Recent Improvements Using Constraint Integer Programming for Resource Allocation and Scheduling

Stefan Heinz[1,⋆], Wen-Yang Ku[2], and J. Christopher Beck[2]

[1] Zuse Institute Berlin, Takustr. 7, 14195 Berlin, Germany
heinz@zib.de
[2] Department of Mechanical & Industrial Engineering
University of Toronto, Toronto, Ontario M5S 3G8, Canada
{wku,jcb}@mie.utoronto.ca

Abstract. Recently, we compared the performance of mixed-integer programming (MIP), constraint programming (CP), and constraint integer programming (CIP) to a state-of-the-art logic-based Benders manual decomposition (LBBD) for a resource allocation/scheduling problem. For a simple linear relaxation, the LBBD and CIP models deliver comparable performance with MIP also performing well. Here we show that algorithmic developments in CIP plus the use of an existing tighter relaxation substantially improve one of the CIP approaches. Furthermore, the use of the same relaxation in LBBD and MIP models significantly improves their performance. While such a result is known for LBBD, to the best of our knowledge, the other results are novel. Our experiments show that both CIP and MIP approaches are competitive with LBBD in terms of the number of problems solved to proven optimality, though MIP is about three times slower on average. Further, unlike the LBBD and CIP approaches, the MIP model is able to obtain provably high-quality solutions for all problem instances.

1 Introduction

In previous work, we provided empirical evidence showing that models based on mixed-integer programming (MIP) and constraint integer program (CIP) were competitive with logic-based Benders decomposition (LBBD) for a class of resource allocation and scheduling problems [1]. A weakness in this work was that we were not able to achieve the same performance with LBBD as in previous work (e.g., [2,3]), which made our conclusions necessarily conservative. In this paper we show that equivalent performance of LBBD to that in the literature can be obtained by using a stronger sub-problem relaxation, strengthening the Benders cuts, and employing a commercial CP solver for the sub-problems. None of these results come as a surprise as they already exist in the literature [2], but

⋆ Supported by the DFG Research Center Matheon *Mathematics for key technologies* in Berlin.

C. Gomes and M. Sellmann (Eds.): CPAIOR 2013, LNCS 7874, pp. 12–27, 2013.

(re)establishing these results replicates the literature, as well as allowing our conclusions with respect to other approaches to be placed on a firmer foundation.

More interestingly, we demonstrate that:

- The further integration of global constraint-based reasoning within the CIP framework, combined with the same stronger relaxation, results in a CIP model with equivalent performance to the LBBD model.
- The existing MIP model can itself be augmented with the stronger relaxation and this extension leads to substantially improved performance to the point MIP is competitive with the improved LBBD and CIP model.

Our experimental investigations show that the instances in one problem set (with unary resources) are now all easily solved by all models in a few seconds. For the more challenging set of instances with non-unary resources, LBBD and one of our CIP models (CIP[CP]) achieve essentially equivalent performance in terms of solving problems to optimality and finding the best known solutions, while CIP[CP] is able to find provably high quality solutions on more problem instances. The extended MIP model and the CIP model based on the MIP formulation (CIP[MIP]) find slightly fewer optimal solutions and require more run-time to do so. However, unlike both LBBD and the CIP[CP] model, the MIP and CIP[MIP] models are able to find solutions with a small optimality gap for *all* instances. Furthermore, the time for the MIP model to find a first feasible solution is twenty times faster than LBBD and twice as fast as CIP[CP] in geometric mean.

Based on our results, declaring a single winner among these three approaches is therefore fraught and perhaps not of fundamental interest (see [4]). However, our results with extended models reinforce our previous conclusions [1]: both CIP and MIP are competitive with LBBD for these scheduling problems and should be considered as core technologies for more general scheduling problems.

The rest of the paper is organized as follows. In Section 2, we formally define our problem. Section 3 presents the necessary background, including a discussion of logic-based Benders decomposition and a short summary of the results from our previous paper. In Section 4, we present the models used in this paper and we discuss detailed results of our experiments in Section 5. Section 6 provides a discussion of our results and we then conclude in Section 7.

2 Problem Definition

We study two scheduling problems referred to as UNARY and MULTI [5,3,1], which are defined by a set of jobs $\mathcal{J}$ and a set of resources $\mathcal{K}$. Each job j must be assigned to a resource k and scheduled to start at or after its release date, $\mathcal{R}_j$, end at or before its deadline, $\mathcal{D}_j$, and execute for p_{jk} consecutive time units. Each job also has a resource assignment cost c_{jk} and a resource requirement r_{jk}. We denote $\mathcal{R}$ as the set of all release dates and $\mathcal{D}$ as the set of all deadlines. Each resource $k \in \mathcal{K}$ has a capacity C_k and a corresponding constraint which states that the resource capacity must not be exceeded at any time. In the UNARY

problems, each resource has unary capacity and all jobs require only one unit of resource. For the MULTI problems, capacities and requirements may be non-unary. A feasible solution is an assignment where each job is placed on exactly one resource and a start time is assigned to each job such that no resource exceeds its capacity at any time point. The goal is to find an optimal solution, that is, a feasible solution which minimizes the total resource assignment cost.

3 Background

In this section we give the necessary background w.r.t. the logic-based Benders decomposition and revisit our previous results.

3.1 Logic-Based Benders Decomposition

Logic-based Benders decomposition (LBBD) is a problem decomposition technique that generalizes Benders decomposition [6,7]. Conceptually, some of the variables and constraints of a global problem model are removed, creating a master problem (MP) whose solution (in the case of minimization) forms a lower-bound on the globally optimal solution. The extracted problem components form one or more sub-problems (SPs) where each SP is an inference dual [6]. Based on an MP solution, each sub-problem is solved, deriving the tightest bound on the MP cost function that can be inferred from the current MP solution and the constraints and variables of the SP. If a bound produced by an SP is not satisfied by the MP solution, a *Benders cut* is introduced to the MP. For global convergence, the cut must remove the current MP solution from the feasibility space of the MP without removing any globally optimal solutions.

