Schnell, Alexander; Hartl, Richard F.

Article

On the generalization of constraint programming and boolean satisfiability solving techniques to schedule a resource-constrained project consisting of multi-mode jobs

Operations Research Perspectives

Provided in Cooperation with:
Elsevier

Suggested Citation: Schnell, Alexander; Hartl, Richard F. (2017) : On the generalization of constraint programming and boolean satisfiability solving techniques to schedule a resource-constrained project consisting of multi-mode jobs, Operations Research Perspectives, ISSN 2214-7160, Elsevier, Amsterdam, Vol. 4, pp. 1-11, https://doi.org/10.1016/j.orp.2017.01.002

This Version is available at:
https://hdl.handle.net/10419/178271

Operations Research Perspectives 4 (2017) 1–11

Contents lists available at ScienceDirect

Operations Research Perspectives

journal homepage: www.elsevier.com/locate/orp

On the generalization of constraint programming and boolean satisfiability solving techniques to schedule a resource-constrained project consisting of multi-mode jobs

Alexander Schnell, Richard F. Hartl*

University of Vienna, Department of Business Administration, Chair of Production and Operations Management, Oskar-Morgenstern-Platz 1, 1090 Vienna, Austria

ARTICLE INFO

Article history:
Received 15 May 2016
Revised 16 January 2017
Accepted 16 January 2017
Available online 23 January 2017

Keywords:
Multi-mode resource-constrained project scheduling
Constraint programming
SAT solving
SCIP
Lazy clause generation
Exact algorithm

ABSTRACT

In our paper, we analyze new exact approaches for the multi-mode resource-constrained project scheduling (MRCPSP) problem with the aim of makespan minimization. For the single-mode RCPSP (SRCPSP) recent exact algorithms combine a Branch and Bound algorithm with principles from Constraint Programming (CP) and Boolean Satisfiability Solving (SAT). We extend the above principles for the solution of MRCPSP instances. This generalization is on the one hand achieved on the modeling level. We propose three CP-based formulations of the MRCPSP for the G12 CP platform and the optimization framework SCIP which both provide solution techniques combining CP and SAT principles. For one of the latter we implemented a new global constraint for SCIP, which generalizes the domain propagation and explanation generation principles for renewable resources in the context of multi-mode jobs. Our constraint applies the above principles in a more general way than the existing global constraint in SCIP. We compare our approaches with the state-of-the-art exact algorithm from the literature on MRCPSP instances with 20 and 30 jobs. Our computational experiments show that we can outperform the latter approach on these instances. Furthermore, we are the first to close (find the optimal solution and prove its optimality for) 628 open instances with 50 and 100 jobs from the literature. In addition, we improve the best known lower bound of 2815 instances and the best known upper bound of 151 instances.

1. Introduction

The multi-mode resource-constrained project scheduling problem (MRCPSP) is a generalization of the single-mode RCPSP (SRCPSP) where an additional mode-assignment step has to be considered. The aim is to find the best mode-assignment for a number of jobs subject to nonrenewable resource constraints such that the optimal schedule for the resulting SRCPSP (if existing) optimizes a specific objective function.

For the SRCPSP recent exact algorithms combine a Branch and Bound (BaB) algorithm with principles from Constraint Programming (CP) and Boolean Satisfiability Solving (SAT) (see [5], [15] and [30]). The idea of the CP-SAT algorithms is to combine the domain propagation processed through global constraints (Apt [3]) with the Conflict Analysis (CA) techniques of a SAT solver (Marques-Silva and Sakallah [19]). Therefore, the different propagators of the global constraints generate explanations, i.e. clauses consisting of Boolean literals, for their domain updates and the detected inconsistencies. The latter explanations are transfered to a SAT solving mechanism. The SAT mechanism constructs a conflict graph based on the explanations of the domain propagators and can possibly deduce *nogoods* and *backjumps* via CA.

Roughly speaking, nogoods are valid clauses for a SAT model, like e.g. cutting planes in Mixed-Integer Programming (MIP), which possibly prune branches of the BaB-tree. Backjumps are backtracking moves which lead from the actual node a to a preceding node p whereas $d(a) - d(p) > 1$ holds for the depth levels $d(a)$ and $d(p)$ in the BaB tree. Moreover, the branching strategy of the underlying BaB-algorithm uses conflict statistics of the literals forming the explanations. In general, the algorithms branch on the variables and values based on the number of conflicts the respective literals were involved in (Moskewicz et al. [21]). For a more detailed introduction to the principles of CP and SAT solving and the possible

* Corresponding author.
E-mail addresses: alexander.schnell@univie.ac.at (A. Schnell), richard.hartl@univie.ac.at (R.F. Hartl).

http://dx.doi.org/10.1016/j.orp.2017.01.002

combination of the both to one exact solution algorithm, we refer to Schutt et al. [31], Schutt [27] and Achterberg [1,2].

The lazy clause generation approach (LCG), a CP-SAT hybrid introduced by Ohrimenko et al. [23] and extended by Schutt et al. [28], is up-to-date the best exact approach for the SRCPSP with standard precedence relations and the aim of makespan minimization. Furthermore, LCG was also applied to variants of the SRCPSP with more general constraints and with objective functions differing from makespan minimization. Schutt et al. [31] successfully solve the SRCPSP with generalized precedence relations by LCG. They outperform the state-of-the-art exact approaches for this problem and also on average report better results compared to state-of-the-art heuristics. Moreover, Schutt et al. [29] outperform the state-of-the-art exact algorithm for the SRCPSP with discounted cash flows, again by generalizing LCG to this problem. One can conclude, that LCG is a robust approach for variants of the SRCPSP.

The aim of this paper is to provide a generalization of the CP-SAT hybrids for the SRCPSP to the MRCPSP. Exact approaches for the MRCPSP have been summarized and tested by Hartmann and Drexl [12], whereas they conclude that the approach of Sprecher and Drexl [32] is the exact method of choice. The most recent exact algorithm of Zhu et al. [37] outperforms the latter approach. They implemented a Branch-and-Cut procedure with a preprocessing and a heuristic step to generate good upper bounds as an input for their algorithm. A recent survey on heuristic approaches for the MRCPSP and a detailed experimental evaluation is given by Peteghem and Vanhoucke [35]. Their computational experiments show that the scatter search procedure of Peteghem and Vanhoucke [34] produces the best results. In this context, it is also important to mention the approach of Coelho and Vanhoucke [8] as they combine SAT solving techniques with a metaheuristic for the SRCPSP to solve the MRCPSP.

Our contribution is an extension of recent exact approaches combining CP and SAT techniques which are efficient for the SRCPSP to the MRCPSP, more precisely the MRCPSP with standard precedence relations. This extension can be partly achieved on the modeling level. We propose three CP models for the MRCPSP which can be formulated in optimization frameworks that integrate an exact solution approach combining CP, SAT and MIP techniques. Moreover, for one modeling formulation we implemented a new global constraint `cumulativemm` specially tailored to renewable resources in the context of multi-mode jobs. Note that we already successfully generalized and applied recent CP-SAT approaches to the MRCPSP with generalized precedence relations in [26]. The paper at hand can be seen as a predecessor of the latter paper.

In the remainder of the paper, we proceed as follows. In Section 2, we describe the MRCPSP and its computational complexity. Section 3 introduces three problem formulations in optimization frameworks which support the solution by a BaB algorithm integrating CP, SAT and MIP techniques. In Section 4, we describe the principles of our new global constraint `cumulativemm`. Section 5 discusses the results of our computational experiments and draws a comparison to the state-of-the-art exact approach of Zhu et al. [37]. The paper ends with a conclusion derived from the obtained results.

2. Problem description and complexity

The MRCPSP is a generalization of the SRCPSP, where every job $j \in J = \{0, \ldots, n+1\}$ can be processed in different modes $k \in M_j \subseteq \mathbb{N}$. The jobs 0 and $n+1$ are dummy jobs representing the start and the end of the complete project, i.e. in the beginning every job with no predecessor and every job with no successor is connected to the dummy job 0 and $n+1$ in the precedence network, respectively. Moreover, the jobs can not be preempted.

Moreover, a set of nonrenewable (renewable) resources $N(R) \subseteq \mathbb{N}$ with a maximal capacity of $C_r^\nu,\ r \in N$ ($C_r^\rho,\ r \in R$) is given. Every job's integer duration $d_{j,k} \geq 0$, nonrenewable (renewable) resource consumption $c_{j,k,r}^\nu,\ r \in N (c_{j,k,r}^\rho,\ r \in R)$ is dependent on the selected mode $k \in M_j$.

Nonrenewable resources $r \in N$ like e.g. a project budget or energy are available for the complete planning horizon. Once job j is processed in mode k, $C_r^\nu - c_{j,k,r}^\nu$ units of the nonrenewable resource $r \in N$ are still available for the remaining jobs. Moreover, a constant amount C_r^ρ of a renewable resource $r \in R$ like e.g. a number of machines or workers is available at every point in time.

