An Efficient SMT Approach to Solve MRCPSP/max Instances with Tight Constraints on Resources

Miquel Bofill, Jordi Coll(✉), Josep Suy, and Mateu Villaret

University of Girona, Girona, Spain
{miquel.bofill,jordi.coll,josep.suy,mateu.villaret}@imae.udg.edu

Abstract. The Multi-Mode Resource-Constrained Project Scheduling Problem with Minimum and Maximum Time Lags (MRCPSP/max) is a generalization of the well known Resource-Constrained Project Scheduling Problem. Recently, it has been shown that the benchmark datasets typically used in the literature can be easily solved by relaxing some resource constraints, which in many cases are dummy. In this work we propose new datasets with tighter resource limitations. We tackle them with an SMT encoding, where resource constraints are expressed as specialized pseudo-Boolean constraints and then translated into SAT. We provide empirical evidence that this approach is state-of-the-art for instances highly constrained by resources.

1 Introduction

The Resource-Constrained Project Scheduling Problem (RCPSP) is the problem of finding the start times of a set of non-preemptive activities while respecting some constraints, namely precedence relations between activities and correct usage of shared resources with limited capacity. There exist many extensions of this problem [5], e.g., involving multiple execution modes per activity (MRCPSP), or generalized precedence relations (RCPSP/max). The exact methods with the best known performance in such scheduling problems are based in Lazy Clause Generation [13,16], Failure-Directed Search (FDS) [17] and SAT Modulo Theories (SMT) [1,3,4].

In this work we address the Multi-mode Resource-Constrained Project Scheduling Problem with Minimal and Maximal Time Lags (MRCPSP/max),[1] which combines MRCPSP and RCPSP/max. The goal is to determine a start time and an execution mode for each activity, in order to obtain a schedule which satisfies all the resource and generalized precedence constraints, and which has minimum makespan (i.e., the total duration of the whole set of activities). Up to our knowledge, the best exact approaches for this problem are based in Constraint Integer Programming with cumulative constraint handlers [12] and FDS [17].

[1] This problem is denoted $MPS|temp|C_{max}$ in [5] and $m,1|gpr|C_{max}$ in [8]. It is also known as the Multi-mode RCPSP with Generalized Precedence Relations.

DOI: 10.1007/978-3-319-66158-2_5

Most of the recent works on MRCPSP/max have been evaluated using the benchmark datasets created in [14]. In [17] it was observed that, in those instances, the resource constraints are not the hardest component. In this work, we further analyse the reason why the resource constraints seem not to play an important role in those instances, concluding that such constraints are trivially satisfied in many cases. Therefore, we have crafted new instance sets, where resource constraints take a more important role. Moreover, we present an SMT-based system to solve the MRCPSP/max. We use Integer Difference Logic (IDL) to deal with generalized precedences, and recently introduced specialized pseudo-Boolean (PB) constraints [4] to deal with resource constraints. Such PB constraints allow compact encodings into SAT, and have been shown to improve performance in the MRCPSP. We provide experiments showing that our SMT approach performs better than other exact approaches on instances with tight constraints on resources.

2 The Multi-mode RCPSP with Minimum and Maximum Time Lags (MRCPSP/max)

The MRCPSP/max is defined by a tuple (V, M, p, E, g, R, B, b) where:

