Exact Solution Procedures for RCPSP/π

Kai Watermeyer, Marc-André Aßbrock, Stefan Kreter, and Jürgen Zimmermann

Clausthal University of Technology, Germany
{marc-andre.assbrock,stefan.kreter,juergen.zimmermann}@tu-clausthal.de

Keywords: Project scheduling, Partially renewable resources, Integer linear programming, Constraint programming, Lazy clause generation.

1 Introduction

In this paper we consider exact solution procedures for the resource-constrained project scheduling problem under partially renewable resources (RCPSP/π). For the first time, a constraint programming (CP) formulation for that problem is presented. More precisely, we investigate the results obtained by the CP solver IBM CP Optimizer and the lazy clause generation (LCG) solver CHUFFED for this new formulation and take also two integer linear programming (ILP) models into account which are solved using IBM CPLEX.

For partially renewable resources the availability is associated to a subset of periods of the planning horizon. The resource requirement of activities that consume partially renewable resources for their execution may not exceed a resource capacity given for the corresponding subset of periods. For all other periods, the resource requirement is not restricted. Partially renewable resources include renewable and non-renewable resources as special cases and can be used to model typical labor regulations and timetabling constraints (Álvarez-Valdés *et al.* 2015). Although many real world applications can be modeled by project scheduling problems under partially renewable resources, only few authors investigated this type of resource so far. Böttcher *et al.* (1999) introduced RCPSP/π and presented an enumeration scheme and an ILP formulation as well as approximation methods. Schirmer (2000) presented examples of conditions that can be modeled by partially renewable resources and heuristic solution methods. Heuristic procedures for RCPSP/π were also developed by Álvarez-Valdés *et al.* (2006, 2008, 2015). In addition, Álvarez-Valdés *et al.* presented some preprocessing steps to identify trivial problem instances and reduce the problem size, respectively.

In recent years, CP techniques were more and more used to solve project scheduling problems and provide very good results, e.g., for RCPSP. In the following, we investigate if this success of CP can be identified also for RCPSP/π.

In Section 2 the RCPSP/π is described formally. Section 3 presents the ILP formulation from Böttcher *et al.* (1999) and an alternative approach to model the resource constraints. The idea for this alternative modeling approach is used in Section 4 to introduce a CP formulation, which can be solved using classical CP solvers as well as LCG solvers. Finally, in Section 5 a performance study on instances from literature and conclusions are given.

2 Problem description

The project in question is given by an activity-on-node network $N = (V, A)$ where the set of nodes V corresponds to n activities $1, \ldots, n$ that have to be carried out without interruption, as well as to two fictitious activities 0 (source of N) and $n + 1$ (sink of N) that represent the beginning and the completion of the project, respectively. A is the set of arcs in N and each arc $\langle i, j \rangle \in A$ represents a precedence relation between activities i and j, i.e., activity j cannot be started before activity i is completed. With $p_i \in \mathbb{N}_0$ we denote

the processing time of activity $i \in V$, where $p_0 = p_{n+1} := 0$, and $S_i \in \mathbb{N}_0$ is the start time of i. We assume that every project starts at time 0, i.e., $S_0 := 0$ and therefore S_{n+1} equals the project duration. The set of partially renewable resources needed for carrying out the project activities is given by $\mathcal{R}$. For each $k \in \mathcal{R}$ a subset of periods of the planning horizon Π_k and a resource capacity R_k is given. Activity $i \in V$ consumes r_{ik} units of resource k in each time period out of set $\{S_i, S_i+1, \ldots, S_i+p_i-1\} \cap \Pi_k$. A vector $S = (S_0, S_1, \ldots, S_{n+1})$ of start times for all activities $i \in V$ is termed a schedule. A schedule is said to be feasible if it satisfies all temporal and resource constraints. Given some schedule S and point in time t the active set $\mathcal{A}(S,t) := \{i \in V \mid S_i \leq t < S_i + p_i\}$ contains all activities that are started at a point in time less or equal than t and that are not finished until time t. Then, $r_k(S,t) := \sum_{i \in \mathcal{A}(S,t)} r_{ik}$ is the total amount of resource $k \in \mathcal{R}$ required to carry out those activities in progress at time $t \in \Pi_k$. A mathematical formulation for RCPSP/π can be given by

$$\begin{aligned}
\text{Minimize} \quad & S_{n+1} && && (1)\\
\text{subject to} \quad & S_j \geq S_i + p_i && \langle i,j \rangle \in A && (2)\\
& \sum\nolimits_{t \in \Pi_k} r_k(S,t) \leq R_k && k \in \mathcal{R} && (3)\\
& S_i \in W_i && i \in V. && (4)
\end{aligned}$$

The objective (1) is to minimize the project duration which is realized by minimizing the start time of the project end. Constraints (2) ensure the given precedence relations and inequalities (3) guarantee that the total amount of required units of resource $k \in \mathcal{R}$ within periods Π_k does not exceed the resource capacity R_k. Constraints (4) restrict the domain for the start time S_i of every activity $i \in V$ to $W_i := \{ES_i, \ldots, LS_i\}$, with ES_i and LS_i being the earliest and latest time feasible start time of i, respectively. ES_i and LS_i can be calculated with a label-correcting algorithm.