Furthermore, a job $j \in J$ cannot end after a job from its successor set $\mathfrak{S}_j$ has started, i.e. in our paper we only consider standard precedence relations. As objective, we consider makespan minimization.

The solution of the MRCPSP can be divided into two steps. The first step consists of finding a feasible mode-assignment w.r.t. the nonrenewable resource capacities. The knapsack problem is polynomially reducible to the latter problem, i.e. already the mode-assignment step is **NP**-complete for $|N| \geq 2$ (Kolisch and Drexl [17]). The second step consists of finding an optimal schedule for a SRCPSP instance, i.e. of finding a schedule which minimizes the makespan and respects the precedence constraints and the renewable resource capacities for a given mode-assignment.[1] Note, that the SRCPSP with the objective of makespan minimization is strongly **NP**-complete (Blazewicz et al. [6]).

In total, one has to find a feasible mode-assignment at which the minimal makespan of the resulting SRCPSP is not larger than the minimal makespan detected for any other feasible mode-assignment.

As a preprocessing step one can remove redundant nonrenewable resources, inefficient and non-executable modes (see [32] and [11]). Furthermore, lower and upper bounds $lb(s_j)$ and $ub(s_j)$ can be deduced for the starting times s_j by applying forward (backward) recursion [7]. This approach is based on longest path calculations in the precedence network where the arc weights correspond to the minimal mode durations of every job $j \in J$ w.r.t. the remaining modes. For the evaluation of $ub(s_j)$, an upper bound T on the makespan is needed. T can be given by a problem specific heuristic or $T_{\max}$ defined in Section 4.

3. CP-models for the MRCPSP

There are two solution frameworks which provide a solution algorithm consisting of a combination of CP, SAT and MIP techniques.

The first is the Constraint Integer Programming framework SCIP, developed by Achterberg [2] and maintained and extended by members of the Zuse Institute in Berlin. SCIP provides a general BaB algorithm for optimization and allows the user to implement plugins, e.g. special branching strategies, primal heuristics and constraint handlers (i.e. global constraints). Moreover, default plugins exist to use SCIP as a stand-alone CP or MIP solver. Furthermore, when the formulated model only consists of default constraint handlers provided by SCIP, the solution algorithm integrates techniques from CP, SAT Solving and MIP.

The second framework is the G12 Constraint Programming Platform [9] provided by the NICTA research team [22]. The user can formulate a problem in the modeling language Zinc [20] and choose between different solution algorithms. Thereby, LCG can also be chosen for the solution of a model. With the G12 Con-

[1] Note that, it can happen that no feasible schedule for the resulting SRCPSP exists, if mode m has been chosen for job j and $c_{j,m,r}^\rho > C_r^\rho$.

straint Programming Platform [9] the user is not that flexible as with SCIP but he can solve his model with the state-of-the-art exact algorithm for different variants of the SRCPSP.

In the following, we present three CP-models for the MRCPSP in the above optimization frameworks. Section 3.1 contains one model which can be used within the Zinc-modeling language and Section 3.2 describes two models which can be implemented within SCIP. We also implemented a new constraint handler `cumulativemm` for SCIP, which we apply for a model in Section 3.2. The principles of the latter are described in Section 4.

3.1. Formulation for the Zinc-modeling language

For the modeling of the starting time of job i and the mode assignment of job i, we use the integer variables s_i and x_j, respectively. With the latter variables and the notation of Section 2, the MRCPSP can be formulated as follows in the CP-modeling language Zinc:

$$\min \quad s_{n+1} \tag{1}$$

$$s.t. \quad s_i + d_{i,x_i} \le s_j, \quad \forall j \in \mathfrak{S}_i, \forall i \in J \tag{2}$$

$$\sum_{i \in J} c^{\nu}_{i,x_i,r} \le C^{\nu}_r, \quad \forall r \in N \tag{3}$$

$$\texttt{cumulative}(\mathbf{s}, \mathbf{d}, \mathbf{c}^r, C^{\rho}_r), \quad \forall r \in R \tag{4}$$

$$s_j \in \left\{lb(s_j), \ldots, ub(s_j)\right\}, \quad \forall j \in J \tag{5}$$

$$x_j \in M_j, \quad \forall j \in J \tag{6}$$

where

$$\mathbf{s} = [s_j : j \in J] \tag{7}$$

$$\mathbf{d} = [d_{j,x_j} : \ j \in J] \tag{8}$$

$$\mathbf{c}^r = [c^{\rho}_{j,x_j,r} : \ j \in J] \tag{9}$$

The dummy job $n+1$ represents the end of the project. It can only be processed in mode 1 and $d_{n+1,1}$, $c^{\nu}_{n+1,1,r}$, $c^{\rho}_{n+1,1,r} = 0$, $\forall r \in R \cup N$. Moreover, every non-dummy job which has no non-dummy successor is connected to the dummy job $n+1$ in the precedence network. Thus, minimising the makespan is equal to minimising s_{n+1}.

(2) are multi-mode precedence constraints. With (3) and (4), we assure that the available capacities of the nonrenewable and renewable resources are not exceeded. Hereby, in (4) we use the scheduling specific global constraint `cumulative` (Baptiste et al. [4]). To apply this constraint for the MRCPSP, we define the variable vectors $\mathbf{s}, \mathbf{d}$ and $\mathbf{c}^r$ in (7)–(9).

In the above formulation, variables appear in the indices of parameters, like e.g. in d_{i,x_i}. This modeling technique can only be applied if the respective solver supports the global `element`-constraint introduced by Hentenryck and Carillon [33].

In general, the `element`-constraint has the following form:

$$\texttt{element}(y, \mathbf{x}, z) \tag{10}$$

(10) guarantees, that the yth element of the variable (or parameter) vector $\mathbf{x}$ equals the variable z, i.e. $\mathbf{x}_y = z$. Clearly, if $\mathbf{x}$ has n entries, it must hold that $y \le n-1$ if zero is the first index. Propagation algorithms captured by the `element`-constraint can infer domain updates for the variable z in case of domain updates of y or of the entries $\mathbf{x}_i$ of $\mathbf{x}$ and vice versa.

In the case of our model, the terms d_{i,x_i} and $c^{\rho}_{i,x_i,r}$ are internally transformed to new variables d'_i, $c'^{\rho}_{i,r}$ and $c'^{\nu}_{i,r}$ by posting the following constraints:

$$\texttt{element}(x_i, [d_{i,k} : \ k \in M_i], d'_i), \quad \forall i \in J \tag{11}$$

$$\texttt{element}(x_i, [c^{\rho}_{i,k,r} : \ k \in M_i], c'^{\rho}_{i,r}), \quad \forall i \in J, \ \forall r \in R \tag{12}$$

$$\texttt{element}(x_i, [c^{\nu}_{i,k,r} : \ k \in M_i], c'^{\nu}_{i,r}), \quad \forall i \in J, \ \forall r \in N \tag{13}$$

Thus, in our application, after a transformation of the respective solver only the variables d'_i and $c'^{\rho}_{i,r}$ are used in the `cumulative`-constraint.

3.2. Formulations for SCIP

SCIP provides the `optcumulative`-constraint introduced by Heinz et al. [14] to model renewable resource constraints in the context of multi-mode jobs. However, to apply the above constraint for the MRCPSP, we have to introduce integer starting time variables $s_{i,k}$ for every job i and mode k and the binary variables $x_{i,k}$ for the mode assignment of job i. Note, that the mode-assignment is modeled by binary variables as SCIP does not support the `element`(...)-constraint.

With the latter variables and the notation of Section 2, the MRCPSP can be formulated as follows in SCIP with the `optcumulative`-constraint:

$$\min \quad s_{n+1,1} \tag{14}$$

$$s.t. \sum_{k \in M_i} x_{i,k} = 1, \quad \forall i \in J \tag{15}$$

$$s_{i,k} + d_{i,k} \cdot x_{i,k} \le s_{j,l}, \quad \forall j \in \mathfrak{S}_i, \forall i \in J, \ \forall k \in M_i, \ \forall l \in M_j \tag{16}$$

$$\sum_{i \in J} \sum_{k \in M_i} c^{\nu}_{i,k,r} \cdot x_{i,k} \le C^{\nu}_r, \quad \forall r \in N \tag{17}$$

$$\texttt{optcumulative}(\bar{\mathbf{s}}, \bar{\mathbf{x}}, \bar{\mathbf{d}}, \bar{\mathbf{c}}^r, C^{\rho}_r), \quad \forall r \in R \tag{18}$$

$$s_{j,k} \in \left\{lb(s_{j,k}), \ldots, ub(s_{j,k})\right\}, \ \forall j \in J, \ \forall k \in M_j \tag{19}$$

$$x_{j,k} \in \{0,1\}, \quad \forall j \in J, \ \forall k \in M_j \tag{20}$$

where

$$\bar{\mathbf{s}} = \mathbf{s}^0 \circ \cdots \circ \mathbf{s}^{n+1}, \quad \text{where } \mathbf{s}^i_k = s_{i,k}, \ \forall k \in M_i, \ \forall i \in J \tag{21}$$

$$\bar{\mathbf{x}} = \mathbf{x}^0 \circ \cdots \circ \mathbf{x}^{n+1}, \quad \text{where } \mathbf{x}^i_k = x_{i,k}, \ \forall k \in M_i, \ \forall i \in J \tag{22}$$

$$\bar{\mathbf{d}} = \mathbf{d}^0 \circ \cdots \circ \mathbf{d}^{n+1}, \quad \text{where } \mathbf{d}^i_k = d_{i,k}, \ \forall k \in M_i, \ \forall i \in J \tag{23}$$

$$\bar{\mathbf{c}}^r = \mathbf{c}^{0,r} \circ \cdots \circ \mathbf{c}^{n+1,r}, \quad \text{where } \mathbf{c}^{i,r}_k = c^{\rho}_{i,k,r}, \forall k \in M_i, \ \forall i \in J, \ \forall r \in R \tag{24}$$