- $V = \{A_0, A_1, \ldots, A_n, A_{n+1}\}$ is a set of non-preemptive activities. A_0 and A_{n+1} are dummy activities representing the start and the end of the schedule, respectively. The set of non-dummy activities is defined as $A = \{A_1, \ldots, A_n\}$.
- $M \in \mathbb{N}^{n+2}$ is a vector of naturals, being M_i the number of modes in which activity i can be executed. $M_0 = M_{n+1} = 1$ and $M_i \geq 1, \forall A_i \in A$.
- p is a vector of vectors of naturals, being $p_{i,o}$ the duration of activity i when is executed in mode o, with $1 \leq o \leq M_i$, $p_{0,1} = p_{n+1,1} = 0$.
- E is a set of pairs of activities which have a time lag defined.
- g is a four dimensional vector of naturals, being $g_{i,j,o,o'}$ the time lag from activity A_i in mode o to activity A_j in mode o'. If $g_{i,j,o,o'} >= 0$, it is a minimum time lag, and means that if A_i is running in mode o and A_j is running in mode o', then A_j must start at least $g_{i,j,o,o'}$ units of time after the start of A_i. If $g_{i,j,o,o'} < 0$, it is a maximum time lag, and means that A_i must start at most $|g_{i,j,o,o'}|$ units of time after the start of A_j.
- $R = \{R_1, \ldots, R_{v-1}, R_v, R_{v+1}, \ldots, R_q\}$ is a set of resources. The first v resources are renewable, and the last $q - v$ resources are non-renewable.
- $B \in \mathbb{N}^q$ is a vector of naturals, being B_k the capacity of resource R_k.
- b is a matrix of naturals corresponding to the resource demands of activities per mode: $b_{i,k,o}$ represents the amount of resource R_k required by activity A_i in mode o, $b_{0,k,1} = 0$ and $b_{n+1,k,1} = 0, \forall k \in \{1, \ldots, q\}$. For renewable resources, this is the required amount per time step, whilst for non-renewable resources, it is the total amount required by the activity during its execution.

A schedule is a vector of naturals $S = (S_0, \ldots, S_{n+1})$ where S_i denotes the start time of activity A_i, $S_0 = 0$, and S_{n+1} is the makespan (completion time of the project). A schedule of modes is a vector of naturals $SM =$

$(SM_0, \ldots, SM_{n+1})$ where SM_i, satisfying $1 \leq SM_i \leq M_i$, denotes the mode of each activity A_i. A solution of the MRCPSP/max problem is a schedule of modes SM and a schedule S of minimal makespan S_{n+1} satisfying the generalized precedence (1), non-renewable (2) and renewable (3) resource constraints:

$$((SM_i = o) \wedge (SM_j = o')) \rightarrow (S_j - S_i \geq g_{i,j,o,o'}) \quad \forall (A_i, A_j) \in E, \forall o \in [1, M_i], \forall o' \in [1, M_j] \tag{1}$$

$$\left(\sum_{A_i \in A} \sum_{o \in [1,M_i]} ite(SM_i = o;\, b_{i,k,o};\, 0) \right) \leq B_k \quad \forall R_k \in \{R_{v+1}, \ldots, R_q\} \tag{2}$$

$$\left(\sum_{A_i \in A} \sum_{o \in [1,M_i]} ite\big((SM_i = o) \wedge (S_i \leq t) \wedge (t < S_i + p_{i,o});\, b_{i,k,o};\, 0\big) \right) \leq B_k \quad \forall R_k \in \{R_1, \ldots, R_v\}, \forall t \in H \tag{3}$$

where $ite(c; e_1; e_2)$ is an *if-then-else* expression denoting e_1 if c is true and e_2 otherwise, $H = \{0, \ldots, T\}$ is the scheduling horizon, and T (the length of the scheduling horizon) is an upper bound (UB) for the makespan.

3 Formulation

We propose a SAT modulo Integer Difference Logic formulation to solve the MRCPSP/max. This is a formulation for the decision version of the problem, i.e., it models the problem of finding a feasible schedule for an MRCPSP/max instance whose makespan is smaller or equal than a given UB. The optimization is achieved by successive calls to an SMT solver, as described in Sect. 4.

In our formulation, we use some precomputed values which serve to bound the execution times of the activities. Given an UB, these values can be recomputed according to different criteria [2]; here we consider time lag constraints between activities as done in [12]. By ES_i we denote the earliest time instant at which A_i can start, by LS_i the latest time instant at which A_i can start, and by LC_i the latest time instant at which A_i can end.