The standard solution procedure for an LBBD model is to iteratively solve the MP to optimality, solve each sub-problem, add the Benders cuts, and re-solve the MP. Iterations continue until all SPs are satisfied by the MP solution, which has thereby been proved to be globally optimal.

For the resource allocation and scheduling problem, since the MP assigns each job to a resource and there are no inter-job constraints, the SPs are independent, single-machine feasibility scheduling problems. If all SPs are feasible, then the assignment found by the MP is valid for all resources and the corresponding cost is the global minimum. Otherwise, each infeasible SP generate a Benders cut involving a set of jobs that cannot be feasibly scheduled.

Experience with LBBD models has shown that two aspects of the formulation are critical for achieving good performance: the inclusion of a relaxation of each SP in the MP and a strong, but easily calculated Benders cut [8].

The Sub-Problem Relaxation. For the problem studied here two relaxations have been proposed. Here we label them as the *single* relaxation [9] and the *interval* relaxation [2]. The former consists of one linear constraint per SP representing to total area (i.e., time by capacity) available on that resource. Formally, the relaxation can be formulated as follows:

$$\sum_{j \in \mathcal{J}} p_{jk} r_{jk} \, x_{jk} \leq C_k \cdot (\max_{j \in \mathcal{J}}\{\mathcal{D}_j\} - \min_{j \in \mathcal{J}}\{\mathcal{R}_j\}) \qquad \forall k \in \mathcal{K} \qquad (1)$$

where x_{jk} is the binary resource choice variable equal to 1 if and only if job j is assigned to resource k.

The interval relaxation consists of $O(|\mathcal{J}|^2)$ linear constraints per SP representing the total area of a number of overlapping intervals and sets of jobs. The interval relaxation is formulated as follows:

$$\sum_{j \in \mathcal{J}(t_1,t_2)} p_{jk} r_{jk} \, x_{jk} \leq C_k \cdot (t_2 - t_1) \qquad \forall k \in \mathcal{K}, \ \forall (t_1, t_2) \in \mathcal{E} \qquad (2)$$

where $\mathcal{E} = \{(t_1, t_2) \mid t_1 \in \mathcal{R}, t_2 \in \mathcal{D}, t_1 < t_2\}$. We denote with $\mathcal{J}(t_1, t_2)$ the set of jobs that execute between t_1 and t_2: $\mathcal{J}(t_1, t_2) = \{j \in \mathcal{J} \mid t_1 \leq \mathcal{R}_j, t_2 \geq \mathcal{D}_j\}$.

If all jobs have the same time window the interval relaxation collapses to the single relaxation.

The Benders Cut. Given that the SPs are feasibility problems without any visibility to the global optimization function, the only possible Benders cut is a no-good constraint preventing the same set of jobs from being assigned to the resource again. Therefore, the cut will take the form of Constraint (11) in Model 3. Note that $\mathcal{J}_{hk}$ is a set of jobs that cannot be feasibly scheduled together on resource k. A strengthened cut can be produced by finding a subset of $\mathcal{J}_{hk}$ that also cannot be feasibly scheduled on resource k. Hooker [2] suggests a greedy procedure to find a minimal infeasible set by removing each job, one by one, from $\mathcal{J}_{hk}$ and resolving the SP. If the SP is still infeasible the corresponding job can be removed from the infeasible set, otherwise it is replaced in the set and the greedy procedure continues.

3.2 Previous Results

Our previous work compared five models: constraint programming (CP), mixed-integer programming (MIP), logic-based Benders decomposition (LBBD) and two constraint integer programming models (CIP[CP] and CIP[MIP]) [1]. The LBBD model used the single relaxation and the non-strengthened cut. While the need for a cut is unique to LBBD, it may be possible and useful to include the problem relaxation in any model that makes use of an linear programming relaxation. In particular, to be consistent with the LBBD formulation, in our previous work we used the single relaxation in CIP[CP] tested. The MIP and CIP[MIP] models were based on a different formulation so it was not obvious how to incorporate this type of relaxation.[1]

Table 1a (Section 5) reproduces the summary of our previous results [1], omitting the CP results as they are not extended here. Based on these results,

[1] Below we show how to do this.

we concluded that both CIP and MIP technologies are at the least competitive with LBBD as a state-of-the-art technique for these problems. One caveat to this conclusion (noted in Heinz & Beck [1]) was that previous work had achieved stronger results for LBBD [2,3]. However, even taking into account those stronger results, both MIP and CIP models found provably high-quality feasible solutions for all instances while LBBD does not.

4 Model and Solver Extensions

The primary contributions of this paper are:

- Implementation of new presolving, propagation, and primal heuristics in the SCIP solver and their application to the resource allocation/scheduling problems.
- Extension of all models to include the interval relaxation. As noted, this extension is not new for LBBD.
- Replication of previous results using LBBD with the interval relaxation and strengthened cuts.

In this section, we present the extensions to the CIP solving techniques and the mathematical models used in our experiments.

4.1 Constraint Integer Programming

In our previous work [1], we presented the first integration of the `optcumulative` global constraint, a cumulative resource constraint with optional activities, into the paradigm of CIP [10,11]. We focused mainly on its linear relaxation and incorporated a straightforward propagation via the cumulative constraints.