Again, we minimize the starting time $s_{n+1,1}$ of the dummy job $n+1$, which can only be processed in mode 1. With (15), (16) and (17), we formulate the uniqueness of the mode-assignments, the multi-mode precedence constraints and the nonrenewable resource constraints, respectively. (18) guarantees that the maximal capacities of the renewable resources are not exceeded. To guarantee a correct input for `optcumulative` we have to use the variable vectors $\bar{\mathbf{s}}$ and $\bar{\mathbf{x}}$ and the parameter vectors $\bar{\mathbf{d}}$ and $\bar{\mathbf{c}}^r$ which are given in (21)–(24). In this context, the operator $\circ$ is defined as the concatenation of two vectors, whereas the vector $\mathbf{c} = \mathbf{a} \circ \mathbf{b}$ is obtained by appending the elements of $\mathbf{b}$ coordinate-wise to $\mathbf{a}$.

The above SCIP-formulation has two major disadvantages. Firstly, we have to introduce starting time variables for every job-mode combination $(i, k), i \in J, k \in M_i$.

The second disadvantage has to do with the implementation of the `optcumulative`-constraint [14]. The domain propagation step and the inconsistency check for a variable $s_{i,m}$ in the `optcumulative`-constraint only considers variables $s_{j,k}$, $j \ne i$ for which $x_{j,k} = 1$ in the recent node of the BaB-tree. However, also

variables $s_{j,k}$, $j \neq i$, where the mode-assignment has not been done yet, can be considered for the domain propagation and the inconsistency check of a variable $s_{i,m}$.

To overcome the above disadvantages, we aimed at implementing a new global constraint `cumulativemm` for SCIP to be able to apply a more general form of domain propagation and explanation generation for renewable resources in the context of multi-mode jobs where we only have to introduce starting time variables s_j for every job $j \in J$. The principles of the `cumulativemm`-constraint are outlined in Section 4.

With our new constraint and again binary variables $x_{i,k}$ for the mode-assigment, we formulate the MRCPSP as follows in SCIP:

$$\min \quad s_{n+1} \tag{25}$$

$$s.t. \sum_{k \in M_i} x_{i,k} = 1, \quad \forall i \in J \tag{26}$$

$$s_i + \sum_{k \in M_i} d_{i,k} \cdot x_{i,k} \leq s_j, \quad \forall j \in \mathfrak{S}_i, \forall i \in J \tag{27}$$

$$\sum_{i \in J} \sum_{k \in M_i} c^{\nu}_{i,k,r} \cdot x_{i,k} \leq C^{\nu}_r, \quad \forall r \in N \tag{28}$$

$$\texttt{cumulativemm}(\mathbf{s}, \bar{\mathbf{x}}, \bar{\mathbf{d}}, \bar{\mathbf{c}}^r, C^{\rho}_r), \quad \forall r \in R \tag{29}$$

$$s_j \in \left\{ lb(s_j), \ldots, ub(s_j) \right\}, \quad \forall j \in J, \ \forall k \in M_j \tag{30}$$

$$x_{j,k} \in \{0, 1\}, \quad \forall j \in J, \ \forall k \in M_j \tag{31}$$

4. Principles of the `cumulativemm`-constraint

With our `cumulativemm`-constraint, one can model renewable resource constraints for multi-mode jobs. The main ingredients of the `cumulativemm`-constraint are a feasibility check, constraint propagation and explanation generation (see Sections 4.1 and 4.2) for a certain resource $r \in R$.

Our constraint enforces feasibility w.r.t. the renewable resource $r \in R$:

$$\sum_{j \in J, \ m \in M_j: \ t - d_{j,m} + 1 \leq s_j \leq t \ \wedge \ x_{j,m} = 1} c_{j,m,r} \leq C^{\rho}_r, \quad \forall \, t \in \{1, \cdots, T\}$$

The other constraints in the SCIP-model of the MRCPSP are modeled by SCIP-intern constraints and can therefore be handled by the SCIP-intern solution principles.

The constraint propagation procedure consists of a redundancy check and a domain reduction step. We firstly check the multi-mode data for redundancy in the current node of the BaB tree. In concrete, if we assume the maximal mode duration, the maximal resource consumption and the maximal processing interval for every job $j \in J$ in the current node and the underlying schedule is feasible w.r.t. the renewable resource $r \in R$, we can locally remove our constraint from the solution procedure. This is due to the fact, that in the above case, it cannot be violated anymore in the succeeding branches of the BaB tree.

The domain reduction step is mainly based on the calculation of a minimal problem instance (MPI) [13] in every processed node of the BaB tree, i.e. the transformation of the multi-mode data to a single-mode representative. Therefore, we calculate a minimal processing version $MPV_{j,r} = (\text{domain}(s_j); d^{\min}_j; c^{\min}_{j,r})$ for every job $j \in J$ and renewable resource $r \in R$ as follows:

$$d^{\min}_j = \min_{k \in M_j} \left\{ d_{j,k} : ub(x_{j,k}) > 0 \right\} \tag{32}$$

$$c^{\min}_{j,r} = \min_{k \in M_j} \left\{ c^{\rho}_{j,k,r} : ub(x_{j,k}) > 0 \right\} \tag{33}$$

In (32) and (33), we calculate the minimal duration and resource consumption of resource $r \in R$ w.r.t. the modes which have not been excluded ($ub(x_{j,k} > 0)$ in the recent node of the BaB tree. With the MPI at hand, we can apply standard constraint propagation algorithms for renewable resources like e.g. timetable propagation (TP) and edge finding [4]. Our current implementation of the `cumulativemm`-constraint only integrates TP. TP is based on the evaluation and reasoning on the so-called *compulsory parts* cp_j of the jobs $j \in J$ (Schutt et al. [28]):

$$\begin{aligned} \textbf{If} \quad & lb(s_j) + d^{min}_j > ub(s_j) : \\ & cp_j = \left\{ ub(s_j), \ldots, lb(s_j) + d^{min}_j - 1 \right\} \\ \textbf{Else} \quad & cp_j = \emptyset \end{aligned} \tag{34}$$

If $cp_j \neq \emptyset$, job j is surely processed at all timepoints $t \in cp_j$.

Example 4.1 TP for multi-mode jobs. Assume that in the course of the BaB-algorithm of SCIP and after the redundancy check, our constraint propagation procedure has the following input:

$$(\text{domain}(s_1), \text{domain}(x_{1,1}), d_{1,1}, c^{\rho}_{1,1,1}) = (\{3,4,5\}, \{0,1\}, 2, 1)$$
$$(\text{domain}(s_1), \text{domain}(x_{1,2}), d_{1,2}, c^{\rho}_{1,2,1}) = (\{3,4,5\}, \{0,1\}, 3, 2)$$
$$(\text{domain}(s_2), \text{domain}(x_{2,1}), d_{2,1}, c^{\rho}_{2,1,1}) = (\{2,3,4\}, \{1\}, 3, 2)$$

The maximal capacity of the renewable resource 1, $C^{\rho}_1 = 2$. We can see that job 2 is processed in mode 1, as $x_{2,1} = 1$. Thus, $d^{min}_2 = 3$ and $c^{min}_{2,1} = 2$. Moreover, job 2 is surely processed at the time point 4, as its compulsory part $cp_2 = \{4\}$. Next, we consider job 1 with $d^{min}_1 = 2$ and $c^{min}_{1,1} = 1$. $cp_1 = \emptyset$ but we can deduce a domain update. As $lb(s_1) + d^{min}_1 = 3 + 2 \geq 4$, starting job 1 at its lower bound would lead to a resource conflict at the time point 4. The TP algorithm will find the largest time point $t_1 - 1 = 4$ such that the capacity is violated ($2 + 1 > 2$). After that $lb(s_1)$ would be updated to $t_1 = 5$ which equals $ub(s_1)$ and a new compulsory part of job 1 $cp_1 = \{5, 6\}$ is evaluated.

Note, that the principles of our constraint propagation procedure are standard techniques. These are applied in a similar way in CP solvers like e.g. JaCop [16] which provide the `cumulative`-constraint supporting variable durations and resource consumptions for every job.

