Reconsidering Mixed Integer Programming and MIP-based Hybrids for Scheduling

Stefan Heinz[1,⋆] 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
jcb@mie.utoronto.ca

Abstract. Despite the success of constraint programming (CP) for scheduling, the much wider penetration of mixed integer programming (MIP) technology into business applications means that many practical scheduling problems are being addressed with MIP, at least as an initial approach. Furthermore, there has been impressive and well-documented improvements in the power of generic MIP solvers over the past decade. We empirically demonstrate that on an existing set of resource allocation and scheduling problems standard MIP and CP models are now competitive with the state-of-the-art manual decomposition approach. Motivated by this result, we formulate two tightly coupled hybrid models based on constraint integer programming (CIP) and demonstrate that these models, which embody advances in CP and MIP, are able to out-perform the CP, MIP, and decomposition models. We conclude that both MIP and CIP are technologies that should be considered along with CP for solving scheduling problems.

1 Introduction

While scheduling is often touted as a success story for constraint programming (CP) [1,2],[3] the wider success and exposure of mixed-integer programming (MIP) in many domains means that, for many practitioners, MIP is the default first approach for a new scheduling problem. In addition, driven to some extent by commercial pressures, there have been substantial improvements in MIP solvers over the past five to ten years [3] while the progress of commercial constraint programming solvers has not been as well documented. For scheduling researchers, these points suggest that solving scheduling problems using state-of-the-art MIP solvers should be considered.

In parallel, hybrid optimization methods that seek to combine the strengths of CP and MIP have been developed over the past 15 years [4]. Most notably, state-of-the-art methods for resource allocation and scheduling problems are based around logic-based Benders decomposition (LBBD) [5,6]. This loosely coupled hybrid approach decomposes the global problem into a master problem and a set of sub-problems, and then

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

[3] "Scheduling is a 'killer application' for constraint satisfaction" [2, p. 269].

employs an iterative problem solving cycle to converge to an optimal solution. One drawback of LBBD is that the decomposition is problem-specific and requires significant creative effort. In contrast, tightly coupled hybrids that seek to combine MIP and CP into a single solver and model [7,8] have not yet been widely applied to scheduling problems, though there have been some positive results [9,10].

In this paper, we focus on scheduling problems that combine resource allocation and scheduling. Given a set of jobs that each require the use of one of a set of alternative resources, a solution assigns each job to a resource and schedules the jobs such that the capacity of each resource is respected at all time points. Our investigations are presented in two steps reflecting our dual motivations. First, to investigate the advances in MIP and CP solving, we compare existing MIP, CP, and LBBD models. We show that while LBBD performance is consistent with earlier results, the CP and MIP models have substantially improved [6,11]. The improvements of MIP solvers lead to significantly better performance than both CP and LBBD. Second, based on such observations, we present two tightly coupled hybrids within the constraint integer programming (CIP) framework [7,12]. One model is motivated by adding linear relaxations to a CP model and while the other is based on adding global constraint propagation to a standard MIP model. Experiments show that both CIP models achieve performance better than the three previous models, both in terms of the number of problems solved and run time.

This paper does not introduce new modeling techniques or algorithms. For our comparison of standard MIP, CP, and LBBD models such novelty would defeat the purpose and the CIP models are based on known linear relaxations and inference techniques. The contributions of this paper lie in the demonstration (1) that, contrary to a common assumption in the CP scheduling community, MIP is a competitive technology for some scheduling problems and (2) that CIP is a promising hybrid framework for scheduling.

In the next section, we formally present the scheduling problems. Section 3 is our first inquiry: we define the CP, MIP, and LBBD models and present our experimental results. In Section 4, we formally present CIP while Section 5 defines two CIP models of our scheduling problems. Then in Section 6 we present and analyze our experiments comparing the CIP models to the existing models. In Section 7, we discuss perspectives and weaknesses of the work and, in the final section, conclude.

2 Problem Definition

We study two scheduling problems referred to as UNARY and MULTI [6,13]. Both 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 due date, $\mathcal{D}_j$, and execute for p_{jk} consecutive time units. Each job also has a resource assignment cost, c_{jk}, and a resource requirement, r_{jk}. Each resource, $k \in \mathcal{K}$, has a capacity, C_k, and the constraint that the resource capacity must not be exceeded at any time. In the UNARY problem, the capacities of the resources and the requirements of the jobs are one. For MULTI, capacities and requirements may be non-unary. A feasible solution is an assignment where each job is placed on exactly one resource and no resource is over capacity. The goal is to find an optimal solution, that is, a feasible solution which minimizes the total resource assignment cost.

$$
\begin{aligned}
\min \quad & \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}} c_{jk}\, x_{jk} \\
\text{s.t.} \quad & \sum_{k\in\mathcal{K}} x_{jk} = 1 && \forall j \in \mathcal{J} && (1)\\
& \texttt{optcumulative}(\boldsymbol{S}, \boldsymbol{x_{\cdot k}}, \boldsymbol{p_{\cdot k}}, \boldsymbol{r_{\cdot k}}, C_k) && \forall k \in \mathcal{K} && (2)\\
& 0 \le \mathcal{R}_j \le S_j \le \max_{k\in\mathcal{K}}\{(\mathcal{D}_j - p_{jk})\, x_{jk}\} && \forall j \in \mathcal{J} && (3)\\
& x_{jk} \in \{0,1\} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} \\
& S_j \in \mathbb{Z} && \forall j \in \mathcal{J}
\end{aligned}
$$

Model 1. Constraint programming model.

