisting of the fictitious project start 0, n real activities, and the fictitious project completion $n+1$. Each activity $i \in V$ has a deterministic processing time $p_i \in \mathbb{Z}_{\geq 0}$ during which the activity can not be interrupted. General temporal constraints between two activities $i, j \in V$ are represented by arcs $\langle i, j \rangle$ between the corresponding nodes of the network. The arc weight $\delta_{ij} \in \mathbb{Z}$ of an arc $\langle i, j \rangle \in E$ corresponds to a minimal time lag between the start times of activity i and activity j which has to be satisfied. The maximal project duration given by $\bar{d}$ can be represented by an arc from $n+1$ to 0 weighted with $-\bar{d}$. As it is common practice in literature, let d_{ij} be the longest distance from node $i \in V$ to node $j \in V$ in the project network which can be calculated by the Floyd-Warshall tripel algorithm. Then $ES_i = d_{0i}$ and $LS_i = -d_{i0}$ are the earliest and latest start time of activity $i \in V$, respectively, and $\mathcal{T}_i = \{ES_i, ES_i + 1, ..., LS_i\}$ implies all time-feasible integer start time points of activity i.

When executing the activities a set of partially renewable resources $\mathcal{R} = \{1, \ldots, m\}$ have to be observed. Each activity $i \in V$ has a resource demand $r_{ik} \in \mathbb{Z}_{\geq 0}$ for each period it is executed. In addition each partially renewable resource $k \in \mathcal{R}$ has a resource capacity R_k, which only applies to a subset of not necessarily consecutive time periods of

the given planning horizon $\Pi_k \subseteq \{1, 2, \ldots, \bar{d}\}$. For all time periods not contained in Π_k it is assumed that the capacity is not restricted. Obviously, the relevant composite resource consumption of activity i of resource k depends on the number of time periods activity i is in execution during the capacitated time periods of resource k and can be calculated by $r^c_{ik}(S_i) = |\{S_i + 1, S_i + 2, \ldots, S_i + p_i\} \cap \Pi_k| \cdot r_{ik}$ (Watermeyer *et al.* 2018).

The objective of our problem is to find a time- and resource-feasible schedule $S = (S_0, S_1, \ldots, S_{n+1})$ which minimizes the project duration S_{n+1}. A schedule is called time-feasible, if $S_j - S_i \geq \delta_{ij}\ \forall \langle i, j\rangle \in E$, what means that all temporal constraints are observed. To obtain a resource feasible schedule the accumulated resource demand occurring across all capacitated periods must not exceed R_k for any partially renewable resource $k \in \mathcal{R}$. Formally, the RCPSP/max,π described above can be formulated as follows:

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

3 Generation scheme

Algorithm 1 shows our generation scheme which is based on the generation scheme for the RCPSP/max developed by Franck *et al.* (2001). Let C be the set of scheduled activities. In the initialization process project start 0 is fixed at $t = 0$ and appended to C. Also counter u is set to zero. In the main step for each partially renewable resource $k \in \mathcal{R}$ and each activity not scheduled so far the minimal composite demand r^{min}_{ik} and the maximal composite demand r^{max}_{ik} is calculated. As well as the remaining capacity RC_k

Algorithm 1 Generation Scheme

$\mathcal{C} := \{0\}$, $S_0 := 0$, $u := 0$
$r_{ik}(t)$ for all $i \in V$ and $k \in \mathcal{R}$ and $t \in \mathcal{T}_i$
while $\mathcal{C} \neq V$ **do**
  $r^{min}_{ik} := \min_{t \in \mathcal{T}_i} r_{ik}(t)$ for all $i \in V \setminus \mathcal{C}$ and $k \in \mathcal{R}$
  $r^{max}_{ik} := \max_{t \in \mathcal{T}_i} r_{ik}(t)$ for all $i \in V \setminus \mathcal{C}$ and $k \in \mathcal{R}$
  $RC_k := R_k - \sum_{i \in V \setminus \mathcal{C}} r^{min}_{ik} - \sum_{i \in \mathcal{C}} r_{ik}(S_i)$ for all $k \in \mathcal{R}$
  for all $k \in \mathcal{R}$ **do**
    if $RC_k > \sum_{i \in V \setminus \mathcal{C}} r^{max}_{ik}$ **then** $\mathcal{R} = \mathcal{R} \setminus \{k\}$
  $\mathcal{E} := \{i \in V \setminus \mathcal{C} \mid Pred^{\prec_D}(i) \subseteq \mathcal{C}\}$
  priority based choice of an activity $j^* \in \mathcal{E}$ to be scheduled next
  $Z_{j^*} := \{t \in \mathcal{T}_{j^*} \setminus Tabu_{j^*} | r_{j^*kt} - r^{min}_{j^*k} \leq RC_k$ for all $k \in \mathcal{R}$ and
    $\min_{k \in \mathcal{R}}\{r_{j^*kt} - r_{j^*k\tau}\} < 0$ for all $\tau \in \mathcal{T}_{j^*} | \tau < t\}$
  if $Z_{j^*} = \emptyset$ **then** $u := u + 1$ and **Unschedule**
  else
    priority based choice of a point in time $t^* \in Z_{j^*}$ as start time of j^*
    $S_{j^*} := t^*$, $\mathcal{C} := \mathcal{C} \cup \{j^*\}$
    for all $h \in V \setminus \mathcal{C}$ **do** ($*$ update ES_h and LS_h $*$)
      $ES_h := \max(ES_h, S_{j^*} + d_{j^*h})$
      $LS_h := \min(LS_h, S_{j^*} - d_{hj^*})$
return S