3 ILP formulations for RCPSP/π

The ILP formulation for RCPSP/π from Böttcher *et al.* (1999) makes use of time-indexed binary decision variables x_{it} that equal 1 if activity $i \in V$ starts at time $t \in W_i$ and 0 otherwise. The formulation is given by

$$\begin{aligned}
\text{Minimize} \quad & \sum\nolimits_{t \in W_{n+1}} t \cdot x_{n+1,t} && && (5)\\
\text{subject to} \quad & \sum\nolimits_{t \in W_i} x_{it} = 1 && i \in V && (6)\\
& \sum\nolimits_{t \in W_j} t \cdot x_{jt} \geq \sum\nolimits_{t \in W_i} t \cdot x_{it} + p_i && \langle i,j \rangle \in A && (7)\\
& \sum\nolimits_{i \in V} r_{ik} \sum\nolimits_{t \in \Pi_k} \sum\nolimits_{\tau \in Q_{it} \cap W_i} x_{i\tau} \leq R_k && k \in \mathcal{R} && (8)\\
& x_{it} \in \{0,1\} && i \in V, t \in W_i. && (9)
\end{aligned}$$

Since $S_i = \sum_{t \in W_i} t \cdot x_{it}$ holds, constraint (5) corresponds to (1) and constraint set (7) corresponds to (2). Equations (6) ensure that each activity $i \in V$ is started at exactly one point in time out of set W_i. With $Q_{it} := \{t - p_i + 1, \ldots, t\}$ being the set of possible start times for that activity i is in the active set at time t, the left hand side of constraint set (8) equals the total requirement on resource $k \in \mathcal{R}$ within time periods Π_k and therefore the resource capacity R_k is not exceeded.

Within their branch-and-bound algorithm, Böttcher *et al.* (1999) make use of the consumption $sc_{ikt} := |\{t, t+1, \ldots, t+p_i-1\} \cap \Pi_k| \cdot r_{ik}$ of activity $i \in V$ on resource $k \in \mathcal{R}$

when started in period $t \in W_i$. The resource restrictions (8) can thus be rewritten as

$$\sum_{i \in V} \sum_{t \in W_i} x_{it} \cdot sc_{ikt} \leq R_k \qquad k \in \mathcal{R}. \tag{10}$$

In the next section we show how values sc_{ikt} can be used to give a CP formulation for RCPSP/π.

4 A CP formulation for RCPSP/π

A CP formulation is given by a set of decision variables with corresponding domains and constraints that link variables to some other variables. Within CP solvers constraint propagation and search algorithms are strictly separated. Constraint propagation can be understood as the process of generating additional constraints from existing constraints and therefore reducing the domains of the decision variables. Search algorithms are used to systematically explore the solution space (Baptiste *et al.* 2001).

A CP formulation for RCPSP/π that can be solved using IBM CP Optimizer as well as the LCG solver CHUFFED can be given by

$$\begin{aligned}
\text{Minimize} \quad & S_{n+1} && && (11)\\
\text{subject to} \quad & S_j \geq S_i + p_i && \langle i,j \rangle \in A && (12)\\
& \texttt{element}(S_i, \boldsymbol{sc}_{ik}, sc'_{ik}) && i \in V, k \in \mathcal{R} && (13)\\
& \sum_{i \in V} sc'_{ik} \leq R_k && k \in \mathcal{R} && (14)\\
& S_i \in W_i && i \in V. && (15)
\end{aligned}$$

To ensure that decision variable sc'_{ik} takes the correct value sc_{ikS_i} `element` constraints are used in (13), where $\boldsymbol{sc}_{ik}$ is an array containing for all $t \in W_i$ the corresponding sc_{ikt} value.

Within the constraint propagation based solver IBM CP Optimizer we use interval variables (*IloIntervalVar*) to model the project activities and end-to-start constraints (*IloEndBeforeStart*) to model the precedence relations (12).

LCG is a hybrid CP approach that combines features from SAT solving and finite domain propagation (Ohrimenko *et al.* 2009). It is the state of the art solution method for RCPSP, RCPSP/max, and RCPSP/max-cal (cf. Schutt *et al.* 2011, 2013; Kreter *et al.* 2015).