3 Reconsidering MIP

In this section, we present existing models using CP, MIP, and LBBD to solve the resource allocation/scheduling problems. We then present our results and a discussion. Unless otherwise indicated, the details of these models are due to Hooker [6].

Constraint Programming We use the standard CP model for our problem, defining two sets of decision variables: binary resource assignment variables, x_{jk}, which are assigned to 1 if and only if job j is assigned to resource k, and integer start time variables, S_j, which are assigned to the start-time of job j. Model 1 states the model.

The objective function minimizes the total resource allocation costs. Constraints (1) ensure that each job is assigned to exactly one resource. In Constraints (2), $\boldsymbol{S}$, $\boldsymbol{p_{\cdot k}}$, and $\boldsymbol{r_{\cdot k}}$ are vectors containing the start time variables, the processing times, and demands for each job if assigned to resource k. The global constraint `optcumulative` is the standard `cumulative` scheduling constraint [1] with the extension that the jobs are optionally executed on the resource and that this decision is governed by the $\boldsymbol{x_{\cdot k}}$ vector of decision variables. The `optcumulative` constraint enforces the resource capacity constraint over all time-points. Constraints (3) enforce the time-windows for each job.

We implement this model using IBM ILOG CP Optimizer. The assignment and start time variables are realized via optional and non-optional `IloIntervalVar` objects. For Constraints (1) we used the `IloAlternative` constraint linking the non-optional start time variables to the corresponding optional assignment variables. The `optcumulative` constraint is implemented by a cumulative constraint which contains the corresponding optional `IloIntervalVar`. For solving, we use the default search of IBM ILOG CP Optimizer which is tuned to find good feasible solutions.[4]

Mixed Integer Programming One of the standard MIP models for scheduling problems is the *time-indexed* formulation. The decision variable, y_{jkt}, is equal to 1 if and only if job j, starts at time t, on resource k. Sums over appropriate subsets of these variables form the resource capacity requirements. The model we use is defined in Model 2 where $T_{jkt} = \{t - p_{jk}, \ldots, t\}$.

[4] Philippe Laborie, personal communication, November 23, 2011.

$$
\begin{aligned}
\min \quad & \sum_{k \in \mathcal{K}} \sum_{j \in \mathcal{J}} \sum_{t=\mathcal{R}_j}^{\mathcal{D}_j - p_{jk}} c_{jk}\, y_{jkt} & & \\
\text{s.t.} \quad & \sum_{k \in \mathcal{K}} \sum_{t=\mathcal{R}_j}^{\mathcal{D}_j - p_{jk}} y_{kjt} = 1 & \forall j \in \mathcal{J} & \quad (4) \\
& \sum_{j \in \mathcal{J}} \sum_{t' \in T_{jkt}} r_{jk}\, y_{jkt'} \leq C_k & \forall k \in \mathcal{K},\ \forall t & \quad (5) \\
& y_{jkt} \in \{0, 1\} & \forall j \in \mathcal{J},\ \forall k \in \mathcal{K},\ \forall t &
\end{aligned}
$$

Model 2. Mixed integer programming model with $T_{jkt} = \{t - p_{jk}, \ldots, t\}$.

As in the CP model, the objective function minimizes the weighted resource assignment cost. Constraints (4) ensure that each job starts exactly once on one resource while Constraints (5) enforce the resource capacities on each resource at each time-point.

To solve this model, we rely on the default branch-and-bound search in the IBM ILOG CPLEX solver, a state-of-the-art commercial MIP solver.

Logic-based Benders Decomposition Logic-based Benders decomposition (LBBD) is a manual decomposition technique that generalizes classical Benders decomposition [5]. A problem is modeled as a master problem (MP) and a set of sub-problems (SPs) where the MP is a relaxation of the global problem designed such that a solution generates one or more SPs. Each SP is an inference dual problem that derives the tightest bound on the MP cost function that can be inferred from the current MP solution.

Solving a problem by LBBD is done by iteratively solving the MP and then solving each SP. If the MP solution satisfies all the bounds generated by the SPs, the MP solution is globally optimal, as it is a relaxation of the global problem. If not, a *Benders cut* is added to the MP by the violated SPs and the MP is re-solved. For models where the SPs are feasibility problems, it is sufficient to solve the SPs to feasibility or generate a cut that removes the current MP solution.

As in the CP model, the LBBD model defines two sets of decision variables: binary resource assignment variables, x_{jk}, and integer start time variables, S_j. The former variables are in the master problem while the latter are in sub-problems.

Formally, the LBBD master and sub-problem models are defined in Model 3. The objective function and first constraints are as in the CP model. Constraints (6) are a linear relaxations of each resource capacity constraint. They state that the area of the rectangle with height C_k and width from the smallest release date to the largest deadline must be greater than the sum of the areas of the jobs assigned to the resource. Constraints (7) are the Benders cuts. Let H indicate the index of the current iteration and $\mathcal{J}_{hk}$ denote the set of jobs that resulted in an infeasible sub-problem for resource k in iteration $h < H$. The Benders cut, then, simply states that the set of jobs assigned to resource k in iteration h should not be reassigned to the same resource.