which results from R_k minus the consumption of all scheduled activities $i \in C$ as well as the minimal necessary resource consumption r_{ik}^{min} of all not yet scheduled activities $i \in V \setminus C$. If RC_k outruns the maximal potential resource consumption r_{ik}^{max} of all activities $i \in V \setminus C$, the resource has no longer to be taken in consideration. Afterwards, the eligible set $\mathcal{E}$ containing all activities $i \in V \setminus C$ whose immediate predecessors regarding the distance order $\prec_D$ are scheduled is established (Neumann *et al.* 2003). An activity $j^* \in \mathcal{E}$ is selected based on a certain priority rule and the related set Z_{j^*} of resource- and time-feasible start times which are not dominated by an earlier feasible start time is determined. If $Z_{j^*} = \emptyset$ an unscheduling step is performed and counter u is increased by one. Otherwise a point in time $t^* \in Z_{j^*}$ is chosen based on a priority rule and assigned as S_{j^*} while j^* is added to C. Finally, for all unscheduled activities $i \in V \setminus C$ ES_i and LS_i are updated. This procedure is repeated until all activities $i \in V$ are scheduled time- and resource-feasible.

Algorithm 2 Unschedule

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** $\mathcal{U} := \{i \in \mathcal{C} \mid \min\{r_{ikS_i}, r_{j^*k}\} > 0$ for at least one $k \in \mathcal{R}\}$
5: **for all** $i \in \mathcal{U}$ **do**
6: $\quad \mathcal{C} := \mathcal{C} \setminus \{i\}$
7: $\quad Tabu_i = Tabu_i \cup \{S_i\}$
8: $Tabu_{j^*} := \emptyset$
9: **for all** $i \in \mathcal{C}$ with $S_i > \min_{h \in \mathcal{U}} S_h$ **do**
10: $\quad \mathcal{C} := \mathcal{C} \setminus \{i\}$
11: **for all** $h \in V \setminus \mathcal{C}$ **do**
12: $\quad ES_h := d_{0h}$
13: $\quad LS_h := -d_{h0}$
14: $\quad$ **for all** $i \in \mathcal{C}$ **do**
15: $\quad\quad ES_h := \max(ES_h, S_i + d_{ih})$
16: $\quad\quad LS_h := \min(LS_h, S_i - d_{hi})$

Algorithm 2 shows the unscheduling step which is performed if no time- and resource-feasible starting point of activity j^* exists. In case u is higher than a prescribed maximal number of unscheduling steps $\hat{u}$ the algorithm terminates. Otherwise a set U of activities $i \in C$ which have to be unscheduled and rescheduled in order to obtain a feasible schedule is determined. For this purpose, we first examine if one or more activities $i \in C$ restrict the scheduling timeframe of the chosen activity j^* i.e. increases ES_{j^*} or decreases LS_{j^*}. If $U = \emptyset$, we determine all those activities $i \in C$ using some resources $k \in \mathcal{R}$ activity j^* also requires for execution. Afterwards, all activities $i \in U$ are removed from C as well as all activities $i \in C$ with $S_i > \min_{h \in \mathcal{U}} S_h$ because some of them could possibly be executed earlier. Moreover, for all activities $i \in U$ the current start point S_i is forbidden by storing in the tabu-list $Tabu_i$ whereas $Tabu_{j^*}$ is cleared. Points in time $t \in Tabu_i$ can not be choosen as start time of activity i in the scheduling phase of the generation scheme (compare Algorithm 1 line 11). Finally, for all activities $i \in V \setminus C$ the values for ES_i and LS_i are recalculated.