The idea of integrating explanation generation, i.e. of processing reasons for the deduced domain reductions or inconsistencies to a SAT solving mechanism is rather new. To our knowledge, there are only two optimization frameworks integrating this feature, i.e. SCIP and the G12 Constraint Programming Platform [9]. Schutt [27] describes principles for explanation generation in the context of jobs having variable durations and resource consumptions. These explanation generation techniques are integrated in the `cumulative`-constraint provided by the G12 Constraint Programming Platform [9]. In our `cumulativemm`-constraint, we explain the reasons for the domain reductions or inconsistencies in a different way. In the next two sections we introduce our strategy for explanation generation and compare it to the strategy of Schutt [27].

4.1. Explanations for timetable propagation with multi-mode jobs

In order to integrate our constraint into the SCIP-intern CA mechanism, we have to provide functions for the `cumulativemm`-constraint which derive explanations for the inconsistencies or domain updates detected by the TP algorithm. These explanations can be seen as clauses consisting of boolean literals of the form:

$$\left\{ [\![s_j \leq t]\!], \ [\![s_j \geq t]\!] : \ t = 0, \ldots, T-1 \right\}$$

and

$$\left\{ [\![x_{j,k} == 0]\!], \ [\![x_{j,k} == 1]\!] \right\}$$

Assume now, the TP algorithm found an inconsistency because of the jobs having their compulsory parts cp_j (see (34)) at time t_κ and cause a resource violation:

$$\sum_{j:t_\kappa \in cp_j} c^{\min}_{j,r} > C^\rho_r$$

Then, our constraint handler derives the following explanation:

$$\bigwedge_{j:t_\kappa \in cp_j} EXP_j \Longrightarrow \texttt{false} \tag{35}$$

(35) can be divided into the subexplanations EXP_j for every job participating in the conflict:

$$EXP_j = [[t_\kappa - d^{min}_j + 1 \le s_j]] \wedge [[s_j \le t_\kappa]] \wedge \bigwedge_{m:m\in M_j \text{ and } ub(x_{j,m})=0} [[x_{j,m} == 0]] \tag{36}$$

EXP_j is correct as in a situation, where the variables s_j and $x_{j,m}$ have bounds as in (36), the compulsory parts cp_j of the involved jobs would again include the timepoint t_κ and this would again lead to a resource violation. Note, that d^{min}_j and $c^{min}_{j,r}$ are calculated by (32) and (33), respectively.

In addition to explanations for inconsistencies, our constraint handler also processes explanations for domain updates deduced by our TP algorithm to the SCIP-intern CA mechanism. SCIP stores information about the time the bound changes took place through a so-called *bound change index* (BCI) and about the constraints which processed the domain updates. In the course of the BaB-algorithm, the SCIP-intern CA can ask our constraint handler for an explanation of the new lower bound $lb^*(s_i)$ of job i at the BCI b, if it was deduced by the `cumulativemm`-constraint.

Therefore, we introduce the timepoint $t = lb^*(s_i) - 1$. Jobs $j \in J\backslash\{i\}$ with compulsory parts cp^b_j where $t \in cp^b_j$ such that:

$$c^{\min,b}_{i,r} + \sum_{j\neq i: t\in cp_j} c^{\min,b}_{j,r} > C^\rho_r$$

were responsible for the bound change at the BCI b. Additionally the lower bound $lb(s_i)$ of job i has to exceed a certain value for the domain update. The complete explanation consists of two clauses e_i and f, where e_i contains the minimal lower bound of s_i and f the compulsory parts cp^b_j of jobs $j \neq i$ with $t \in cp^b_j$. e_i is given as follows:

$$e_i = [[t - d^{min,b}_i + 1 \le s_i]] \wedge \bigwedge_{m:m\in M_i \text{ and } ub(x^b_{i,m})=0} [[x_{i,m} == 0]]$$

The clause f is as follows:

$$f = \bigwedge_{j:j\neq i \text{ and } t\in cp^b_j} EXP_j$$

where EXP_j is given through a small variation of (36). We replace t_κ by t, $d^{\min}_j$ by $d^{\min,b}_j$ and $ub(x_{j,m})$ by $ub^b(x_{j,m})$. Thus, we evaluate the latter values for the given BCI b. The complete explanation for the bound change $lb(s_i) \to lb^*(s_i)$ is given through:

$$e_i \wedge f \Longrightarrow [[lb^*(s_i) \le s_i]] \tag{37}$$

The argument for the correctness of the above explanation is the same as in the inconsistency case. Note, that the explanation (37) depends on the BCI b.

In our TP algorithm we process the bound changes in a pointwise manner, i.e. we guarantee that $lb^*(s_i) - lb(s_i) \le d^{min,b}_j$. With this, we want to imitate the pointwise explanations proposed by Schutt et al. [28]. The explanations for the upper bound changes are processed in a symmetric way.

SCIP generates a conflict graph based on the explanations of our constraint handler and the explanation generators of the other constraints and can possibly deduce nogoods and backjumps for the following branches in the BaB algorithm. If a constraint propagation algorithm leads to many domain reductions, CA can be very efficient for the complete solution procedure. An example for the CA process, i.e. for nogood generation on a conflict graph can be found in Schutt et al. [31].

The following example illustrates a possible outcome of our explanation generation procedure.

Example 4.2. Firstly, we extend Example 4.1 by another job with two modes and the following input:

$$(\text{domain}(s_3), \text{domain}(x_{3,1}), d_{3,1}, c^\rho_{3,1,1}) = (\{5\}, \{1\}, 2, 2)$$

Note, that there is a resource conflict at time point $t_\kappa = 6$, as $6 \in cp_1 \cap cp_3$ and $c^\rho_{3,1,1}(=2) + c^{\min}_{1,1}(=1) > 2$. The explanation for this inconsistency is as follows:

$$([[5 \le s_1]] \wedge [[s_1 \le 6]]) \wedge ([[5 \le s_3]] \wedge [[s_3 \le 6]] \wedge [[x_{3,2} == 0]]) \Longrightarrow \texttt{false} \tag{38}$$

Note, that job 3 is processed in mode 1 and for job 1 it holds, that $ub(x_{1,k}) > 0, \forall k \in M_1$. After the initialization of the SCIP-intern CA, SCIP asks our constraint handler for the reason of the lower bound change of s_1 from 3 to 5 from Example 4.1, i.e. an explanation for the literal $[[5 \le s_1]]$.

Our constraint handler gives the following explanation:

$$[[3 \le s_1]] \wedge ([[3 \le s_2]] \wedge [[s_2 \le 4]] \wedge [[x_{2,2} == 0]]) \Longrightarrow [[5 \le s_1]] \tag{39}$$

Every boolean literal from (38) and (39) is added as a new node to the SCIP-intern conflict graph. Moreover, an arc is constructed from every boolean literal of the left-hand side of the explanation to the boolean literal on the right-hand side.

4.2. Comparison to other explanation generation techniques and possible improvements

Schutt [27, p.96] also introduces explanations for the `cumulative`-constraint where the durations and the resource consumptions of the jobs can be variables. In our G12-model of Section 3, we use this constraint with the duration vector $\mathbf{d}$ and the resource consumption vector $\mathbf{c}^\rho_r$ to model the resource constraint for the renewable resource $r \in R$. After the transformation given by (11) and (12), the G12 solution approach will only use the variable vectors $\mathbf{d}'$ and $\mathbf{c}'^\rho_r$ in the `cumulative`-constraint. These are connected to the original durations and resource consumptions by the `element`-constraint (see (11) and (12)). With our notation, the preliminary version of the explanations for a lower bound update of s_i to $lb^*(s_i)$ applied in the `cumulative`-constraint (see [27, p.96]) are as follows (at the BCI b):

$$\begin{aligned} &[[lb^*(s_i) - lb^b(d'_i) \le s_i]] \wedge [[lb^b(d'_i) \le d'_i]] \wedge [[lb^b(c'^\rho_{i,r}) \le c'^\rho_{i,r}]] \quad \wedge \\ \bigwedge_{j:j\neq i \text{ and } t\in cp^b_j} &[[lb^*(s_i) - lb^b(d'_j) \le s_j]] \wedge [[s_j \le lb^*(s_i) - 1]] \wedge [[lb^b(d'_j) \le d'_j]] \quad \wedge \\ \bigwedge_{j:j\neq i \text{ and } t\in cp^b_j} &[[lb^b(c'^\rho_{j,r}) \le c'^\rho_{j,r}]] \Longrightarrow [[lb^*(s_i) \le s_i]] \end{aligned} \tag{40}$$

Schutt [27] notes, that these explanations can be strengthened by choosing different values q_i and l_i instead of $lb^b(d'_i)$ and $lb^b(c'^\rho_{i,r})$, respectively. For example consider the case where the domain of d'_i is internally encoded as a range of consecutive values but the set $\mathcal{D}_i = \{d_{i,m},\ m \in M_i\}$ consists of nonconsecutive values and it holds that $lb^b(d'_i) \notin \mathcal{D}_i$. Then, by using $q = \min\{d_{i,m} : d_{i,m} \ge lb^b(d'_i)\}$, the explanation (40) can be strengthened. Schutt [27] specifies the values q_i and l_i which lead to the strongest explanations.