Because the MP assigns each job to a resource and there are no inter-job constraints, the SPs are independent, single-machine scheduling problems where it is necessary to

$$
\begin{aligned}
\text{(MP)} \quad & \min \quad \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}} c_{jk}\, x_{jk} && && \\
& \text{s.t.} \quad \sum_{k\in\mathcal{K}} x_{jk} = 1 && \forall j \in \mathcal{J} && \\
& \qquad \sum_{j\in\mathcal{J}} p_{jk} r_{jk}\, x_{jk} \le \hat{C}_k && \forall k \in \mathcal{K} && (6)\\
& \qquad \sum_{j\in\mathcal{J}_{hk}} (1 - x_{jk}) \ge 1 && \forall k \in \mathcal{K},\ \forall h \in \{1,\dots,H-1\} && (7)\\
& \qquad x_{kj} \in \{0,1\} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} && \\
\text{(SP)} \quad & \qquad \texttt{cumulative}(\boldsymbol{S}, \boldsymbol{p_{\cdot k}}, \boldsymbol{r_{\cdot k}}, C_k) && && \\
& \qquad \mathcal{R}_j \le S_j \le \mathcal{D}_j - p_{jk} && \forall j \in \mathcal{J}_k && \\
& \qquad S_j \in \mathbb{Z} && \forall j \in \mathcal{J}_k &&
\end{aligned}
$$

Model 3. Logic-based Benders decomposition: master problem (MP) on top and sub-problem (SP) for resource k below. $\hat{C}_k = C_k \cdot (\max_{j\in\mathcal{J}}\{\mathcal{D}_j\} - \min_{j\in\mathcal{J}}\{\mathcal{R}_j\})$

assign each job a start time such that its time window and the capacity of the resource are respected. The SP for resource k can be formulated as a constraint program as in Model 3, where $\mathcal{J}_k$ denotes the set of jobs assigned to resource k. The components of SP model are analogous to the parts of the CP model with the exception that the resource assignment decisions are made before the SP models are created.

The MP and SPs are modeled and solved using SCIP [12]. We use the standard bounds propagation [1] of the `cumulative` constraint.

3.1 Experimental Results

Set up We use the following solvers: IBM ILOG CP Optimizer 2.3 for the CP model, IBM ILOG CPLEX 12.2.0.2 running with one thread for the MIP model, and SCIP version 2.0.1.3 integrated with SoPlex version 1.5.0.3 as the underlying linear programming solver [14] for LBBD.

We use the scheduling instances introduced by [6]. 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 two resources the number of maximum number of jobs is 38. For each problem size, we have five instances. 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 [6] for further details w.r.t. generation of instances and the appendix of [15] for further problem instance characteristics.

All computations reported were obtained on Intel Xeon E5420 2.50 GHz computers (in 64 bit mode) with 6 MB cache, running Linux, and 6 GB of main memory. We enforced a time limit of 7200 seconds.

Results For each test set and model, Table 1 displays the number of instances for which a feasible solution was found, for which the optimal solution was found (but

Table 1. Results for each test set (UNARY and MULTI) and each model stating 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 display the shifted geometric mean for the total running time and time until the best solution was found. The time to best solution is only an upper bound in case of IBM ILOG CPLEX since the output log does not display this time point explicitly.

	UNARY					MULTI				
	CP	MIP	LBBD	CIP[CP]	CIP[MIP]	CP	MIP	LBBD	CIP[CP]	CIP[MIP]
feasible	**195**	**195**	175	**195**	**195**	**195**	**195**	119	125	**195**
optimal found	187	**195**	175	194	**195**	119	**148**	119	124	142
optimal proved	19	191	175	194	**195**	5	109	119	123	**133**
best known found	187	**195**	175	194	**195**	130	**155**	119	124	146
total time	3793	12	28	**10**	19	6082	442	228	**212**	395
time to best	**7**	**7**	28	9	17	**64**	209	228	200	217

not necessarily proved), for which the optimal solution was found and proved, and for which the best known solution was found. Optimal solutions are known for all 195 instances of the UNARY set. For the test set MULTI 181 optimal solutions are known. We present the *shifted geometric mean*[5] of the total solve time per instance and time per instance to find the best solution found by the model. The shifted geometric mean reduces the influence of outliers, both very hard and very easy instances. See [7] for a detailed discussion of aggregate measures. For each category we used a bold font to indicate the model(s) which performs best on a given criterion. We postpone the discussion of the final two columns/models for each problem set to Section 6.

These results indicate the MIP and CP models out-perform LBBD on all measures except the number of optimal found and proved where LBBD is superior to CP on both problem sets and superior to MIP on the MULTI set. The CP and MIP models are able to find feasible solutions for all instances while LBBD suffers from the fact that its first globally feasible master solution is by definition optimal and, thus, there are no intermediate feasible solutions available. The total run-times substantially favor MIP on the UNARY set and LBBD on the MULTI set while the time to best solution found favors CP, though tied with MIP on the UNARY problems.

The results indicate that MIP model performs best as it finds feasible solutions for all problems, the most best known solutions, proves optimality for the greatest number of instances overall, and delivers competitive run-times.

To complement this overview, Tables 2 and 3 present detailed results for the CP, MIP, and LBBD models on the UNARY test set and MULTI test set, respectively. The 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. We again use the shifted geometric mean with shift $s = 10$ for time and $s = 100$ for nodes. 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.

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

Table 2. Results for the UNARY test set. Each resource job combination consists of 5 instances for a total of 195. The running times are rounded up and given in seconds.