Here we continue the integration of the `optcumulative` global constraint into a CIP framework, including the addition of presolving techniques, general purpose primal heuristics focusing in the clique structure, and the interval relaxation. All of these techniques have been previously presented in the literature, separately, and not all in the context of CIP. Therefore the techniques, in themselves, do not represent a contribution of this paper. Rather, our contributions here are the integration of these techniques in CIP and the demonstration that their combination leads to state-of-the-art performance.

The Integration of `optcumulative`. In this section, we discuss the integration of the `optcumulative` into the CIP framework via presolving, propagation, conflict analysis, linear relaxation, and primal heuristics.

Presolving. Before the tree search starts, presolving detects and removes redundant constraints and variables. In case of the `optcumulative`, one shrink the time windows of each job and remove *irrelevant* jobs from the scope of the constraint since this leads to potentially tighter linear relaxation (see Equation (2)). In particular, we have developed *dual reduction techniques* that are able to remove

$$
\begin{aligned}
&\min \sum_{k\in\mathcal{K}} \sum_{j\in\mathcal{J}} c_{jk}\, x_{jk} && && \\
&\text{s.t.} \sum_{k\in\mathcal{K}} x_{jk} = 1 && \forall j \in \mathcal{J} && (3)\\
&\quad \texttt{optcumulative}(\boldsymbol{S_{\cdot k}}, \boldsymbol{x_{\cdot k}}, \boldsymbol{p_{\cdot k}}, \boldsymbol{r_{\cdot k}}, C_k) && \forall k \in \mathcal{K} && \\
&\quad \sum_{j\in\mathcal{J}(t_1,t_2)} p_{jk} r_{jk}\, x_{jk} \le C_k \cdot (t_2 - t_1) && \forall k \in \mathcal{K},\ \forall (t_1,t_2) \in \mathcal{E} && (4)\\
&\quad \mathcal{R}_j \le S_{jk} \le \mathcal{D}_j - p_{jk} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} && \\
&\quad x_{jk} \in \{0,1\} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} && \\
&\quad S_{jk} \in \mathbb{Z} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} &&
\end{aligned}
$$

Model 1. A CIP model extending CIP[CP] [1]

redundant jobs [12] from the cumulative constraints. We apply these reductions to the `optcumulative` constraint by assuming that all potentially scheduled jobs are assigned to a resource. Due to the monotonicity of the inference performed, any redundant jobs detected under the all-jobs assumption remain redundant when a subset of jobs is assigned to a resource. Additionally, we can detect a redundant `optcumulative` constraint by assuming all possible jobs are assigned to the resource and checking if the resultant cumulative constraint has a feasible solution. If so, the corresponding constraint can be removed from the problem formulation because a feasible schedule exists with all possible jobs. These inferences are specializations of the existing general dual inference techniques [12].

Propagation. During the tree search, we collect jobs which are assigned to a resource and apply the cumulative propagator [13,14]. For the remaining jobs, we run singleton arc consistency to detect jobs which can no longer be feasibly scheduled and fix the corresponding binary choice variable to zero. The extension here is that if all resource assignment variables for a given resource are fixed, we try to solve the remaining individual cumulative constraint by itself, triggering a backtrack if no such solution exists. The same data structure used in presolving [12], can be used to perform this detection in a sound and general manner. In contrast to LBBD, these (indirect) sub-problems do not need to be solved. If a solution is found or the problem is proved infeasible, the global search space is reduced. However, if they are not solved, the main search continues.

Conflict analysis. We use the explanation algorithms corresponding to the cumulative propagator [15,16] and extend the generated explanations to include only the binary resource choice variables for those start time variables which are part of the explanation. This is different from our previous implementation where we included all binary variables in the conflict. Adding only the binary variables which are part of the cumulative explanation is analogous to the strengthening techniques of the Benders cuts described above.

$$\min \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{J}} c_{jk}\, x_{jk}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} x_{jk} = 1 \qquad \forall j \in \mathcal{J} \tag{5}$$

$$\sum_{t=\mathcal{R}_j}^{\mathcal{D}_j - p_{jk}} y_{kjt} = x_{jk} \qquad \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} \tag{6}$$

$$\sum_{j \in \mathcal{J}} \sum_{t' \in T_{jkt}} r_{jk}\, y_{jkt'} \leq C_k \qquad \forall k \in \mathcal{K},\ \forall t \tag{7}$$

$$\mathcal{R}_j + \sum_{t=\mathcal{R}_j}^{\mathcal{D}_j - p_j} (t - \mathcal{R}_j) \cdot y_{jkt} = S_{jk} \qquad \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} \tag{8}$$

$$\texttt{optcumulative}(\boldsymbol{S_{\cdot k}}, \boldsymbol{x_{\cdot k}}, \boldsymbol{p_{\cdot k}}, \boldsymbol{r_{\cdot k}}, C_k) \qquad \forall k \in \mathcal{K}$$

$$\sum_{j \in \mathcal{J}(t_1, t_2)} p_{jk} r_{jk}\, x_{jk} \leq C_k \cdot (t_2 - t_1) \qquad \forall k \in \mathcal{K},\ \forall (t_1, t_2) \in \mathcal{E} \tag{9}$$

$$\mathcal{R}_j \leq S_{jk} \leq \mathcal{D}_j - p_{jk} \qquad \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}$$

$$x_{jk} \in \{0, 1\} \qquad \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}$$

$$S_{jk} \in \mathbb{Z} \qquad \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}$$

$$y_{jkt} \in \{0, 1\} \qquad \forall j \in \mathcal{J},\ \forall k \in \mathcal{K},\ \forall t \in \{\mathcal{R}_j, \ldots, \mathcal{D}_j - p_{jk}\}$$

Model 2. CIP[MIP]: A CIP model based on the MIP model with channeling Constraints (8). $T_{jkt} = \{t - p_{jk}, \ldots, t\}$.