In our explanation for a domain update (see (37)), we omit the part where the resource consumptions of the involved jobs are explained as in (40). This is due to the fact, that as soon as certain modes are excluded, we can reason about the minimal duration $d_j^{\min}$ and the minimal resource consumption $c_{j,r}^{\min}$. Thus we do not have to introduce explanations for both durations and resource consumptions. This can be an advantage compared to the explanations of Schutt [27].

Assume therefore, that we are in a situation in a node c of the BaB tree where the input is the same for both algorithms, and both algorithms already generated the explanations (37) and (40), respectively at node b with the same set of jobs J_{exp} involved in both explanations. Additionally,

$$d_j^{\min,b} = lb^b(d'_j),\ \forall j \in J_{\text{exp}} \tag{41}$$

$$c_{j,r}^{\min,b} = lb^b(c'^{\rho}_{j,r}),\ \forall j \in J_{\text{exp}} \tag{42}$$

Moreover, the left hand side of (37) is true at node c and

$$\exists\, k \in J_{\text{exp}} :\ \ t_b \in cp_k^c \wedge lb^b\left(c'^{\rho}_{k,r}\right) > lb^c\left(c'^{\rho}_{k,r}\right) \tag{43}$$

whereat the first two lines of (40) are true. As (37) is true, our algorithm will immediately deduce $lb^*(s_i) \le s_i$.

Because of (43), the G12-algorithm will not immediately deduce the latter lower bound update. As the first two lines of (40) are true and (41) and (42) hold, we can conclude that

$$x_j \in M_j^b = \left\{m : m \in M_j \text{ and } ub\left(x_{j,m}^b\right) > 0\right\},\ \forall j \in J_{\text{exp}} \tag{44}$$

and thus $lb^b(c'^{\rho}_{k,r}) \le c'^{\rho}_{k,r}$, i.e. $lb^c(c'^{\rho}_{k,r})$ can be updated to $lb^b(c'^{\rho}_{k,r})$. Now, the complete left hand side of (40) is true and the G12-algorithm will also deduce $lb^*(s_i) \le s_i$.

The update of $lb^c(x_k)$ outlined in (44) and the update of $lb^c(c'^{\rho}_{k,r})$ have to be processed by the `element`-constraints (11) and (12) in the G12-algorithm before the explanation (40) leads to the update of $lb^*(s_i)$. As this update happens immediately with our explanation (37), there are cases where our explanation generation strategy can lead to time savings.

Our explanations can also be strengthened.

Example 4.3. Consider a job 1 with the following input at the current node:

$$(\text{domain}(x_{1,1}), d_{1,1}, c^{\rho}_{1,1,1}) = (\{0\}, 2, 2)$$
$$(\text{domain}(x_{1,2}), d_{1,2}, c^{\rho}_{1,2,1}) = (\{0\}, 3, 3)$$
$$(\text{domain}(x_{1,3}), d_{1,3}, c^{\rho}_{1,3,1}) = (\{1\}, 4, 3)$$

Assume that our TP algorithm would detect an inconsistency at the time point 4 and job 1 is involved in the latter inconsistency, i.e. $4 \in cp_1$. As $d_1^{\min} = 4$, the part of the explanation containing job 1 is as follows:

$$[\![1 \le s_1]\!] \wedge [\![s_1 \le 4]\!] \wedge [\![x_{1,1} == 0]\!] \wedge [\![x_{1,2} == 0]\!]$$

If the global domain of s_1 equals $\{2, \ldots, 6\}$, we can strengthen the explanation as

$$[\![1 \le s_1]\!] \text{ is globally true,}$$
$$[\![x_{1,1} == 0]\!] \wedge [\![x_{1,2} == 0]\!] \implies [\![x_{1,1} == 0]\!],$$
$$[\![x_{1,1} == 0]\!] \wedge [\![s_1 \le 4]\!] \implies 4 \in cp_1,$$
$$c^{\rho}_{1,2,1} = c^{\rho}_{1,3,1}.$$

Thus, we can use the following stronger explanation for the compulsory part of job 1:

$$[\![s_1 \le 4]\!] \wedge [\![x_{1,1} == 0]\!]$$

Motivated by the above example, assume that job i is part of the job set J_{exp} involved in the explanation (37) w.r.t. time point t. Moreover, let $lb^g(s_i)$ be its global lower bound. We can possibly strengthen the explanation (37) by strengthening the part of the explanation integrating job $i \in J_{\text{exp}}$.

This can done in two steps:

Firstly, if $t - d_i^{\min} + 1 < lb^g(s_i)$, omit $[\![t - d_i^{\min} + 1 \le s_i]\!]$.

Set $d_i^{\min} :\leftarrow \min\{d_i^{\min}, t - lb^g(s_i) + 1\}$.

Secondly, we determine the set B_i^* consisting of the modes $m \in M_i$ which fulfill:

$$d_{i,m} \ge d_i^{\min} \tag{45}$$

$$c^{\rho}_{i,m,r} + \sum_{j \in J_{\text{exp}} \setminus \{i\}} c_{j,r}^{\min} > C_r^{\rho} \tag{46}$$

Now, we can substitute

$$\bigwedge_{m : m \in M_i \text{ and } ub(x_{i,m}^b) = 0} [\![x_{i,m} == 0]\!]$$

by

$$\bigwedge_{m : m \in M_i \setminus B_i^*} [\![x_{i,m} == 0]\!]$$

Because of the evaluation of $d_i^{\min}$ and $c_{i,r}^{\min}$ in (32) and (33) and because of (45) and (46) it holds that

$$M_i \setminus B_i^* \subseteq \{m : m \in M_i \text{ and } ub(x_{i,m}) = 0\}$$

Thus, the part of the explanation (37) integrating job i is possibly stronger. Finally, we update $c_{i,r}^{\min} :\leftarrow \min\{c^{\rho}_{i,m,r} : m \in B_i^*\}$. After the latter update, we continue with the next non-processed job.

5. Computational experiments

The three CP models from Section 3 were solved on the Vienna Scientific Cluster (VSC). Thereby, the cluster nodes integrate a X86-64 architecture running under Red Hat/Linux with two six-core Intel Westmere X5650 processors of 2,66GHz and with 24GB RAM. For the solution of the models from Section 3.1, we used the G12 Constraint Programming Platform [9] 2.0.0 provided by the NICTA research team [22]. Thereby, we formulated the models in Zinc and solved them by the LCG-plugin `g12_fdx`. For the implementation of the constraint handler `cumulativemm` and the formulation and solution of the SCIP-models of Section 3.2, we used SCIP 3.1.0 in combination with the programming languages `C/C++`. We set the parameters in SCIP such that feasibility is detected fast (with `SCIP_PARAMEMPHASIS_FEASIBILITY`). Furthermore, we impose a memory limit of 2GB RAM for instances with less than 100 jobs and of 3GB for the 100-job instances.

The three CP-models are denoted by the following abbreviations:

`G12` The model from Section 3.1 formulated in Zinc.
`SCIPopt` The SCIP-model of Section 3.2 integrating the existing `optcumulative`-constraint.
`SCIP` The SCIP-model of Section 3.2 integrating our `cumulativemm`-constraint.

Moreover, for every model we distinguish two solution approaches which differ in the generation of the initial domains:

`Max` The initial domains of the starting time variables are evaluated by forward (backward) recursion based on the trivial upper bound $T_{\max} = \sum_{j \in J} d_j^{\max}$, where $d_j^{\max}$ is the maximal mode duration of job j.
`Best` The initial domains are generated based on twelve different upper bounds $T_1, \ldots, T_{12}$ where T_1 equals the best known upper bound from the literature and $T_l = T_{l-1} + 4,\ \forall l = 2, \ldots, 12$. A model is run on the processor $l = 1, \ldots, 12$ with initial domains based on T_l. Hence, in this

case we apply a parallel approach which applies twelve processors. In the end, we take the best results w.r.t. all twelve processors.

In total, we compare six different solution approaches to the state-of-the-art exact approach (Branch-and-Cut) for the MRCPSP (MMBAC) of Zhu et al. [37]. In addition, the average deviation from the critical path lower bound is compared to the one obtained by the best metaheuristic for this problem (VANP14) [35].

The test instances are from the PSPLIB (Kolisch and Sprecher [18]) and from the instance sets MMLIB and MMLIB+ generated by Peteghem and Vanhoucke [35] (see also [36]). In total, we consider instances with 20, 30, 50 and 100 jobs, whereas we abort the solution process after a time limit of 180 s, 360 s, 5400 s and 7200 s, respectively. To assure a fair comparison to the approach of Zhu et al. [37], we used time limits which are approx. 10% of their time limits for the 20- and 30-job instances. The factor 10 is based on the values of our processor and their processor (Intel Westmere X5650 with 2,66GHz and Xeon with 1.80 Ghz from 2004, respectively) in the Passmark CPU benchmark [25]. Moreover, we also use the comparison factor of approx. 0.68 based on the clock rates of both CPUs for further experiments. This leads to time limits of 1200 s and 2400 s.