		CP				MIP				LBBD				CIP[CP]				CIP[MIP]			
$\|\mathcal{K}\|$	$\|\mathcal{J}\|$	opt	feas	nodes	time	opt	feas	nodes	time	opt	feas	nodes	time	opt	feas	nodes	time	opt	feas	nodes	time
2	10	5	5	1160	**0**	5	5	1	**0**	5	5	62	1	5	5	26	**0**	5	5	1	1
	12	2	5	2 035 k	511	5	5	8	1	5	5	116	1	5	5	59	**0**	5	5	2	1
	14	0	5	134 052 k	–	5	5	77	**1**	5	5	567	2	5	5	131	**1**	5	5	4	2
	16	0	5	134 655 k	–	5	5	49	**1**	5	5	81	**1**	5	5	140	**1**	5	5	3	2
	18	2	5	1 065 k	510	5	5	130	2	5	5	76	**1**	5	5	217	**1**	5	5	49	10
	20	0	5	141 258 k	–	5	5	669	11	5	5	441	3	5	5	270	**1**	5	5	23	10
	22	0	5	131 240 k	–	5	5	118	2	5	5	196	2	5	5	118	**1**	5	5	24	12
	24	1	5	8 424 k	1924	5	5	149	3	5	5	23	16	5	5	163	**1**	5	5	77	24
	26	0	5	116 549 k	–	5	5	1390	16	4	4	301	34	5	5	440	**1**	5	5	115	31
	28	0	5	125 223 k	–	4	5	2057	44	5	5	511	29	5	5	347	**1**	5	5	337	54
	30	0	5	131 057 k	–	4	5	12 k	160	4	4	1837	75	5	5	2140	**9**	5	5	261	68
	32	0	5	128 084 k	–	5	5	257	6	5	5	288	3	5	5	707	**1**	5	5	117	56
	34	0	5	126 592 k	–	5	5	677	18	4	4	275	44	5	5	898	**2**	5	5	190	60
	36	1	5	22 855 k	1975	5	5	346	8	1	1	657	2012	5	5	1015	**2**	5	5	199	100
	38	1	5	7 898 k	1924	5	5	502	16	3	3	984	425	5	5	1077	**1**	5	5	135	88
3	10	3	5	249 k	130	5	5	1	**0**	5	5	357	1	5	5	74	1	5	5	1	1
	12	0	5	127 293 k	–	5	5	3	**0**	5	5	191	1	5	5	120	1	5	5	1	1
	14	0	5	122 754 k	–	5	5	20	**1**	5	5	2760	5	5	5	315	**1**	5	5	4	3
	16	0	5	117 197 k	–	5	5	109	**1**	5	5	224	**1**	5	5	323	**1**	5	5	6	4
	18	0	5	127 851 k	–	5	5	112	**1**	5	5	445	**1**	5	5	633	2	5	5	6	5
	20	0	5	128 864 k	–	5	5	374	**2**	5	5	1899	9	5	5	957	3	5	5	51	14
	22	0	5	126 140 k	–	5	5	258	**2**	5	5	1107	13	5	5	1218	4	5	5	19	13
	24	0	5	141 427 k	–	5	5	587	7	5	5	1746	**6**	5	5	1642	7	5	5	24	13
	26	1	5	13 667 k	1927	5	5	1081	**13**	5	5	18 k	58	5	5	5648	23	5	5	46	19
	28	1	5	17 842 k	1932	5	5	491	14	5	5	3722	**12**	5	5	4592	19	5	5	221	69
	30	0	5	140 336 k	–	4	5	26 k	175	3	3	12 k	134	5	5	19 k	104	5	5	100	**52**
	32	0	5	130 588 k	–	5	5	4520	56	4	4	6229	96	5	5	11 k	**52**	5	5	492	83
4	10	2	5	1 806 k	511	5	5	1	**0**	5	5	263	1	5	5	30	**0**	5	5	1	1
	12	0	5	114 608 k	–	5	5	1	**0**	5	5	590	2	5	5	80	1	5	5	1	1
	14	0	5	110 186 k	–	5	5	3	**1**	5	5	2391	6	5	5	212	**1**	5	5	1	**1**
	16	0	5	123 967 k	–	5	5	37	**1**	5	5	23 k	42	5	5	769	3	5	5	4	4
	18	0	5	120 008 k	–	5	5	7	**1**	4	4	9858	62	5	5	905	3	5	5	2	3
	20	0	5	118 755 k	–	5	5	334	**2**	5	5	20 k	24	5	5	2526	9	5	5	11	12
	22	0	5	127 409 k	–	5	5	1665	**9**	3	3	223 k	246	5	5	8913	45	5	5	114	28
	24	0	5	121 900 k	–	5	5	679	**5**	4	4	44 k	71	5	5	7356	39	5	5	58	24
	26	0	5	129 501 k	–	5	5	4514	**35**	4	4	152 k	257	5	5	38 k	180	5	5	83	44
	28	0	5	125 818 k	–	5	5	15 k	144	4	4	243 k	376	5	5	34 k	176	5	5	272	**90**
	30	0	5	126 857 k	–	4	5	74 k	508	4	4	130 k	130	4	5	64 k	379	5	5	256	**101**
	32	0	5	121 034 k	–	5	5	13 k	211	4	4	527 k	488	5	5	74 k	492	5	5	259	**176**
		19	195	42 746 k	3793	191	195	501	12	175	175	2178	28	194	195	1112	**10**	195	195	66	19

The CP model only solved 19 and 5 instances, respectively, to optimality. Hence, the "nodes" and "time" columns are meaningless since they do not reflect the strength of finding good feasible solutions quickly. We include them for completeness.