Linear relaxation. As discussed above, we use the interval relaxation (Equation (2)) instead of the single relaxation (Equation (1)) in our CIP models. See below for the details of the CIP[CP] and CIP[MIP] models. To generate the relaxation we use the algorithm presented by Hooker [2] to impose only non-redundant constraints.

Primal heuristic. Inspired by the clique structure of the problem (i.e., each job has to be assigned to one resource), we implemented a general purpose primal heuristic that assigns jobs to resources and solves the resulting decomposed scheduling problems. In MIP and CIP, a clique structure refers to a sets of binary variables that must sum to at most one. This structure is easily detectable within a model and can be used within a diving heuristic.

Extended Models. In this section, we present the full CIP models, one based on the CP formulation (CIP[CP]) and the other based on the MIP formulation (CIP[MIP]). Both are extensions of correspondingly named existing models [1].

Model 1 presents the CIP[CP] model with the resource choice variable x_{jk} equal to 1 if and only if job j is assigned to resource k. The objective function is defined in terms of the resource choice variables. Constraints (3) ensure that each job is assigned to exactly one resource, where the resource capacities are enforced by the global `optcumulative` constraints. Constraints (4) state the interval relaxation. This model is equivalent to the existing CIP[CP] model [1]

$$
\begin{aligned}
\text{(MP)} \quad & \min \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{J}} c_{jk}\, x_{jk} \\
& \text{s.t.} \sum_{k \in \mathcal{K}} x_{jk} = 1 && \forall j \in \mathcal{J} \\
& \sum_{j \in \mathcal{J}(t_1,t_2)} p_{jk} r_{jk}\, x_{jk} \leq C_k \cdot (t_2 - t_1) && \forall k \in \mathcal{K},\ \forall (t_1,t_2) \in \mathcal{E} \qquad (10) \\
& \sum_{j \in \mathcal{J}_{hk}} (1 - x_{jk}) \geq 1 && \forall k \in \mathcal{K},\ \forall h \in \{1,\ldots,H-1\} \qquad (11) \\
& x_{kj} \in \{0,1\} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}
\end{aligned}
$$

$$
\begin{aligned}
\text{(SP)} \quad & \texttt{cumulative}(\boldsymbol{S}, \boldsymbol{p_{\cdot k}}, \boldsymbol{r_{\cdot k}}, C_k) \\
& \mathcal{R}_j \leq S_j \leq \mathcal{D}_j - p_{jk} && \forall j \in \mathcal{J}_k \\
& S_j \in \mathbb{Z} && \forall j \in \mathcal{J}_k
\end{aligned}
$$

Model 3. Logic-based Benders decomposition: master problem (MP) on top and subproblem (SP) for resource k below

except for Constraints (4). The interval relaxation is added in the same way as it is included in the LBBD model (see Hooker [2] and Model 3).

For the CIP model based on the MIP formulation (CIP[MIP]–Model 2), we incorporate the interval relaxation by Constraint (9). The primary decision variables of the time-indexed formulation, y_{kjt}, are equal to 1 if and only if job j starts on resource k at time point t. We add an auxiliary set of binary decision variables, x_{jk}, which are assigned to 1 if and only if job j is assigned to resource k.

The objective function is defined with the new set of binary decision variables. Constraints (5) ensure that each job is assigned to exactly one resource. The two sets of decision variables are linked via Constraints (6). The resource capacities are enforced by the knapsack constraints (7) which are given for each time point. The global `optcumulative` constraints are added to achieve additional propagation. Finally, the Constraints (9) state the (redundant) interval relaxation which potentially strengthens the linear programming relaxation.

4.2 Logic-Based Benders Decomposition

We use the LBBD model from Hooker [2] which uses both the interval relaxation and the strengthened cuts. For completeness, we present the model in Model 3.

4.3 Mixed Integer Programming

Given the presence of cumulative constraints in the MULTI version of the problem, the standard MIP model uses a *time-indexed* formulation [8,9,1] employing a set of binary decision variables, y_{jkt}, which are equal to 1 if and only if job j starts

$$\min \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{J}} c_{jk}\, x_{jk}$$

$$\text{s.t.} \sum_{k \in \mathcal{K}} x_{jk} = 1 \qquad \forall j \in \mathcal{J} \tag{12}$$

$$\sum_{t=\mathcal{R}_j}^{\mathcal{D}_j - p_{jk}} y_{kjt} = x_{jk} \qquad \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} \tag{13}$$

$$\sum_{j \in \mathcal{J}} \sum_{t' \in T_{jkt}} r_{jk}\, y_{jkt'} \leq C_k \qquad \forall k \in \mathcal{K},\ \forall t \tag{14}$$

$$\sum_{j \in \mathcal{J}(t_1,t_2)} p_{jk} r_{jk}\, x_{jk} \leq C_k \cdot (t_2 - t_1) \qquad \forall k \in \mathcal{K},\ \forall (t_1,t_2) \in \mathcal{E} \tag{15}$$

$$x_{jk} \in \{0,1\} \qquad \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}$$

$$y_{jkt} \in \{0,1\} \qquad \forall j \in \mathcal{J},\ \forall k \in \mathcal{K},\ \forall t \in \{\mathcal{R}_j, \ldots, \mathcal{D}_j - p_{jk}\}$$

Model 4. Mixed integer programming model with $T_{jkt} = \{t - p_{jk}, \ldots, t\}$

at time t on resource k. As with the CIP[MIP] model above, we extend the MIP model to include a second set of binary variables, x_{kj}, which are equal to 1 if and only if job j is assigned to resource k. This second set of variables introduces the decomposition aspect of the problem into the MIP model since the cost for an assignment is determined only by this set of variables. In addition, these variables make it natural to express the interval relaxation.