We compare the different approaches based on the following measurements:

- `#feas` Number of instances where a feasible solution could be found within the given time limit.
- `#opt` Number of instances solved to optimality within the given time limit.
- `#best` Number of solutions whose makespan equals or improves the best known makespan from the literature.
- t_{tot} The average solution time for all instances (we take the minimal solution time of all processors for one instance in the parallel approach).
- $G_{opt}(\%)$ The average optimality gap for the instances where a feasible solution was found (we take the optimality gap of the processor which found the best makespan in the parallel approach).
- $G_{cplb}(\%)$ The average gap to the lower bound given by the length of the critical path for the instances where a feasible solution was found (again we take the optimality gap of the processor which found the best makespan in the parallel approach).
- I_{lb} Number of instances where we could improve the best known lower bound.
- I_{ub} Number of instances where we could improve the best known makespan from the literature.

Note that for the 20-job instances we do not integrate the columns corresponding to I_{lb} and I_{ub} as all of these instances had been solved to optimality before. The same holds for the 30-job instances but here for a different reason. We did not have access to the detailed lower and upper bounds of Zhu et al. [37] for every instance but only to the accumulated results presented in their paper. In case of the remaining instance sets the best known makespans are reported on the website `www.mmlib.eu`.[2] The latter were evaluated by Peteghem and Vanhoucke [35], Geiger [10] and other authors who have not published their results in a scientific journal, yet. The best known lower bounds for these instances are given by the length of the critical path in the precedence network as to our knowledge up to now no tighter lower bounds have been computed in the literature, yet.

[2] Our evaluations concerning the best known upper bounds are based on the state of this website on the 28.09.2016.

Table 1
Results on the 20-job instances.

	`#feas`	`#opt`	`#best`	t_{tot}	$G_{opt}(\%)$
Time limits of 180 s					
`G12Max`	554	547	548	11.73	0.12
`G12Best`	554	551	553	5.2	0.05
`SCIPMax`	554	529	534	12.33	1.6
`SCIPBest`	554	538	551	9.06	0.87
`SCIPoptMax`	554	489	497	25.72	5.27
`SCIPoptBest`	554	496	524	22.29	4.63
Time limits of 1200 s					
`G12Max`	554	552	554	20.58	0.05
`G12Best`	554	554	554	10.39	0.0
`SCIPMax`	554	547	550	33.96	0.32
`SCIPBest`	554	552	554	22.74	0.07
`SCIPoptMax`	554	504	519	130.37	3.16
`SCIPoptBest`	554	512	543	113.66	2.85
`MMBAC`	554	554	554	32.06	0
`VANP14`	554	–	–	–	0.32

Note that optimality gaps for the G12 approaches are calculated w.r.t. the lower bounds computed by the best SCIP approach. This is due to the fact that we did not find a way to receive these values from the G12 framework. Therefore, we also omit the results concerning the lower bound improvements for the G12 approaches.

Table 1 shows the results for our models and the state-of-the-art exact approach of Zhu et al. [37] (`MMBAC`) on the 554 feasible 20-job instances from the PSPLIB. For these instances `MMBAC` outperforms all of our approaches for the small time limits as they can solve all feasible instances to optimality. The same holds for the single core approaches (`G12Max`, `SCIPMax` and `SCIPoptMax`) when applying the larger time limit. Nevertheless, `G12Max` is highly competitive to the approach of Zhu et al. [37] in this scenario. The parallel approach `G12Best` outperforms the state-of-the-art exact approach from the literature for the larger time limits. We can also solve all feasible instances to optimality, but `MMBAC` is two times slower, when taking into account the clock rates of both processors.

Furthermore, the SCIP-approaches using our `cumulativemm`-constraint significantly outperform the SCIP-approaches integrating the existing `optcumulative`-constraint. With `SCIPBest` we can solve 40 more instances to optimality in an approximately five times lower average solution time t_{tot} compared to `SCIPoptBest` for the larger time limits. Moreover, `SCIPBest` is competitive to `MMBAC` and `G12Best` in this scenario. Note that within a time limit of approximately 3806 s the approach `SCIPBest` can solve all 554 instances to optimality. The average solution time is 28.47 s in the latter case.

Furthermore, for the large time limit, the G12 approaches and the approach `SCIPBest` produce a better optimality gap than the best metaheuristic presented by Peteghem and Vanhoucke [35].[3]

Table 2shows the results for the 552 feasible 30-job instances from the PSPLIB. Again, considering the small time limits `MMBAC` outperforms all of our approaches. However, we can already solve 9 more instances to optimality than `MMBAC` with the single-core approach `G12Max` for the larger time limits. Moreover, in this situation `G12Max` is approx. 1.23 times faster than `MMBAC`, again taking into account the clock rates of both processors. Furthermore, the parallel approach `G12Best` significantly outperforms `MMBAC` both regarding average solution times and solution quality in this scenario. Again, `SCIPBest` (`SCIPMax`) is considerably better than `SCIPoptBest` (`SCIPoptMax`). We can solve 36 (44) more in-

[3] Note that in this case, we can feasibly make this comparison as Peteghem and Vanhoucke [35] use the gap of their best makespan w.r.t. the known optimal solution of an instance for these instances.

Table 2
Results on the 30-job instances.

	#feas	#opt	#best	t_{tot}	$G_{opt}(\%)$	$G_{cplb}(\%)$
Time limits of 360 s						
G12Max	552	466	480	91.2	4.16	14.32
G12Best	552	495	514	58.86	3.01	12.96
SCIPMax	552	486	492	53.06	4.72	14.11
SCIPBest	552	494	507	46.0	3.87	13.39
SCIPoptMax	552	440	451	84.64	10.82	15.60
SCIPoptBest	552	454	473	72.69	8.86	14.10
Time limits of 2400 s						
G12Max	552	515	521	212.44	2.59	12.93
G12Best	552	521	537	180.50	1.97	12.50
SCIPMax	552	500	508	259.71	3.37	13.32
SCIPBest	552	504	517	232.95	2.96	12.97
SCIPoptMax	552	456	470	452.46	8.15	14.94
SCIPoptBest	552	468	480	401.74	7.29	13.79
MMBAC	552	506	529	393.13	–	–
VANP14	552	–	–	–	–	13.66

Table 3
Results on the 50-job instances from MMLIB.

	#feas	#opt	#best	t_{tot}	$G_{opt}(\%)$	$G_{cplb}(\%)$	I_{ub}	I_{lb}
SCIPMax	540	405	415	1409.69	17.73 (17.86)	34.96 (34.21)	14	276
SCIPBest	540	420	440	1252.25	9.86 (9.88)	26.08 (26.01)	18	280
G12Max	532	363	377	1952.36	11.21	27.61	19	–
G12Best	539	367	413	1861.9	8.87	25.34	20	–
VANP14	540	–	–	–	–	23.79	–	–

Table 4
Results on the 100-job instances from MMLIB.

	#feas	#opt	#best	t_{tot}	$G_{opt}(\%)$	$G_{cplb}(\%)$	I_{ub}	I_{lb}
SCIPMax	518	312	322	3127.93	341.33 (146.09)	378.03 (154.19)	18	213
SCIPBest	535	338	348	2740.31	33.71 (27.13)	45.58 (32.20)	23	217
G12Max	219	150	154	5963.62	21.35	25.71	6	–
G12Best	404	245	260	5255.9	16.91	21.29	10	–
VANP14	540	–	–	–	–	24.02 (21.51)	–	–

stances to optimality and are ≈ 42% (43%) faster when using time limits of 2400 s.

As the Branch-and-Cut approach of Zhu et al. [37] is based on a MIP formulation of the MRCPSP, it is highly dependent of a starting solution with a small makespan to reduce the number of binary variables. They use starting solutions computed by a problem specific heuristic whose makespan on average only deviates by 2.18% from the best known makespans of the PSPLIB. An advantage of our SCIP- and G12 approaches is that they still produce competitive results with the relatively high upper bound $T_{\max}$ as input.

Moreover, the input upper bounds leading to the best results in the parallel approach G12Best deviate on average by 19.69% and 22.14% from the best known upper bounds from the literature for the 20- and 30-job instances, respectively.

Furthermore, for this instance set, all of our solution approaches except SCIPopt exhibit a better gap to the critical path lower bound than the best metaheuristic presented by Peteghem and Vanhoucke [35] for the larger time limits.[4]

Now, we present our results for new instances with 50 and 100 jobs generated by Peteghem and Vanhoucke [35]. To our knowledge, these have not been solved exactly before.

Tables 3 and 4 contain the results for the runs with the 540 feasible 50-job and 540 feasible 100-job instances from the MMLIB, respectively. For these instance sets, the approaches integrating the best SCIP model significantly outperform the approaches integrating the G12-model.