For the UNARY problems, the MIP model preforms consistently better than LBBD independently of the problem size. That changes for the MULTI test set where LBBD solved more large problems but eventually also fails to find optimal solutions.

Since the MIP method provides a lower bound, we have a quality measure for the solutions which are not solved to optimality. The mean percentage gap between its best feasible solution and lower bound is 0.94%, demonstrating that MIP is able to find proven good feasible solutions. In contrast, the other two models cannot provide any quality information by themselves since LBBD cannot find any intermediate feasible solutions for these problems and CP does not provide a lower bound.

Table 3. Results for the MULTI test set. Each resource job combination consists of 5 instances. This adds up to a total of 195. The running times are rounded up and given in seconds.

		CP				MIP				LBBD				CIP[CP]				CIP[MIP]			
$\|\mathcal{K}\|$	$\|\mathcal{J}\|$	opt	feas	nodes	time	opt	feas	nodes	time	opt	feas	nodes	time	opt	feas	nodes	time	opt	feas	nodes	time
2	10	2	5	1 555 k	509	5	5	38	1	5	5	52	1	5	5	153	**0**	5	5	6	2
	12	0	5	149 191 k	–	5	5	147	1	5	5	20	1	5	5	156	**0**	5	5	58	4
	14	0	5	160 424 k	–	5	5	202	**1**	5	5	7	3	5	5	343	**1**	5	5	130	5
	16	0	5	157 963 k	–	5	5	2339	**11**	5	5	4	17	5	5	3111	19	5	5	3393	30
	18	0	5	167 188 k	–	4	5	25 k	162	5	5	8	89	5	5	9952	**18**	5	5	11 k	77
	20	0	5	168 579 k	–	3	5	71 k	401	3	3	21	158	5	5	4107	**3**	5	5	11 k	139
	22	0	5	171 979 k	–	2	5	151 k	1442	2	2	10	**703**	2	2	339 k	1325	4	5	417 k	2550
	24	0	5	174 557 k	–	2	5	305 k	2197	0	0	1	–	3	3	354 k	**707**	3	5	66 k	1180
	26	0	5	175 929 k	–	3	5	578 k	**2977**	1	1	1	5193	1	1	1 715 k	5440	2	5	265 k	3261
	28	0	5	173 741 k	–	2	5	333 k	2503	3	3	11	441	3	3	91 k	**160**	2	5	198 k	2598
	30	0	5	180 622 k	–	1	5	669 k	5429	1	1	1	**2972**	0	0	2 390 k	–	1	5	182 k	4180
	32	0	5	177 335 k	–	0	5	816 k	–	1	1	1	5680	3	3	495 k	**282**	1	5	319 k	6123
	34	0	5	182 303 k	–	1	5	322 k	3448	1	1	1	3015	1	1	2 730 k	**1397**	2	5	90 k	4265
	36	0	5	174 330 k	–	1	5	446 k	6052	1	1	1	2044	2	2	1 314 k	**700**	1	5	115 k	4678
	38	0	5	181 485 k	–	0	5	460 k	–	1	1	1	3369	3	3	6 442 k	**1676**	2	5	73 k	5095
3	10	2	5	1 998 k	510	5	5	7	**0**	5	5	50	1	5	5	85	**0**	5	5	4	1
	12	0	5	139 631 k	–	5	5	100	**1**	5	5	268	**1**	5	5	481	**1**	5	5	59	5
	14	0	5	140 052 k	–	5	5	220	**1**	5	5	95	**1**	5	5	1153	2	5	5	234	11
	16	0	5	156 864 k	–	5	5	3622	14	5	5	838	**10**	5	5	15 k	22	5	5	3196	60
	18	0	5	151 398 k	–	5	5	164 k	429	5	5	3197	**21**	4	4	202 k	139	4	5	18 k	296
	20	0	5	165 255 k	–	4	5	409 k	1124	5	5	1614	**6**	5	5	38 k	35	5	5	8427	253
	22	0	5	164 038 k	–	2	5	818 k	6014	5	5	2254	**149**	2	2	633 k	1352	5	5	38 k	505
	24	0	5	163 222 k	–	2	5	439 k	3253	1	1	813	2324	1	1	1 661 k	3165	4	5	41 k	**1001**
	26	0	5	172 448 k	–	0	5	452 k	–	4	4	1341	1351	3	3	651 k	**727**	1	5	269 k	4467
	28	0	5	174 771 k	–	2	5	200 k	1829	0	0	9	–	2	3	726 k	**1261**	2	5	60 k	3057
	30	0	5	179 915 k	–	0	5	376 k	–	0	0	50	–	0	0	1 383 k	–	2	5	75 k	**5435**
	32	0	5	177 797 k	–	0	5	471 k	–	0	0	4	–	1	1	1 748 k	6918	1	5	76 k	**6639**
4	10	1	5	16 295 k	1926	5	5	13	**0**	5	5	14	1	5	5	106	1	5	5	1	1
	12	0	5	146 205 k	–	5	5	18	**1**	5	5	31	**1**	5	5	243	**1**	5	5	61	4
	14	0	5	149 547 k	–	5	5	210	**1**	5	5	389	2	5	5	1119	2	5	5	136	12
	16	0	5	145 424 k	–	5	5	363	2	5	5	252	**1**	5	5	1095	2	5	5	79	11
	18	0	5	158 035 k	–	5	5	18 k	38	5	5	3297	**4**	5	5	29 k	21	5	5	2395	67
	20	0	5	150 735 k	–	5	5	108 k	309	5	5	1298	**27**	5	5	35 k	29	5	5	3650	106
	22	0	5	147 950 k	–	4	5	64 k	324	5	5	3364	**46**	4	4	78 k	167	5	5	16 k	544
	24	0	5	159 240 k	–	0	5	535 k	–	2	2	1980	**1446**	2	2	1 797 k	3021	2	5	253 k	4184
	26	0	5	172 713 k	–	0	5	485 k	–	1	1	16 k	**4070**	0	1	2 437 k	–	2	5	170 k	5530
	28	0	5	174 855 k	–	1	5	370 k	5034	1	1	680	**2804**	1	1	1 310 k	6575	2	5	101 k	3885
	30	0	5	169 821 k	–	0	5	364 k	–	1	1	187	**2105**	0	0	1 782 k	–	0	5	118 k	–
	32	0	5	178 032 k	–	0	5	323 k	–	0	0	136	–	0	0	1 973 k	–	0	5	81 k	–
		5	195	122 736 k	6082	109	195	34 k	442	119	119	223	228	123	125	62 k	**212**	133	195	12 k	395