We use a set of integer variables $\{S_0, \ldots, S_{n+1}\}$ to denote the start time of each activity. We represent the schedule of modes with the set of Boolean variables $\{sm_{i,o} \mid 0 \leq i \leq n+1, 1 \leq o \leq M_i\}$, being $sm_{i,o}$ true if and only if activity A_i is executed in mode o. The constraints are the following:

$$S_0 = 0 \tag{4}$$

$$ES_i \leq S_i \leq LS_i \quad \forall A_i \in \{A_1, \ldots, A_{n+1}\} \tag{5}$$

$$(sm_{i,o} \wedge sm_{j,o'}) \rightarrow (S_j - S_i \geq g_{i,j,o,o'}) \quad \forall (A_i, A_j) \in E, \forall o \in [1, M_i], \forall o' \in [1, M_j] \tag{6}$$

$$\bigvee_{\forall o \in [1,M_i]} sm_{i,o} \quad \forall A_i \in V \tag{7}$$

$$\neg sm_{i,o} \vee \neg sm_{i,o'} \quad \forall A_i \in A, \forall o, o' \in [1, M_i], o < o' \tag{8}$$

where (5) sets the earliest and latest start time of each activity, (6) encodes the generalized precedences and (7) and (8) ensure that each activity runs in exactly one mode—(8) is an at-most-one (AMO) constraint. The formulation of the constraints over resources is a time-indexed formulation [11], where we introduce the set of Boolean variables $x_{i,t,o}$, which are true if and only if activity A_i is being executed in mode o at time t in the schedule:

$$x_{i,t,o} \leftrightarrow (S_i \leq t < S_i + p_{i,o} \wedge sm_{i,o}) \quad \forall A_i \in A, \forall t \in [ES_i, LC_i), \forall o \in [1, M_i] \quad (9)$$

We can express resource constraints as pseudo-Boolean (PB) constraints. A PB constraint has the form $q_1 \cdot x_1 + \cdots + q_n \cdot x_n \# K$, where q_i and K are integer constants, x_i are 0/1 (false/true) variables, and $\# \in \{<, \leq, =, \geq, >\}$ [7]. A well-known approach to encode PB constraints is to represent them as Binary Decision Diagrams (BDDs), and then encode such BDDs into SAT [7]. This method was successfully applied to encode resource constraints for the MRCPSP in [3]. Recently, in [4] it was proposed a way to compactly encode PB constrains into SAT when their variables can be organized in groups that have AMO constraints already enforced. These PB constraints, called AMO-PB constraints, are built up from AMO-products.

Definition 1 (AMO-product). *We refer to an integer linear expression $q_1 \cdot x_1 + \cdots + q_m \cdot x_m$ over 0/1 variables $x_1, \ldots, x_m$, subject to the fact that at most one x_i is true, as an* AMO-product*. We conveniently express AMO-products as QX, where $Q = \langle q_1, \ldots, q_m \rangle$ and $X = \langle x_1, \cdots, x_m \rangle$.*

Definition 2 (AMO-PB). *We refer to an expression of the form $Q_1X_1 + \cdots + Q_nX_n \leq K$, where Q_iX_i are AMO-products and K is an integer constant, as a* PB constraint with AMO relations (AMO-PB)*.*

Notice that an AMO-PB can be seen as a partial function, whose value is undefined if the AMO relation does not hold for some X_i.