Our MIP model is stated in Model 4. The constraints are almost identical to those presented above in the CIP[MIP] model, with the exception that the start time variables, the global `optcumulative` constraints, and the necessary channeling constraints are absent.

Note that the second set of decision variables is redundant. Our preliminary experiments showed that the solver achieves much higher performance with the redundant formulation. The interval relaxation itself is also redundant given Constraints (14). However, they introduce a connection between the capacity constraints of each resource, strengthening the linear programming relaxation.

5 Computational Results

In this section, we compare the performance of the LBBD, the MIP, and the two CIP models. We use the same test sets and the same computational environment as our previous work [1] to allow direct comparison of the results.

5.1 Experimental Setup

Test Sets. The problem instances were introduced by Hooker [5]. Each set contains 195 problem instances with the number of resources ranging from two to four and the number of jobs from 10 to 38 in steps of two. The maximum number of jobs for the instances with three and four resources is 32 while for

Table 1. Summary of the results presented in Heinz & Beck [1] and the results for the extended model of this paper

(a) Results of Heinz & Beck [1], omitting the CP model.

	UNARY				MULTI			
	MIP	LBBD	CIP[CP]	CIP[MIP]	MIP	LBBD	CIP[CP]	CIP[MIP]
feasible	**195**	175	**195**	**195**	**195**	119	125	**195**
optimal found	**195**	175	194	**195**	**148**	119	124	142
optimal proved	191	175	194	**195**	109	119	123	**133**
best known found	**195**	175	194	**195**	**155**	119	124	146
total time	12	28	**10**	19	442	228	**212**	395
time to best	**7**	28	9	17	209	228	**200**	217

(b) Results for the extended models.

	UNARY				MULTI			
	MIP	LBBD	CIP[CP]	CIP[MIP]	MIP	LBBD	CIP[CP]	CIP[MIP]
feasible	**195**	**195**	**195**	**195**	**195**	174	190	**195**
optimal found	**195**	**195**	**195**	**195**	167	**174**	167	142
optimal proved	**195**	**195**	**195**	**195**	155	**174**	163	126
best known found	**195**	**195**	**195**	**195**	172	**174**	168	146
total time	1.7	**1.0**	1.3	10.4	159.6	**37.8**	54.3	383.3
time to best	1.6	**1.0**	1.3	9.8	121.5	37.8	**6.3**	198.4
time to first	1.3	**1.0**	**1.0**	1.8	**2.4**	37.8	5.0	18.7

two resources the number of maximum number of jobs is 38. In addition, there are five instances for each problem size. For the MULTI problems, the resource capacity is 10 and the job demands are generated with uniform probability on the integer interval $[1, 9]$. See Hooker [5] for further details w.r.t. the generation of instances, and the appendix of [17] for further problem instance characteristics.

Computational Environment. All experiments are performed on Intel Xeon E5420 2.50 GHz computers (in 64 bit mode) with 6 MB cache and 6 GB of main memory, running Linux. For solving the MIP models we used IBM ILOG CPLEX 12.4 in its default setting. The master problem of the LBBD approach is solved with SCIP 3.0.0 [11] using SoPlex [18] version 1.7.0.1 as the linear programming solver. For the sub-problems we used IBM ILOG CP Optimizer 12.4 using the default settings plus extended filtering and depth first search. For the CIP models we used the same solver as for the master problem of the LBBD. For each instance we enforced a time limit of 2 hours and allow for a single thread.

5.2 Previous Results

Table 1a presents the summary of the previous results [1], omitting the pure CP model which we do not extend here. For each test set (UNARY and MULTI) and each model, Table 1a states the number of instances for which (i) a feasible solution was found, (ii) an optimal solution was found, (iii) an optimal was found and proved, and (iv) the best known solution was found. Secondly we present

the shifted geometric mean[2] for the total running time and the time until the best solution was found. The time to the best solution is only an upper bound in case of CPLEX since the output log does not display this time point explicitly.

5.3 Results

In the same fashion as in Table 1a, we summarize the results for the extended models in Table 1b. Additionally, we included the shifted geometric mean for the time when the first feasible solution was found.

The UNARY Test Set. The results for the UNARY test set show that all models improved drastically w.r.t. our previous results. All approaches are able to solve all instances in a few seconds or less. While this was expected for LBBD and the CIP models, it comes as a bit of a surprise for the MIP model. Analyzing the results for LBBD, we see that it needs only one iteration for each instance: the first optimal master solution is always proved feasible for all sub-problems. This result indicates that the interval relaxation tightens the master problem significantly. Since the MIP and CIP models have basically the same linear relaxation as LBBD, we believe that the tightness of the interval relaxation explains the improved results for these models as well.

The MULTI Test Set. For the MULTI test set, we observe a substantial improvement for all models except CIP[MIP]. The MIP models solves 155 instances of 195 compared to 109, the LBBD approach proves optimality for 174 instances compared to 119, and the CIP[CP] formulation handles 163 instances compared to 123 before. Similar observations can be made for the running times: LBBD, MIP, and CIP[CP] are now a factor six, three, and four faster than before, respectively. In terms of relative speed, the close-to-uniform speed-ups results in basically the same ratios among the different models as in Beck & Heinz. Only the CIP[MIP] performance remained consistent, while all other models improved in a similar way.

As in our previous results, the MIP and CIP[MIP] models are able to find a feasible solution for all instances. The CIP[CP] does that for 190 instances compared to 125 instances before, while LBBD is only able to find feasible solutions for the problems that it solves to optimality. Comparing the quality of the solutions among the solvers, we observe that the MIP model finds the optimal or best known solutions for 172 instances.[3] By the same metric LBBD and CIP[CP] find best known solutions for 174 and 168 instances, respectively. If the best of the four models is chosen for each instance (resulting in the *virtual best solver*), 187 instances can be solved to proven optimality with a total running time in shifted geometric mean of 19.9 seconds.