Overall, all three approach fail to find optimal solutions when the problem size increases. It is notable, that CP and MIP consistently provide high quality solutions independently of the problem size.

3.2 Discussion

The results of the CP model are different form those of Hooker [6] and those recently reproduced in [11]. It was shown that instances with 18 jobs or more could not be solved to optimality and finding even feasible solution was an issue. Using IBM ILOG CP Optimizer instead of IBM ILOG Solver and IBM ILOG Scheduler leads to a significant increase in the number of instances for which a high quality solution was found. However, it also leads to a substantial decrease in the number of instances solved to proven optimality. From our perspective these results are an improvement over those of Hooker as high quality solutions are found for all instances even though no quality

gap is provided. We believe we are using substantially the same model as Hooker and so attribute the difference in performance to the different underlying CP solvers.

Results on the LBBD for the UNARY instances were not presented by Hooker [6] but they are consistent with previously published results using a separate implementation (using IBM ILOG CPLEX and IBM ILOG Scheduler) by Beck [13]. In contrast, the LBBD results for the MULTI test set, are not consistent with the previous implementation of Beck [13]. He solved 175 instances, 56 more than our LBBD model. We suspect that using SCIP for solving the sub-problems instead of IBM ILOG Solver and IBM ILOG Scheduler leads to these differences. We plan to further investigate this issue.

The MIP results are substantially better than those reported by Hooker. This model significantly out-performs the CP and LBBD model for the UNARY test set. For the MULTI instances, the MIP method is competitive to LBBD w.r.t. proven optimality (taking into account the results of [13]). Overall, however, the MIP approach dominates this test set as well since it finds high quality solutions for those instances which are not solved to proven optimality. As this was not the case on the MULTI problems in Hooker's 2005 paper [6] and we use the same models, the difference appears to be due to the changes in the underlying MIP solver in the past six years.

Given these results, the question arises of whether we can combine CP and MIP techniques to achieve even better performance. As noted above, this question is not new as attested by a number of publications over the past decade, notably [16,4], as well as by the existence of the CPAIOR conference series. Indeed, the LBBD framework itself is one positive answer to this question. However, the decomposition model suffers at least two weaknesses. First, a workable decomposition is difficult to develop and then limited in its applicability in the face of simple side constraints (e.g., the addition of precedence constraints between jobs on different resources). Second, for some models such as the ones studied here, LBBD cannot find good feasible solutions before finding an optimal one. For larger problems, therefore, LBBD is likely not to return a usable result at all, a significant weakness from a practical point of view.

In seeking to preserve the advantages of the MIP model, in the balance of this paper, we focus on an alternative to decomposition-based hybridization in the form of constraint integer programming (CIP). Our goals are:

- to increase problem solving performance through the combination of CP-style inference and MIP-style relaxation (cf. [16])
- to maintain the modeling flexibility of CP and MIP
- to maintain the higher level structure and modeling flexibility of global constraints

4 Constraint Integer Programming

The power of CP arises from the possibility to directly model a given problem with a variety of expressive constraints and to use constraint-specific inference algorithms to reduce search. In contrast, MIP only admits very specific constraint forms (i.e., linear and integrality constraints) but uses sophisticated techniques to exploit the structure provided by this limited constraint language.

Constraint Integer Programming (CIP) [7,12] seeks to combine the advantages and compensate for the weaknesses of CP and MIP. Intuitively, a constraint integer programming is a constraint program over integer and continuous variables with the restriction that, once the integer variables are assigned, the remaining problem (if any) is a linear program. Formally a constraint integer program can be defined as follows.