4 Performance analysis

In order to evaluate the performance of our generation scheme we conduct a computational study performed on an Intel Core i7-7700K CPU with 4.2 GHz and 64 GB RAM under Windows 10. The generation scheme was coded in FICO® Xpress Optimization. The

instance set we used was established by Watermeyer *et al.* (2018) including 729 instances with 10, 20, and 50 activities and is based on the well-known UBO instances of Schwindt (1998) extended by 30 partially renewable resources with varying specifications.

Within the computational study for the choice of the activity scheduled next the priority rules LSTd (smallest "Latest Start Time dynamic" first) and TFd (smallest "Total Float dynamic" first), whereas for the choice of the scheduling point in time the objectives T_{min} (earliest "Start Time"), RD (minimal total "Resource Demand") and RL ("Resource Leveling") were tested. Starting with $\bar{d}$ as the RCPSP/max upper bound $\sum_{\langle i,j\rangle \in E} |\delta_{ij}|$, we perform a preprocessing to specify $\bar{d}$ including two deterministic runs with the priority rules LSTd-T_{min} and TFd-T_{min}. In the main step for each of the six combinations of the priority rules 100 runs were conducted per instance, whereby the choice of j^* and t^* is taken randomly based on selection probabilities. If a feasible solution with $S_{n+1} < \bar{d}$ is found, $\bar{d}$ is set to $S_{n+1} - 1$.

Table 1 shows preliminary results for all combinations of the established priority rules. Displayed are the percentage of instances (%feas) our generation scheme was able to find a feasible solution, the average percentage gap (%Gap) with regard to the best solution of the MIP found in at most 3.600 seconds and the average computing time (⌀CPU) in seconds required per run.

Table 1. Preliminary results of the computational study

		$UBO10^{\pi}$			$UBO20^{\pi}$			$UBO50^{\pi}$		
		T_{min}	RD	RL	T_{min}	RD	RL	T_{min}	RD	RL
LSTd	%feas	99.04	99.59	99.45	96.16	97.39	97.94	95.58	96.25	96.28
	%Gap	1.36	1.28	1.40	5.34	5.28	4.92	18.79	19.72	20.86
	⌀CPU	0.63	0.61	0.61	2.42	2.09	2.05	35.71	32.59	33.35
TFd	%feas	98.90	99.45	99.60	96.85	97.40	97.94	95.89	96.38	96.50
	%Gap	1.40	1.47	1.36	5.89	5.70	5.73	18.81	19.78	21.07
	⌀CPU	0.62	0.61	0.60	2.31	1.96	1.94	34.28	31.22	31.37

The results show that our generation scheme is able to generate feasible solutions for nearly all tested instances in particular by using the resource-based priority rules RD and RL. Note, that for some instances of $UBO50^{\pi}$ the generation scheme is able to find better solutions than the MIP in one hour. For the tested instances the quality of the solutions generated with the priority rules LSTd and TFd is very similar. For smaller instances the resource-based rules RD and RL mostly outperform the time-based rule T_{min}, whereas for larger instances it turns into its opposite. Besides higher gaps it can be observed that the computation time increases by a growing number of activities. In a next step the generation scheme should be coded in C++ and further priority rules should be examined.

References

Álvarez-Valdés R., E. Crespo, J.M. Tamarit and F. Villa, 2008, "GRASP and path relinking for project scheduling under partially renewable resources", *European Journal of Operational Research*, Vol. 189, pp. 1153-1170.

Böttcher J., A. Drexl, R. Kolisch, F. Salewski, 1996, "Project scheduling under partially renewable resource constraints", Technical Report, *Manuskripte aus den Instituten für Betriebswirt-schaftslehre 398*, University of Kiel.

Franck B., K. Neumann and C. Schwindt, 2015, "Truncated branch-and-bound, schedule-construction, and schedule-improvement procedures for resource-constrained project scheduling". *OR Spektrum*, Vol. 23, pp. 297-324.

Neumann K., C. Schwindt, J. Zimmermann, 2003, "Project scheduling with Time Windows and Scarce Resources", ed.2, Springer, Berlin.

Schirmer A., 1999, "Project scheduling with scarce resources: models, methods and applications", Dr. Kovač, Hamburg.

Schwindt C., 1998, "Generation of resource-constrained project scheduling problems subject to temporal constraints", *Technical Report WIOR-543*, University of Karlsruhe.

Watermeyer K. and J. Zimmermann, 2018, "A Branch-and-Bound Procedure for the Resource-Constrained Project Scheduling Problem with Partially Renewable Resources and Time Windows", *Proceedings of the 16th International Conference on Project Management and Scheduling*, Rom, pp. 259-262.