5 Performance study and conclusions

To test the introduced formulations we used the test set from Schirmer (2000) and the extension of that test set from Álvarez-Valdés *et al.* (2006). We excluded trivial instances from our study, as Álvarez-Valdés *et al.* (2006) did. An instance is called trivial, if the schedule where every activity is started at its earliest time feasible start time is feasible. In contrast to Álvarez-Valdés *et al.* (2006), we did not perform their whole preprocessing procedure, because preliminary tests showed that this is not beneficial for the regarded solution approaches when considering the overall run time.

All tests were conducted on an Intel Core i7-2760QM with a frequency of 2.4GHz and 8GB RAM. The following table summarizes the results for the ILP formulation of Böttcher *et al.* (1999) (ILP_B), the ILP formulation with alternative resource constraints (10) (ILP_SC), both solved by IBM CPLEX, and the CP-formulation solved by IBM CP Optimizer (CPO) and the LCG solver CHUFFED (LCG). The first column in Table 1 states the number of activities (n) and the second column states the number of remaining non-trivial instances. For every formulation ILP_B, ILP_SC, CPO, and LCG the left-hand

column states the average CPU time in seconds (∅CT) and the right-hand column states the number of instances for that the optimality was proven within the run time limit of 10 minutes (#SOL). For instances that could not be solved to optimality within the time limit, 10 minutes are taken into account when computing ∅CT. The instances with 10 and 20 activities were solved by all solution approaches in less than one second of average run time, so we did not include them in the table.

Table 1. Computation times and percentage of instances solved

n	#instances	ILP_B		ILP_{SC}		CPO		LCG	
		∅CT	#SOL	∅CT	#SOL	∅CT	#SOL	∅CT	#SOL
30	453	2.80	453	2.78	453	1.22	453	0.66	453
40	386	11.46	385	11.38	385	9.19	384	7.26	384
60	346	61.61	328	61.41	328	56.17	320	49.91	321

Both ILP formulations find more optimal solutions than the CP approaches for instances with 40 and 60 activities and are significantly faster for some of the test instances. In contrast, the average run times of CPO and LCG are smaller than those obtained by ILP_B and ILP_{SC}. These results show that none of the combinations of formulation and solver is preferable to all instances. Combining the results of all combinations we were able to close all open test instances with up to 40 activities. In the future we plan to investigate structural properties of the instances in order to find out which instances should be solved by which method. In addition, disaggregated precedence constraints can be used in the ILP formulations to improve the LP-relaxation.

Acknowledgements

We would like to thank Ramón Álvarez-Valdés for providing us with the test instances.

References

Álvarez-Valdés R., E. Crespo, J.M. Tamarit and F. Villa, 2006, "GRASP and path relinking for project scheduling under partially renewable resources", *European Journal of Operational Research*, Vol. 189, pp. 1153-1170.

Álvarez-Valdés R., E. Crespo, J.M. Tamarit and F. Villa, 2008, "A scatter search algorithm for project scheduling under partially renewable resources", *Journal of Heuristics*, Vol. 12, pp. 95-113.

Álvarez-Valdés R., J.M. Tamarit and F. Villa, 2015, "Partially Renewable Resources". In: Schwindt C. and J. Zimmermann (Eds.), Handbook on Project Management and Scheduling, Vol. 1, Springer, Cham et alibi, pp. 203-227.

Baptiste P., C. Le Pape and W. Nuijten, 2001, "Constraint-Based Scheduling", Kluwer Academic Publishers, Norwell, MA.

Böttcher J., A. Drexl, R. Kolisch and F. Salewski, 1999, "Project scheduling under partially renewable resource constraints", *Management Science*, Vol. 45, pp. 544-559.

Kreter S., A. Schutt and P.J. Stuckey, 2015, "Modeling and solving project scheduling with calendars". In: Pesant, G. (Ed.), Principles and Practice of Constraint Programming – CP 2015, Vol. 9255 of Lecture Notes in Computer Science, Springer, Cham et alibi, pp. 262-278.

Ohrimenko O., P.J. Stuckey and M. Codish, 2009, "Propagation via lazy clause generation", *Constraints*, Vol. 14, pp. 357-391.

Schirmer A., 2000, "Project scheduling with scarce resources", Dr. Kovac, Hamburg.

Schutt A., T. Feydy, P.J. Stuckey and M.G. Wallace, 2011, "Explaining the cumulative propagator", *Constraints*, Vol. 16, pp. 250-282.

Schutt A., T. Feydy, P.J. Stuckey and M.G. Wallace, 2013, "Solving RCPSP/max by lazy clause generation", *Journal of Scheduling*, Vol. 16, pp. 273-289.