Definition 1 ([7]). *A* constraint integer program (CIP) $(\mathfrak{C}, I, c)$ *consists of solving*

$$c^\star = \min\{c^T x \mid \mathfrak{C}(x),\ x \in \mathbb{R}^n,\ x_j \in \mathbb{Z}, \forall j \in I\}$$

with a finite set $\mathfrak{C} = \{\mathcal{C}_1, \ldots, \mathcal{C}_m\}$ *of constraints* $\mathcal{C}_i : \mathbb{R}^n \to \{0,1\}$, $i = \{1, \ldots, m\}$, *a subset* $I \subseteq N = \{1, \ldots, n\}$ *of the variable index set, and an objective function vector* $c \in \mathbb{R}^n$. *A CIP must fulfill the following additional condition:*

$$\forall \hat{x}_I \in \mathbb{Z}^I\ \exists (A', b') : \quad \{x_C \in \mathbb{R}^C \mid \mathfrak{C}(\hat{x}_I, x_C)\} = \{x_C \in \mathbb{R}^C \mid A' x_C \leq b'\} \tag{8}$$

with $C := N \setminus I$, $A' \in \mathbb{R}^{k \times C}$, *and* $b' \in \mathbb{R}^k$ *for some* $k \in \mathbb{Z}_{\geq 0}$.

Restriction (8) ensures that the sub-problem remaining after fixing all integer variables is a linear program. Note that the restriction does not forbid nonlinear or arbitrary global constraints – as long as the non-linearity only refers to the integer variables.

The central solving approach for CIP as implemented in the SCIP framework [12] is branch-and-cut-and-propagate: as in CP and MIP solvers, SCIP performs a branch-and-bound search. Also as in MIP, a linear relaxation, strengthened by additional cutting planes if possible, is solved at each search node and used to guide and bound search. Similar to CP solvers, inference in the form of constraint propagation is used at each node to further restrict search and detect dead-ends. Moreover, as in SAT solving, SCIP uses conflict analysis and restarts.

CIP has been applied to MIP [12], mixed-integer nonlinear programming [17], nonlinear pseudo-Boolean programming [18], chip verification [19], and scheduling [10].

5 Two CIP Models

We define two CIP models in this section: CIP[CP] is motivated by the CP model and adds a linear relaxation and the solving techniques of modern MIP solvers to the CP model defined above; the CIP[MIP] model is inspired by the standard MIP model and can be seen as adding the `cumulative` constraint propagation plus (linear) channeling constraints to the MIP model.

The CIP[CP] Model The CIP[CP] model is identical to Model (1) with the addition of a linear relaxation of the `optcumulative` constraint. As noted below, a key part of solving MIPs and CIPs is exploiting the linear relaxation of the problem. Therefore, in addition to the constraints in the CP model, all of which are linear or integrality constraints except `cumulative`, we add the `optcumulative` linear relaxation represented by Constraint (6) of the LBBD model. Model 4 displays this model.

The default parameters of SCIP are used to solve the CIP[CP] model with the addition of a variable prioritization rule. The x_{jk} are given higher branching priority than the S_{jk} variables. This rule means that the start time variables will not be branched on until all resource assignment variables are fixed.

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

Model 4. CIP[CP]: A CIP model based on the CP model.

The CIP[MIP] Model The CIP[MIP] model adds the `optcumulative` constraint and channeling constraints to Model (2). For completeness, the CIP[MIP] model is formally defined in Model 5. Note that the `optcumulative` constraint is logically redundant as the MIP model is a complete model of the problem.

5.1 Solving CIP Models

To solve the CIP models, we use the hybrid problem solving techniques implemented in SCIP. These techniques include the following.

Presolving. The purpose of presolving, which takes place before the tree search, is to (1) reduce the size of the model by removing irrelevant information such as fixed variables; (2) strengthen the linear relaxation by exploiting integrality information; (3) extract structural information from the model which can be used for branching heuristics and cutting plane generation. The `optcumulative` constraint can contribute to a number of reformulations in presolving, including normalization of the demands and the resource capacity and detection of irrelevant jobs that do not influence the feasibility or optimality of the remaining jobs on that resource. For example, if a job has a latest completion time which is smaller than the earliest start time of all remaining jobs then this job is irrelevant and can be ignored.

Propagation. Following [20], we adapt the standard bounds-based cumulative propagation: we propagate all jobs that are known to execute on the resource with standard `cumulative` propagation [1]. Then, for each job j that is still optional, we perform singleton arc-consistency (SAC) [21]: we assume that the job will execute on the resource and trigger propagation.[6] If the propagation derives a dead-end, we can soundly conclude that the job cannot execute on the resource and appropriately set the x_{jk} variable. Otherwise, we retain the pruned domains for the implicit S_{jk} variable. In either case, the domains of all other variables are restored to their states before SAC.

[6] SAC is similar but more general than the shaving technique in the scheduling literature [22].

$$
\begin{aligned}
\min \quad & \sum_{k\in\mathcal{K}}\sum_{j\in\mathcal{J}}\sum_{t=\mathcal{R}_j}^{\mathcal{D}_j-p_{jk}} c_{jk}\, y_{jkt} \\
\text{s.t.} \quad & \sum_{k\in\mathcal{K}}\sum_{t=\mathcal{R}_j}^{\mathcal{D}_j-p_{jk}} y_{kjt} = 1 && \forall j \in \mathcal{J} \\
& \sum_{j\in\mathcal{J}}\sum_{t'\in T_{jkt}} r_{jk}\, y_{jkt'} \leq C_k && \forall k \in \mathcal{K},\ \forall t \\
& \sum_{t=\mathcal{R}_j}^{\mathcal{D}_j-p_j} y_{jkt} = x_{jk} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} \qquad (9) \\
& \sum_{t=\mathcal{R}_j}^{\mathcal{D}_j-p_j} t \cdot y_{jkt} = S_{jk} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} \qquad (10) \\
& \texttt{cumulative}(\boldsymbol{S_{\cdot k}}, \boldsymbol{x_{\cdot k}}, \boldsymbol{p_{\cdot k}}, \boldsymbol{r_{\cdot k}}, C_k) && \forall k \in \mathcal{K} \\
& y_{jkt} \in \{0,1\} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K},\ \forall t \\
& x_{jk} \in \{0,1\} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K} \\
& S_{jk} \in \mathbb{Z} && \forall j \in \mathcal{J},\ \forall k \in \mathcal{K}
\end{aligned}
$$

Model 5. CIP[MIP]: A CIP model based on the MIP model with channeling Constraints (9), (10).