The key idea in the use of AMO-PBs is that, if the AMO constraints over the variables of each AMO-product are already enforced, then the SAT encoding of the AMO-PB does not need to forbid the inconsistent assignments which do not satisfy the AMO constraints. In some formulations the needed AMO constraints are implicitly enforced, and hence there is no need to add additional clauses to the encoding. In our formulation of the MRCPCP/max, Constraints (8) explicitly enforce that at most one of all the variables $sm_{i,o}$ can be true for a particular activity A_i. Similarly, at most one of all the variables $x_{i,t,o}$ can be true for a particular activity A_i and time t, i.e., an activity can be running in at most one execution mode o at a time. Note, however, that the AMO relation of variables $x_{i,t,o}$ for an activity A_i and time t follows from the conjunction of Constraints (8) and (9). Interestingly, the generalized precedence relations (6) can introduce further implicit AMO relations between variables $x_{i,t,o}$. Let us consider a case in which $(A_i, A_j) \in E$, and it holds that $g_{i,j,o,o'} \geq p_{i,o}$, for all execution modes o and o'. In such cases we will say that A_i and A_j have an *end-start* precedence, meaning that A_j will always start after A_i has ended, so they will never be running at a same time. Therefore, in this case $x_{i,t,o}$ and $x_{j,t,o'}$ cannot be both true, because an AMO constraint is implicitly enforced by Constraints (6) and (9). We can take profit of all these explicit AMO constraints over $sm_{i,o}$, and the implicit AMO relations over $x_{i,t,o}$, to express the resource constraints as AMO-PBs.

Similarly to what is done in [4], we precompute a set $\mathcal{P}(t) = \{P_1, \ldots, P_m\}$ for each time t, where all P_j in $\mathcal{P}(t)$ are disjoint sets of activities, and $P_1 \cup \cdots \cup P_m$ contains all the activities A_i which can be running at time t according to their ES_i and LC_i. Moreover, all the activities in a set P_j are pairwise mutually exclusive due to end-start precedences. Hence, there is an AMO relation enforced over the variables $x_{i,t,o}$ of all activities A_i in each P_j. Then, the constraints over renewable resources can be formulated as AMO-PBs, where the j-th AMO-product contains the variables $x_{i,t,o}$ for all $A_i \in P_j$, $o \in [1, M_i]$:

$$\sum_{P_j \in \mathcal{P}(t)} Q(P_j, k) \cdot X(P_j, k, t) \leq B_k \qquad \forall R_k \in \{R_1, \ldots, R_v\}, \forall t \in [0, H] \quad (10)$$

where $$Q(P_j, k) = \langle b_{i,k,o} \mid A_i \in P_j \wedge o \in [1, M_i] \wedge b_{i,k,o} > 0 \rangle$$

$$X(P_j, k, t) = \langle x_{i,t,o} \mid A_i \in P_j \wedge o \in [1, M_i] \wedge b_{i,k,o} > 0 \rangle$$

The non-renewable resource constraints 2 can also be represented using AMO-PB constraints. In this case, the i-th AMO-product will contain all the variables $sm_{i,o}$ for all $o \in [1, M_i]$:

$$\sum_{A_i \in A} Q(A_i, k) \cdot X(A_i, k) \leq B_k \qquad \forall R_k \in \{R_{v+1}, \ldots, R_q\} \quad (11)$$

where $$Q(A_i, k) = \langle b_{i,k,o} \mid o \in [1, M_i] \rangle \qquad X(A_i, k) = \langle sm_{i,o} \mid o \in [1, M_i] \rangle$$

AMO-PB constraints (10) and (11) are encoded into SAT as described in [4].

4 Optimization Procedure

The formulation in Sect. 3 models whether it is possible to find a schedule whose makespan is not greater than a given UB. Some SMT solvers have built-in optimization mechanisms which let to specify an objective function [10,15]. We use the Yices SMT solver [6], because we have experienced that it performs very well in scheduling problems. However, this solver does not provide optimization. In order to get the optimal solution, we have implemented an optimization procedure that can use any off-the-shelf SMT solver as an oracle. Note that our formulation requires an UB in order to specify some constraints. An UB which is commonly used in scheduling problems is *trivialUB*, which is the makespan resulting from scheduling all the activities in a way such that only one runs at a time [2]. Since this UB tends to be significantly larger than the optimal makespan, we implement an optimization procedure similar to the one in [17]:

1. **Find a LB**. We optimize a relaxed version of the MRCPSP/max which does not include Constraints (10), by following a top-bottom search: starting from

trivialUB, we make successive satisfiability calls to the SMT solver, and after each call we set UB to be the makespan of the previous call minus 1. The procedure ends when the optimality is certified.
2. **Find an UB**. We optimize a single-mode version of the MRCPSP/max, by enforcing the execution modes to be the ones of the optimal solution found in step 1. We perform a bottom-top search, i.e., we try increasing upper bounds, starting from LB, until a model is found, or *trivialUB* is reached.
3. **Solve MRCPSP/max**. This last step is only required when UB > LB. In this case, we follow a top-bottom search starting at UB.