For a more detailed indication of the results for the MULTI test set, Table 2 presents results for each problem size for the CIP, MIP, and LBBD models. The

[2] The shifted geometric mean of values $t_1, \ldots, t_n$ is $\left(\prod(t_i + s)\right)^{1/n} - s$, with shift $s = 10$.

[3] For the MULTI test set the optimal solution value is known for 189 instances.

Table 2. Detailed results for the MULTI test set. For each resource job combination consists of 5 instances (for a total of 195) we display on line.

		MIP				LBBD				CIP[CP]				CIP[MIP]			
$\|\mathcal{K}\|$	$\|\mathcal{J}\|$	opt	feas	arith	geom	opt	feas	arith	geom	opt	feas	arith	geom	opt	feas	arith	geom
2	10	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.9	1.8
	12	5	5	1.1	1.1	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	3.7	3.5
	14	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.0	1.0	5	5	4.9	4.7
	16	5	5	13.1	8.0	5	5	1.0	**1.0**	5	5	8.3	4.7	5	5	40.3	29.9
	18	5	5	36.2	16.9	5	5	1.3	**1.3**	5	5	1.8	1.7	5	5	75.4	66.2
	20	5	5	89.6	29.0	5	5	4.1	3.7	5	5	1.6	**1.5**	5	5	120.2	70.4
	22	4	5	2983.3	812.4	5	5	796.8	**51.4**	3	5	3090.8	382.5	4	5	3036.0	1293.9
	24	3	5	3026.7	883.0	4	4	1733.8	**214.8**	2	5	4321.5	573.4	2	5	4399.3	1508.8
	26	4	5	3013.5	1069.2	5	5	912.1	**209.0**	4	4	2122.4	464.9	3	5	3414.9	1746.9
	28	4	5	2394.7	378.9	5	5	993.7	536.5	4	5	1444.4	**42.0**	3	5	3822.6	1910.5
	30	3	5	3788.2	861.2	3	3	2930.3	**401.2**	2	5	4321.8	587.6	1	5	5802.8	3590.5
	32	3	5	3054.7	**792.1**	0	0	–	–	2	4	4400.1	1140.5	0	5	–	–
	34	3	5	3444.0	**879.7**	2	2	4400.3	1745.1	1	3	5760.4	1995.3	1	5	5843.7	4089.2
	36	2	5	4386.6	1534.1	1	1	5942.7	4770.2	3	4	3476.7	**548.4**	2	5	4709.5	2319.1
	38	2	5	5590.6	4980.2	1	1	6268.8	5848.7	2	5	4360.6	**1334.0**	0	5	–	–
3	10	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.4	1.3
	12	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.0	1.0	5	5	4.5	4.3
	14	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.2	1.2	5	5	11.3	10.3
	16	5	5	4.5	4.2	5	5	3.3	**3.0**	5	5	3.4	3.3	5	5	43.6	33.1
	18	5	5	76.4	46.0	5	5	7.1	5.8	5	5	5.0	**4.8**	5	5	594.9	244.0
	20	4	5	1470.9	98.5	5	5	1.5	**1.5**	5	5	7.8	6.9	4	5	1622.8	355.1
	22	4	5	1832.6	554.6	5	5	2.4	**2.3**	5	5	6.9	6.6	4	5	2543.7	1008.3
	24	5	5	1703.2	304.5	5	5	9.3	**6.7**	5	5	346.6	78.6	4	5	2225.2	967.5
	26	3	5	3826.9	1652.8	5	5	31.8	**19.8**	5	5	98.4	40.2	1	5	5942.0	4766.2
	28	3	5	3901.1	987.6	5	5	85.3	**35.4**	3	5	2885.5	194.9	2	5	4782.6	3030.9
	30	3	5	4028.1	3100.2	4	4	1523.6	**178.3**	4	5	1911.1	520.9	0	5	–	–
	32	2	5	4840.8	3601.3	4	4	2882.6	1951.8	3	5	2969.5	**559.0**	1	5	6125.7	5475.9
4	10	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.0	**1.0**
	12	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	1.0	**1.0**	5	5	2.4	2.3
	14	5	5	1.0	**1.0**	5	5	1.6	1.6	5	5	1.7	1.7	5	5	12.8	11.7
	16	5	5	1.2	1.2	5	5	1.0	**1.0**	5	5	1.7	1.7	5	5	22.3	20.4
	18	5	5	3.3	3.1	5	5	2.5	**2.5**	5	5	4.5	4.4	5	5	96.5	76.6
	20	5	5	43.8	25.3	5	5	1.8	**1.8**	5	5	4.4	4.3	5	5	159.3	116.0
	22	5	5	128.2	60.0	5	5	4.3	**3.7**	5	5	20.7	15.0	5	5	1244.4	482.9
	24	4	5	2695.6	1399.0	5	5	16.0	**12.1**	5	5	59.3	42.9	1	5	6236.7	5773.1
	26	3	5	3825.4	2787.8	5	5	15.7	**14.9**	5	5	293.0	112.7	1	5	6227.4	5750.3
	28	3	5	5361.9	2124.2	5	5	9.9	**9.6**	4	5	1562.9	200.0	2	5	5010.5	3646.9
	30	2	5	5035.9	3253.6	5	5	112.6	**31.7**	4	5	2243.0	581.1	0	5	–	–
	32	1	5	5927.9	4691.0	5	5	343.6	**118.3**	2	5	4412.3	1519.1	0	5	–	–
		155	195	1962.5	159.6	174	174	929.5	**37.8**	163	190	1286.1	54.3	126	195	2825.3	383.3

first two columns define the instance size in terms of the number of resources $|\mathcal{K}|$ and the number of jobs $|\mathcal{J}|$. For each model, we report the number of instances solved to proven optimality "opt" and the number instances for which a feasible solution was found, "feas", including the instances which are solved to optimality. For the total running time we report the arithmetic mean ("arith") and the shifted geometric mean ("geom") with shift $s = 10$. All running times that are less than 1.0 second are set to 1.0. For each resource-job combination, the best time is shown in bold. For clarity, when a model did not solve any instances of a given size, we use '–' instead of 7200 for the running time.