The G12 approaches only produce better average optimality gaps and gaps w.r.t. to the critical path lower bound, also if the average value is computed w.r.t. to the instances where both the SCIP and the respective G12 approaches, can compute a feasible solution. This can be observed when regarding the values in brackets near the gap measurements in the SCIP approaches.[5]

The performance difference of the G12 approaches compared to our best SCIP approaches for the 20- and 30-job instances and the 50- and 100-job instances can be explained in the following way. In the G12 formulation, we use variables as indices of parameter vectors as a concept to model the mode assignment. The G12 solver internally transforms these constructs to a number of element-constraints (see Section 3.1). In this context, $|J| \cdot (1 + |R| + |N|)$ new variables d'_i, $c'^{\rho}_{i,r}$ and $c'^{\nu}_{i,r}$ and constraints (see (12), (11) and (13)) have to be introduced. In general, the BaB algorithm has to consider all of these additional variables and constraints in the course of its internal branching, constraint propagation and CA mechanisms. One can see that this solver-internal transformation still works in an efficient way for the 20- and 30-job instances. However, for the 50- and 100-job instances the additional introduction of new variables and constraints leads to a great loss of efficiency of the overall procedure. In the SCIP formulation, we do not need a transformation of the original model. More precisely,

[4] However, one should always keep in mind that this metaheuristic is significantly faster in practice than our exact approach.

[5] Note that the value in brackets for VAN14 corresponds to the gap w.r.t. the instances where the approach SCIPBest can find a feasible solution.

we tackle the mode assignment by binary variables and the internal handling of `cumulativemm`. Obviously, this leads to a significantly more efficient behavior of the overall solution procedure for the 50- and 100-job instances.

A hint to the observation that our best SCIP algorithms (`SCIPMax` and `SCIPBest`) should work in a more effective way than the respective G12 algorithms was already given in Section 4.2. A reason why we cannot observe the latter for the 20- and 30-job instances can also be the additional overhead of the SCIP implementation. The SCIP internal algorithm captures a wide variety of additional techniques, e.g. specialized MIP concepts, which only have a small contribution to the overall solution procedure for our special problem. This overhead does not occur in the G12 algorithm. Hence, this additional computational effort can lead to the weaker performance of our best SCIP algorithms compared to the G12 algorithms for the 20- and 30-job instances. For the 50- and 100-job instances the impact of this overhead is of lesser extent and the above assumption about the more efficient behavior of our best SCIP approaches is computationally supported.

In total, we can solve approx. 78% of the 50-job instances and 63% of the 100-job instances to optimality within the given time limits. Moreover, we improved the best known makespans of 18 instances with 50 jobs and 23 instances with 100 jobs. Furthermore, we can improve the lower bounds for approx. 52% of the 50-job instances from `MMLIB` on average by approx. 28%. For 40% of the 100-job instances from `MMLIB` tighter lower bounds could be computed. These are on average approx. 23% better than the best known lower bounds. Furthermore, the input upper bounds leading to the best results in the parallel approach `SCIPBest` are on average 37.41% and 131.20% higher than best known makespans for the 50- and 100-job instances, respectively. This is again an indication that our approaches can also produce good results with relatively high upper bounds as input.

Note that in case of the 50-job instances, we found out that at least 65% of the heuristic solutions given by Peteghem and Vanhoucke [35] correspond to an optimal solution. For the 100-job instances, this holds for at least 47%. Furthermore, the gap to the critical path lower bound computed by the approach `SCIPBest` is competitive compared to the metaheuristic `VANP14` for the 50-job instances. However, for the 100-job instances our SCIP approaches reach their limits w.r.t. producing high quality upper bounds in total for all instances.

To test the influence of the five different project parameters based on which the new data sets of `MMLIB` were generated (see [35] and [36]), i.e. the order strength (OS), and the renewable and nonrenewable resource strength and resource factor ($RS^{\rho(\nu)}$ and $RF^{\rho(\nu)}$), on the performance of our best solution approaches `SCIPBest` (measured by the optimality gap), we conducted a multiple linear regression analysis similar to Peteghem and Vanhoucke [35] for the 50- and 100-job instances of `MMLIB`. For the 50-job instances, the parameters -0.188, -0.561 and 0.379 are significant coefficients (confidence level of 1%) for OS, RS^{ρ} and RF^{ρ} in the multiple linear regression model, respectively. Moreover, the constant 0.168 is also significant. The coefficient of determination R^2 is approx. 0.66. Thus, for these instance, we can observe that with an increasing number of precedence relations, the performance of the approach `SCIPBest` increases. This can be explained by the fact that the size of solution space tendentially decreases with an increasing number of precedence relations. Furthermore, the less scarce the renewable resources become, the better is the performance of `SCIPBest`. Finally, one can observe, that the performance of `SCIPBest` decreases when the average resource consumption of the multi-mode jobs is increased. For the 100-job instances only the coefficients -17.61 and 11.60 are significant for RS^{ρ} and RF^{ρ}, respectively. Tables 5 and 6 show the results for the different parameter groups which correspond to significant coefficients for the 50- and 100-job instances, respectively.

Table 5
50 jobs: The results for parameter groups corresponding to significant coefficients.

	`#opt`	`#best`	t_{tot}	$G_{opt}(\%)$
OS = 0.25	132	142	1489.39	14.18
OS = 0.5	142	145	1200.68	10.70
OS = 0.75	146	153	1066.67	4.71
$RS^{\rho} = 0.25$	85	93	2927.24	27.99
$RS^{\rho} = 0.5$	155	167	828.96	1.60
$RS^{\rho} = 0.75$	180	180	0.54	0
$RF^{\rho} = 0.5$	262	268	208.68	0.38
$RF^{\rho} = 1$	158	172	2295.81	19.94

Table 6
100 jobs: The results for parameter groups corresponding to significant coefficients.

	`#opt`	`#best`	t_{tot}	$G_{opt}(\%)$
$RS^{\rho} = 0.25$	40	43	5534.49	875.72
$RS^{\rho} = 0.5$	122	127	2385.39	5.88
$RS^{\rho} = 0.75$	176	178	1383.83	0.11
$RF^{\rho} = 0.5$	220	223	1383.83	4.23
$RF^{\rho} = 1$	118	125	4096.78	583.59

Finally, Table 7 shows the results of the approaches integrating our best SCIP-model for the renownedly more complex 1620 feasible 50-job and 1620 feasible 100-job instances from the instance set `MMLIB+`. As instances of `MMLIB+` on average integrate more nonrenewable and renewable resources than those of `MMLIB`, one can assume that the performance of the G12 solution approaches is even worse for these instances. Therefore, we only consider the approaches integrating our best SCIP model for these evaluations. For this instance set 2318 lower bounds are on average improved by approx. 51%. Furthermore, the input upper bounds leading to the best results with `SCIPBest` deviate on average by 555.21% from the best known makespans for `MMLIB+`.

Note that the value in brackets near G_{cplb} for `VANP14` corresponds to the gap to the critical path lower bound for the instances where `SCIPBest` can find a feasible solution.

A file with more detailed results for every examined instance set is provided as supplementary data to this paper.

A disadvantage of our approaches is that not for every instance of `MMLIB100` and of `MMLIB+` a feasible solution can be found. This disadvantage can be overcome by running a heuristic (e.g. the one of [35] with a limit of 5000 schedules) in the beginning to obtain a feasible initial solution for every instance as input for our solution approaches. Furthermore, in this context, the influence of scheduling specific primal heuristics integrated into the SCIP framework on our solution approaches would be an interesting future research topic.

6. Conclusion

In our paper, we introduced a generalization of the exact CP-SAT approaches for the SRCPSP to the MRCPSP. This generalization can on the one hand be achieved on the modeling level. We introduced formulations of the MRCPSP in optimization frameworks (G12 and SCIP) which integrate a BaB-algorithm in combination with CP and SAT techniques. One formulation is usable via the Zinc modeling language which supports the solution of models by LCG, the state-of-the-art exact approach for variants of the SR-CPSP. Moreover, we proposed two formulations for the optimization framework SCIP. One of the latter is based on our new constraint handler called `cumulativemm` and the other one on the

Table 7
Results on the MMLIB+ instances.

	#feas	#opt	#best	t_{tot}	G_{opt}(%)	G_{cplb}(%)	I_{ub}	I_{lb}
SCIPMax	3169	596	670	4236.01	565.11	751.96	77	2308
SCIPBest	3175	670	777	6150.85	478.65	656.52	110	2318
VANP14	3240	–	–	–	–	92.76 (81.20)	–	–

existing optcumulative-constraint introduced by Heinz et al. [14].

The computational experiments show that for the 20- and 30-job instances at least one of our proposed approaches can outperform the state-of-the-art exact algorithm of Zhu et al. [37] for the MRCPSP when the approaches are compared based on the clock rates of the CPUs on which they are run. In this situation, our parallel G12-algorithm is almost two times faster than their approach on the 20-job instances. Moreover, on the 30-job instances already our single-core approach using a trivial upper bound as input for the G12-algorithm can solve 9 more instances to optimality and is approx. 1.23 times faster. A clear advantage of our approaches is that they also produce competitive results when large upper bounds are used as input. In contrast, the quality of the approach of Zhu et al. [37] is highly dependent on small UBs as input to reduce the initial number of variables.