Steps 1 and 2 are intended to find tight bounds on the makespan. We perform these steps because reaching the optimal makespan of the MRCPSP/max decreasingly from *trivialUB* or increasingly from a LB, have shown in preliminary experiments to be far more time consuming than finding bounds by means of relaxed formulations. The main differences with respect to [17] are two: in [17], a MIP formulation in which the renewable resource constraints are relaxed using energetic reasoning (instead of completely ignoring them) is used in step 1; on the other hand, in [17] built-in optimization methods are used in all three steps.

5 New Benchmark Datasets for MRCPSP/max

Most of the recent approaches on solving the MRCPSP/max have been evaluated on the datasets generated in [14] and available in the PSPLib [9]. In particular, there are three datasets with 270 problem instances each one, namely mm30, mm50 and mm100 (instances have 30, 50 and 100 non-dummy activities). The number of execution modes ranges from 3 to 5, and there are 3 renewable and 3 non-renewable resources in each instance. The works of [12,17] have reported the best exact results, considering the mm30, mm50 and mm100 datasets. The former presented a handler for an extension of the cumulative constraint for the MRCPSP/max integrated into the SCIP optimization framework. The latter proposed a Failure-Directed Search Constraint Programming approach which shown to perform very well in different scheduling problems. Also, in [17] it was pointed out that, in the previously mentioned instances, resource constraints are not the hardest part of the problem. They used the relaxations on resource constraints mentioned in Sect. 4 to find a LB and an UB and it turned out that, in most of the cases, the LB and the UB were equal, and therefore an optimal solution was found without the need of encoding the whole original problem.

We have studied why the resources play such a minor role in those datasets. For 1432 out of the total 2430 non-renewable resources, it is trivially true that the demands do not exceed their capacity, i.e., any assignment of modes satisfies Constraint (2) for these resources. We have also checked if the relaxed optimal solutions obtained in step 1 of our optimization procedure satisfied the renewable resource constraints, which weren't enforced in this relaxation. We have observed that the relaxed optimal solution found for 593 out of the 810 instances satisfied the renewable resource constraints, although they were not encoded, and hence were also optimal solutions of the original problem. This number may be larger because some instances timed out without having found the optimal solution of the relaxation, and therefore have not been counted.

Table 1. Resource capacities of the new datasets. The capacity of each resource of each instance have been generated uniformly and independently at random in the interval indicated in the corresponding cell of the table.

Set name	mm30_1	mm30_2	mm50_1	mm50_2	mm100_1	mm100_2
Renewable capacity	[30,39]	[20,29]	[30,39]	[20,29]	[30,39]	[20,29]
Non-renewable capacity	[135,164]	[135,164]	[235,264]	[235,264]	[485,514]	[485,514]