The table indicates that all models appear to scale exponentially with the number of jobs. The results for LBBD and CIP[CP] show the increase at a lower rate than for MIP and CIP[MIP]. Nonetheless, LBBD and CIP[CP] both fail to find and prove optimal solutions on some of the larger instances. It is interesting to note that for the instances with two resources, all models suddenly start to

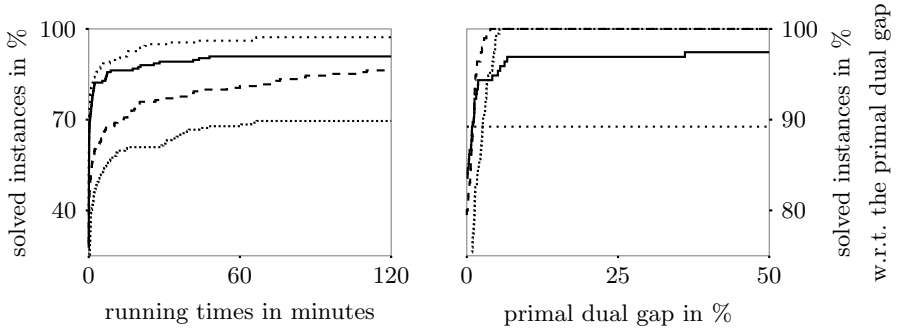

Fig. 5. Performance diagrams for the MULTI test set. The MIP model is dashed (- - -), the LBBD model dotted (.....), the CIP[CP] model solid (——), and the CIP[MIP] model is densely dotted (.....).

struggle with 22 or more jobs: the shifted geometric means of the run-time for all models increase one or two orders of magnitude in moving from 20 to 22 jobs. We return to this observation below.

Since the models do not solve or fail to solve exactly the same instances, we depict two performance diagrams for the MULTI test set in Figure 5. The left-hand graph shows the evolution of the number of problems solved to optimality over time. It can be observed that LBBD and CIP behave very similarly while MIP performs worse in the beginning but increases its success with more run-time. CIP[MIP] performs consistently worse than all other models. The right-hand graph displays the percentage of instances for which a solution with given optimality gap (primal bound minus dual bound divided by the primal bound) or better was found. On this basis, both MIP and CIP[MIP] models outperform the other two models by finding solutions with an optimality gap of less than 5% for all problem instances. CIP finds solutions with a gap of 10% or better on about 97% of the instances while LBBD finds the optimal solution on 89% of the instances and is, of course, unable to find any sub-optimal feasible solutions.

6 Discussion

The results of the experiments presented above support and reinforce the conclusions of Heinz & Beck [1]: both CIP[CP] and MIP should be considered to be state-of-art models, along with LBBD, for the tested resource allocation and scheduling problems. On the basis of the number of problem instances solved to optimality LBBD has a marginal advantage over CIP[CP] which itself is marginally better then MIP. However, on other measures of solution quality (number of instances with feasible solutions and the quality of those solutions), the ranking is reversed.

An examination of the sub-problems in the two-resource instances that LBBD and CIP[CP] fail to solve reveals that most of the cumulative constraint/sub-problems which have to be proven to be feasible or infeasible have a very small

slack and the jobs have a wide and often identical time windows. Slack is the difference between the rectangular area available on the resource (time by capacity) and the sum of the areas (processing time by resource requirement) of the jobs. Alternatively, slack can be understood to be the tightness of the single relaxation (Equation (1.)). The small slack results from the fact that one resource is consistently less costly than the others and so it appears promising to assign many jobs as possible within the limits of the interval relaxation.

All approaches suffer from not being able to handle small slack and wide time window problems efficiently on cumulative resources. This is the underlying reason for the main disadvantage of LBBD which gets stuck at such a sub-problems and fails completely to find a feasible solution. All other approaches have the same issue of not been able to solve these implicit sub-problems, but are able to provide high quality primal solutions. To overcome this issue, stronger cumulative inference techniques [19] may be worth consideration.

As we are comparing the CIP approach against a start-of-the-art LBBD implementation, we should also compare with a state-of-the-art commercial MIP solver when solving MIP models. It has, however, been standard for the past few years for commercial MIP solvers to use multiple cores. If we run IBM ILOG CPLEX with its default settings (using all available cores, eight in our case) on the MULTI instances we can solve 171 instances to proven optimality with a shifted geometric mean of 71.4 seconds. The fact that this performance is only marginally better than what we observe for CIP and similar to LBBD results, strengthens their claims to state-of-the-art status.

Finally, the results of the single-core virtual best solver, solving 187 instances with a shifted geometric mean of 19.9 second, indicate that none of the models is dominant. One of the arguments for pursuing CIP is that it is a framework that strives to combine the advantages of the other approaches in order to overcome the individual disadvantages.

Future Work. There are a number of areas of future work both on extending these approaches to related scheduling problems and in developing the technology of CIP for scheduling.

Continued development of CIP. We have demonstrated through the integration of the `optcumulative` constraint that global constraint-based presolving, inferences, and relaxations can lead to state-of-the-art performance. We intend to further pursue the integration of global constraint reasoning into a CIP framework for scheduling and other optimization problems.