Moreover, the SCIP-approaches which apply our cumulativemm-constraint are significantly better than the SCIP-approaches integrating the existing optcumulative-constraint. For the 20- and 30-job instances, we could solve 40 and 36 more problem instances to optimality in approximately five and two times smaller average solution times, respectively.

Finally, we are the first to exactly solve new MRCPSP instances with 50 and 100 jobs from the literature [35]. On these instance sets, our SCIP-approaches with cumulativemm significantly outperform the G12-approaches. In total, we close (find the optimal solution and prove its optimality for) 628 open instances with 50- and 100-jobs from the literature [35]. Moreover, we improve the best known makespans reported Online [24] for 151 of these instances. In addition, we improve the best known lower bound of 2815 instances on average by approx. 46%.

Acknowledgments

This article was supported by the Open Access Publishing Fund of the University of Vienna. We would like to thank the SCIP team and especially Stefan Heinz and Jens Schulz for their valuable information about the implementation of constraint handlers. Moreover, we are grateful to Prof. Luca Di Gaspero for his hint on an efficient CP-formulation of the MRCPSP. Finally, we would like to thank the anonymous reviewers for their highly valuable comments. Their input led to an improvement of the presentation of our article.

Supplementary material

Supplementary material associated with this article can be found, in the online version, at 10.1016/j.orp.2017.01.002.

References

[1] Achterberg T. SCIP-a framework to integrate constraint and mixed integer programming. Konrad-Zuse-Zentrum für Informationstechnik Berlin; 2004.
[2] Achterberg T. SCIP: solving constraint integer programs. Math Program Comput 2009;1(1):1–41.
[3] Apt KR. Principles of constraint programming. Cambridge University Press; 2003.
[4] Baptiste P, Pape CL, Nuijten W. Constraint-based scheduling: applying constraint programming to scheduling problems, vol. 39. Netherlands: Springer; 2001.
[5] Berthold T, Heinz S, Lübbecke ME, Möhring RH, Schulz J. A constraint integer programming approach for resource-constrained project scheduling. In: Lodi A, Milano M, Toth P, editors. Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. Lecture notes in computer science, vol. 6140. Springer Berlin Heidelberg; 2010. p. 313–17. ISBN 978-3-642-13519-4
[6] Blazewicz J, Lenstra JK, Kan AHGR. Scheduling subject to resource constraints: classification and complexity. Discrete Appl Math 1983;5(1):11–24.
[7] Brucker P, Knust S. Complex scheduling. Springer Verlag; 2006. ISBN 3540295453
[8] Coelho J, Vanhoucke M. Multi-mode resource-constrained project scheduling using RCPSP and SAT solvers. Eur J Oper Res 2011;213(1):73–82. http://dx.doi.org/10.1016/j.ejor.2011.03.019. ISSN 0377-2217, URL http://www.sciencedirect.com/science/article/pii/S037722171100230X
[9] . G12 constraint programming platform. G12 constraint programming platform; 2013a. http://nicta.com.au/research/projects/constraint_programming_platform
[10] Geiger MJ. A multi-threaded local search algorithm and computer implementation for the multi-mode, resource-constrained multi-project scheduling problem. Eur J Oper Res 2017;256(3):729–41. http://dx.doi.org/10.1016/j.ejor.2016.07.024. ISSN 0377-2217, URL http://www.sciencedirect.com/science/article/pii/S0377221716305616.
[11] Hartmann S. Project scheduling with multiple modes: a genetic algorithm. Ann Oper Res 2001;102(1–4):111–35. doi:10.1023/A:1010902015091. ISSN 0254-5330, URL http://dx.doi.org/10.1023/A.
[12] Hartmann S, Drexl A. Project scheduling with multiple modes: a comparison of exact algorithms. Networks 1998;32(4):283–97.
[13] Heilmann R. A branch-and-bound procedure for the multi-mode resource-constrained project scheduling problem with minimum and maximum time lags. Eur J Oper Res 2003;144(2):348–65.
[14] Heinz S, Ku W-Y, Beck JC. Recent improvements using constraint integer programming for resource allocation and scheduling. In: Gomes C, Sellmann M, editors. Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. Lecture notes in computer science, vol. 7874. Springer Berlin Heidelberg; 2013. p. 12–27.
[15] Horbach A. A boolean satisfiability approach to the resource-constrained project scheduling problem. Ann Oper Res 2010;181(1):89–107.
[16] JaCoP. JaCoP - Java constraint programming solver. 2014. http://jacop.osolpro.com.
[17] Kolisch R, Drexl A. Local search for nonpreemptive multi-mode resource-constrained project scheduling. IIE Trans 1997;29(11):987–99.
[18] Kolisch R, Sprecher A. PSPLIB-a project scheduling problem library. Eur J Oper Res 1997;96(1):205–16.
[19] Marques-Silva JP, Sakallah KA. GRASP: a search algorithm for propositional satisfiability. IEEE Trans Comput 1999;48(5):506–21.
[20] Marriott K, Nethercote N, Rafeh R, Stuckey PJ, De La Banda MG, Wallace M. The design of the zinc modelling language. Constraints 2008;13(3):229–67.
[21] Moskewicz MW, Madigan CF, Zhao Y, Zhang L, Malik S. Chaff: engineering an efficient SAT solver. In: Proceedings of the 38th annual design automation conference, DAC '01. New York, NY, USA: ACM; 2001. p. 530–5. doi:10.1145/378239.379017. ISBN 1-58113-297-2.
[22] NICTA research team. NICTA research team. 2013b. http://nicta.com.au/research.
[23] Ohrimenko O, Stuckey PJ, Codish M. Propagation via lazy clause generation. Constraints 2009;14(3):357–91. ISSN 1383-7133
[24] Online. The benchmark data set for the mrcpsp. 2016. www.mmlib.eu, Accessed:2016-09-28.
[25] Passmark CPU benchmark. Passmark CPU benchmark. 2014. https://www.cpubenchmark.net/cpu_list.php.
[26] Schnell A, Hartl RF. On the efficient modeling and solution of the multi-mode resource-constrained project scheduling problem with generalized precedence relations. OR Spectrum 2016;38(2):283–303. doi:10.1007/s00291-015-0419-6. ISSN 1436-6304.
[27] Schutt A. Improving scheduling by learning. Department of Computer Science and Software Engineering, The University of Melbourne; 2011. Phd thesis.
[28] Schutt A, Feydy T, Stuckey PJ, Wallace MG. Explaining the cumulative propagator. Constraints 2011;16(3):250–82. doi:10.1007/s10601-010-9103-2. ISSN 1383-7133.
[29] Schutt A, Chu G, Stuckey PJ, Wallace MG. Maximising the net present value for resource-constrained project scheduling. In: Beldiceanu N, Jussien N, Pinson E, editors. Integration of AI and OR techniques in contraint programming for combinatorial optimzation problems. Lecture notes in computer science, vol. 7298. Springer Berlin Heidelberg; 2012. p. 362–78. ISBN 978-3-642-29827-1.

[30] Schutt A, Feydy T, Stuckey PJ. Explaining time-table-edge-finding propagation for the cumulative resource constraint. In: Gomes C, Sellmann M, editors. Integration of AI and OR techniques in constraint programming for combinatorial optimization problems. Lecture notes in computer science, vol. 7874. Springer Berlin Heidelberg; 2013a. p. 234–50. ISBN 978-3-642-38170-6.
[31] Schutt A, Feydy T, Stuckey PJ, Wallace MG. Solving RCPSP/max by lazy clause generation. J Scheduling 2013b;16(3):273–89.
[32] Sprecher A, Drexl A. Multi-mode resource-constrained project scheduling by a simple, general and powerful sequencing algorithm. Eur J Oper Res 1998;107(2):431–50.
[33] Hentenryck PV, Carillon J-P. Generality versus specificity: an experience with AI and OR techniques.. In: Proceedings of the national conference on artificial intelligence (AAAI); 1988. p. 660–4.
[34] Peteghem VV, Vanhoucke M. Using resource scarceness characteristics to solve the multi-mode resource-constrained project scheduling problem. J Heuristics 2011;17(6):705–28. doi:10.1007/s10732-010-9152-0. ISSN 1381-1231.
[35] Peteghem VV, Vanhoucke M. An experimental investigation of metaheuristics for the multi-mode resource-constrained project scheduling problem on new dataset instances. Eur J Oper Res 2014;235(1):62–72. http://dx.doi.org/10.1016/j.ejor.2013.10.012. ISSN 0377-2217, URL http://www.sciencedirect.com/science/article/pii/S0377221713008357.
[36] Vanhoucke M, Coelho J, Batselier J. An overview of project data for integrated project management and control. J Modern Project Manage 2016;3(3). ISSN 2317-3963. URL http://journalmodernpm.com/index.php/jmpm/article/view/158.
[37] Zhu G, Bard JF, Yu G. A branch-and-cut procedure for the multi-mode resource-constrained project-scheduling problem. INFORMS J Comput 2006;18(3):377–90.