These characteristics, in addition to the fact that all the mentioned instances have been closed, suggest us that there is a need for new and more challenging datasets, in particular regarding the hardness of resource constraints. The reason why most renewable and non-renewable resources barely constrain the instances is because the capacities are far large enough to supply the demands of the activities in a large amount of the possible combinations of mode assignments. For this reason, we propose to use as a basis the same instances, but shrinking the capacities of the resources to amounts which make them non-dummy. For the case of the renewable resources, we have conducted some preliminary experiments to see approximately which capacity is needed to make the optimal makespan of an instance increase. We have observed that, given the original demand values and precedence network topologies, for capacities smaller than 30, rarely any instance has an optimal makespan equal to the optimal of the MRCPSP/max without resource constraints. Regarding the non-renewable resources, the original dataset was created with demands ranging from 1 to 10. Considering that all the activities require the intermediate amount of 5 units for each non-renewable resource, it would be needed a capacity of $5n$ to supply the demands, being n the number of activities of the instance. Considering these facts, we have generated two new versions of each one of the mm30, mm50 and mm100 datasets, namely mm{30,50,100}_1 and mm{30,50,100}_2. They are the result of replacing the resource capacities as stated in Table 1. The new mm{30,50,100}_2 datasets are intended to be very constrained by resources. This is indeed the case, since we have not been able to find any relaxed solution satisfying the renewable resource constraints, and no non-renewable constraint is dummy. Sets mm{30,50,100}_1 are a bit softer regarding renewable resources.

6 Results and Conclusions

The experiments have been run on a 8 GB Intel® Xeon® E3-1220v2 machine at 3.10 GHz. We have run our solver using Yices 2.4.2 [6] as the core SMT solver, and compared the performance of our system with [17] (FDS) and [12] (SCIP). We have run all three solvers in the same machine on both the old and

Table 2. Results for each solver (rows) and each dataset (columns), with a timeout of 600 s; *avg* denotes the average running time, in seconds, required to optimally solve the instances (computed on the instances which did not time out); *to* denotes the number of instances that timed out before reaching the optimum.

		30	50	100	30_1	50_1	100_1	30_2	50_2	100_2
FDS	avg	0.48	1.30	15.19	27.80	125.12	36.00	72.91	185.41	-
	to	0	0	3	1	113	267	24	216	270
SCIP	avg	14.49	26.74	134.03	88.97	163.79	-	178.33	281.23	-
	to	1	24	90	49	206	270	155	257	270
SMT	avg	0.87	17.14	67.97	16.39	120.38	-	54.36	224.47	-
	to	0	0	65	1	108	270	12	192	270

the new datasets. The results[2] are contained in Table 2. FDS is doubtlessly the best solver for the original datasets, with only 3 timeouts in the hardest dataset (mm100) and average runtimes one order of magnitude lower than the other approaches. This is because its MIP relaxation works very well with generalized precedence relations. It must be noted that SCIP does not start by solving a relaxed MRCPSP/max with respect to resources, which penalizes this approach in these datasets. On the other hand we can see that, in the new datasets, which are more constrained by resources, SMT is able to solve more instances than the other approaches in all cases except for the mm100_1 dataset. We remark that 2 out of the 3 instances that FDS solves in this dataset are optimally solved already in the relaxation solving steps. SMT is the best approach in sets mm{30,50}_2, which have the strongest resource constraints.

In conclusion, thanks to identifying some weaknesses of existing instances, we are able to provide new and challenging datasets for MRCPSP/max. This is especially noticeable in mm100 sets which, with the exception of 3 instances for FDS, could not be solved in less than 600 s using the state-of-the-art exact methods for MRCPSP/max. We have provided an SMT formulation showing to be more efficient than other state-of-the-art approaches for heavily resource constrained instances. Interestingly, our approach takes an off-the-shelf SMT solver and, instead of using specialized propagators, it uses recent specialized encodings of PB constraints into SAT to deal with resource constraints. The resulting encoding uses only the IDL theory, to deal with generalized precedences. Left as future work are the incorporation of better relaxations on resource equations, as the ones in [17], for a better LB identification. Also other optimization approaches could be considered, as well as the use of a full SAT encoding.

Acknowledgments. Work supported by grants TIN2015-66293-R (MINECO/FEDER, UE), MPCUdG2016/055 (UdG), and *Ayudas para Contratos Predoctorales 2016* (grant number BES-2016-076867, funded by MINECO and co-funded by FSE). We thank the authors of [12,17] for sharing with us their solvers.

[2] The solver used in the experiments, detailed results and the new instances are available at http://imae.udg.edu/recerca/LAP/.