Other scheduling problems. Hooker [2] has presented LBBD models for extensions of the problem studied here with a number of different optimization functions. For such problems, LBBD is able to produce feasible sub-optimal solutions without necessarily finding an optimal solution. Therefore, one of the main advantage of the MIP and CIP techniques compared to LBBD does not appear. It will be valuable to understand how adaptations of the MIP and CIP models

presented here perform on such problems. Another important class of scheduling problems has temporal constraints among jobs on different resources. Such constraints destroy the independent sub-problem structure that LBBD and, to a lesser extent, the other models exploit. However, exact techniques currently struggle on such problems including flexible job shop scheduling [20].

Scaling. As shown in Table 2, all models are unable to find optimal solutions as the number of jobs increases. With even more jobs, the only achievable performance measure will be the quality of feasible solutions that are found. We expect LBBD to perform poorly given that it cannot find sub-optimal solutions. However, as the problem size increases the time-indexed formulation on the MIP model will also fail due to model size. CIP[CP] and the pure CP model [1] would appear to be the only exact techniques likely to continue to deliver feasible solutions. Confirming this conjecture, as well as comparing the model performance to incomplete techniques (i.e., heuristics and metaheuristics) is therefore another area for future work.

7 Conclusions

The primary conclusions of these experiments with more sophisticated problem models are consistent with and reinforce those of Heinz & Beck [1]: CIP is a promising scheduling technology that is comparable to the state-of-the-art manual decomposition approach on resource allocation/scheduling problems and MIP approaches, though often discounted by constraint programming researchers, deserve consideration as a core technology for scheduling.

In arriving at these conclusions, we used two primary measures of model performance: the number of problem instances solved to proven optimality and the proven quality of solutions found, given that not all instances were solved to optimality. CIP comes second to LBBD by the former measure and to MIP by the latter. Depending on the importance placed on these measures any of the three algorithms could be declared the "winner". For practical purposes, we believe that the importance of proven solution quality should not be underestimated: in an industrial context it is typically better to consistently produce proven good solutions than to often find optimal solutions but sometimes fail to find any feasible solution at all.

References

1. Heinz, S., Beck, J.C.: Reconsidering mixed integer programming and MIP-based hybrids for scheduling. In: Beldiceanu, N., Jussien, N., Pinson, É. (eds.) CPAIOR 2012. LNCS, vol. 7298, pp. 211–227. Springer, Heidelberg (2012)
2. Hooker, J.N.: Integrated Methods for Optimization. Springer (2007)
3. Beck, J.C.: Checking-up on branch-and-check. In: Cohen, D. (ed.) CP 2010. LNCS, vol. 6308, pp. 84–98. Springer, Heidelberg (2010)
4. Hooker, J.N.: Testing heuristics: We have it all wrong. Journal of Heuristics 1, 33–42 (1995)

5. Hooker, J.N.: Planning and scheduling to minimize tardiness. In: van Beek, P. (ed.) CP 2005. LNCS, vol. 3709, pp. 314–327. Springer, Heidelberg (2005)
6. Hooker, J.N., Ottosson, G.: Logic-based Benders decomposition. Mathematical Programming 96, 33–60 (2003)
7. Benders, J.: Partitioning procedures for solving mixed-variables programming problems. Numerische Mathematik 4, 238–252 (1962)
8. Hooker, J.N.: Planning and scheduling by logic-based Benders decomposition. Operations Research 55, 588–602 (2007)
9. Yunes, T.H., Aron, I.D., Hooker, J.N.: An integrated solver for optimization problems. Operations Research 58(2), 342–356 (2010)
10. Achterberg, T.: Constraint Integer Programming. PhD thesis, Technische Universität Berlin (2007)
11. Achterberg, T.: SCIP: Solving Constraint Integer Programs. Mathematical Programming Computation 1(1), 1–41 (2009)
12. Heinz, S., Schulz, J., Beck, J.C.: Using dual presolving reductions to reformulate cumulative constraints. ZIB-Report 12-37, Zuse Institute Berlin (2012)
13. Baptiste, P., Pape, C.L., Nuijten, W.: Constraint-Based Scheduling. Kluwer Academic Publishers (2001)
14. Beck, J.C., Fox, M.S.: Constraint directed techniques for scheduling with alternative activities. Artificial Intelligence 121(1-2), 211–250 (2000)
15. Heinz, S., Schulz, J.: Explanations for the cumulative constraint: An experimental study. In: Pardalos, P.M., Rebennack, S. (eds.) SEA 2011. LNCS, vol. 6630, pp. 400–409. Springer, Heidelberg (2011)
16. Schutt, A., Feydy, T., Stuckey, P.J., Wallace, M.G.: Explaining the cumulative propagator. Constraints 16(3), 250–282 (2011)
17. Heinz, S., Beck, J.C.: Reconsidering mixed integer programming and MIP-based hybrids for scheduling. ZIB-Report 12-05, Zuse Institute Berlin (2012)
18. Wunderling, R.: Paralleler und objektorientierter Simplex-Algorithmus. PhD thesis, Technische Universität Berlin (1996)
19. Beldiceanu, N., Carlsson, M., Poder, E.: New filtering for the *cumulative* constraint in the context of non-overlapping rectangles. In: Trick, M.A. (ed.) CPAIOR 2008. LNCS, vol. 5015, pp. 21–35. Springer, Heidelberg (2008)
20. Fattahi, P., Saidi Mehrabad, M., Jolai, F.: Mathematical modeling and heuristic approaches to flexible job shop scheduling problems. Journal of Intelligent Manufacturing 18(3), 331–342 (2007)