References

1. Ansótegui, C., Bofill, M., Palahí, M., Suy, J., Villaret, M.: Satisfiability modulo theories: an efficient approach for the resource-constrained project scheduling problem. In: Proceedings of the Ninth Symposium on Abstraction, Reformulation, and Approximation (SARA), pp. 2–9. AAAI (2011)
2. Artigues, C., Demassey, S., Neron, E.: Resource-Constrained Project Scheduling: Models, Algorithms, Extensions and Applications. Wiley, Hoboken (2013)
3. Bofill, M., Coll, J., Suy, J., Villaret, M.: Solving the multi-mode resource-constrained project scheduling problem with SMT. In: 28th International Conference on Tools with Artificial Intelligence (ICTAI), pp. 239–246. IEEE (2016)
4. Bofill, M., Coll, J., Suy, J., Villaret, M.: Compact MDDs for pseudo-Boolean constraints with at-most-one relations in resource-constrained scheduling problems. In: International Joint Conference on Artificial Intelligence (IJCAI) (2017, to appear)
5. Brucker, P., Drexl, A., Mhring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: notation, classification, models, and methods. Eur. J. Oper. Res. **112**(1), 3–41 (1999)
6. Dutertre, B., de Moura, L.: The yices SMT solver. Technical report, Computer Science Laboratory, SRI International (2006). http://yices.csl.sri.com
7. Eén, N., Sorensson, N.: Translating pseudo-Boolean constraints into SAT. J. Satisfiability Boolean Model. Comput. **2**, 1–26 (2006)
8. Herroelen, W., Demeulemeester, E., Reyck, B.: A classification scheme for project scheduling. In: Weglarz, J. (ed.) Project Scheduling. International Series in Operations Research & Management Science, vol. 14, pp. 1–26. Springer, New York (1999). doi:10.1007/978-1-4615-5533-9_1
9. Kolisch, R., Sprecher, A.: PSPLIB - a project scheduling problem library. Eur. J. Oper. Res. **96**(1), 205–216 (1997)
10. de Moura, L., Bjørner, N.: Z3: an efficient SMT solver. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 337–340. Springer, Heidelberg (2008). doi:10.1007/978-3-540-78800-3_24
11. Pritsker, A.A.B., Waiters, L.J., Wolfe, P.M.: Multiproject scheduling with limited resources: a zero-one programming approach. Manag. Sci. **16**, 93–108 (1969)
12. Schnell, A., Hartl, R.F.: On the efficient modeling and solution of the multi-mode resource-constrained project scheduling problem with generalized precedence relations. OR Spectr. **38**(2), 283–303 (2016)
13. Schutt, A., Feydy, T., Stuckey, P.J., Wallace, M.G.: Solving the resource constrained project scheduling problem with generalized precedences by lazy clause generation. CoRR abs/1009.0347 (2010). http://arxiv.org/abs/1009.0347
14. Schwindt, C.: Generation of resource constrained project scheduling problems subject to temporal constraints. Inst. für Wirtschaftstheorie und Operations-Research (1998)
15. Sebastiani, R., Tomasi, S.: Optimization in SMT with $\mathcal{LA}(\mathbb{Q})$ cost functions. In: Gramlich, B., Miller, D., Sattler, U. (eds.) IJCAR 2012. LNCS, vol. 7364, pp. 484–498. Springer, Heidelberg (2012). doi:10.1007/978-3-642-31365-3_38
16. Szeredi, R., Schutt, A.: Modelling and solving multi-mode resource-constrained project scheduling. In: Rueher, M. (ed.) CP 2016. LNCS, vol. 9892, pp. 483–492. Springer, Cham (2016). doi:10.1007/978-3-319-44953-1_31
17. Vilím, P., Laborie, P., Shaw, P.: Failure-directed search for constraint-based scheduling. In: Michel, L. (ed.) CPAIOR 2015. LNCS, vol. 9075, pp. 437–453. Springer, Cham (2015). doi:10.1007/978-3-319-18008-3_30