Linear Relaxation. The linear relaxation can be solved efficiently to optimality and used in two primary ways: (1) to provide a guiding information for the search and (2) as the source of a valid lower bound on the objective function.

Branching Heuristics. As in CP and MIP, the branching decisions are crucial in CIP. SCIP uses *hybrid branching*, a heuristic which combines several metrics including cost, propagation, and constraint activity to decide on a branching variable [23].

Conflict Analysis. The idea of conflict analysis is to reason about infeasible sub-problems which arise during the search in order to generate *conflict clauses* [24,25]. These conflict clauses are used to detect similar infeasible sub-problems later in the search. In conflict analysis, a bound change made during the search needs to be explained by a set of bounds which imply the bound change. The explanations are used to build up a conflict graph which is used to derive valid conflict clauses. Each time the `optcumulative` has to explain a bound change it first uses the standard `cumulative` explanation algorithm [26,27] to derive an initial explanation. The explanation is extended with the bounds of all resource assignment variables which are (locally) fixed to one. In case of the SAC propagation, a valid explanation is the bounds of all resource assignment variables which are fixed to one at the moment of the propagation.

6 Experiments with CIP

In this section we compare the two CIP models with the CP, MIP, and LBBD models above. The experimental set-up and hardware is as defined in Section 3.1. The CIP

models are implemented with SCIP version 2.1.0.3 integrated with SoPlex version 1.5.0.3 as the underlying linear programming solver.

Table 1 shows that the CIP models are very strong performers with CIP[MIP] dominating all other models on the UNARY instances and being competitive with MIP, the best previous model, on the MULTI instances.

UNARY. On the UNARY problems (Table 2), the CIP[MIP] model finds and proves optimality for all 195 problem instances while CIP[CP] times-out on only one instance (with 30 jobs and 4 resources). This performance is better than the other models. Like the CP and MIP models, the CIP models find feasible solutions for all UNARY instances.

The CIP[CP] model is slightly faster than the MIP model, about three times faster than LBBD, and twice as fast as CIP[MIP]. However, note that number of nodes used by CIP[MIP] is 20 times smaller than for CIP[CP] which has the second lowest shifted geometric mean number of nodes. We return to this observation in Section 7. The node count of LBBD includes only the nodes in the master problem search not the sub-problems. The time, however, includes both master and sub-problem solving.

MULTI. The CIP models perform best in terms of optimality on the MULTI problem instances (Table 3). CIP[MIP] finds and proves optimality for 133 of 195 while CIP[CP] achieves the same on 123 instances. Recall that LBBD and MIP perform reasonably with 119 and 109 instances solved to optimality respectively while CP finds and proves only 5 optimal solutions. CIP[MIP], MIP, and CP find feasible solutions for all MULTI instances while CIP[CP] only finds feasible solutions for 125 instances with 119 for LBBD. Comparing CIP[MIP] and MIP on solution quality shows that CIP[MIP] has a mean percentage-gap of 0.68%, better than MIP at 0.94%. Furthermore, CIP[MIP] achieves an equal or better solution than MIP on 164 of 195 instances.

CIP[CP] and LBBD achieve similar run-times, about twice as fast as MIP. CIP[MIP] is 1.8 times slower than CIP[CP] albeit while solving more problems.

7 Discussion

Existing Models The results of our first experiment indicate that both MIP and CP technology have progressed to the point where LBBD is no longer the clearly dominant choice for solving resource allocation and scheduling problems. Indeed, our results indicate that a monolithic MIP model can perform much better across all criteria while a monolithic CP model is a stronger at finding high quality solutions quickly.

We do not want to make broad generalizations from these results. In particular, we have studied only two (closely related) types of resource allocation and scheduling problems. Furthermore, the size of the time-indexed MIP model scales with the time granularity and so there will clearly be a point where both CP and LBBD out-perform it. Given the inability of the LBBD model to return intermediate solutions for these problems, we can further predict that CP will eventually be the only usable model (of these three) for finding feasible solutions as problem size scales.

However, we believe that our results support our claim that MIP-based models for scheduling problems should be reconsidered in light of modern MIP solver performance. At the least, we have shown MIP models to be competitive on a set of benchmarks in the literature. As a point of even stronger support, commercial MIP solvers

Table 4. Percentage of run-time over all instances spend for the linear relaxation and `optcumulative` propagation.

	CIP[MIP]		CIP[CP]	
Test set	linear relaxation	propagation	linear relaxation	propagation
UNARY	69.6 %	8.4 %	1.5 %	96.2 %
MULTI	58.3 %	22.6 %	1.7 %	12.3 %

now routinely make use of multi-core machines. Within the same computational environment as above but using eight threads, the MIP model solves 192 and 136 instances solved to optimality for UNARY and MULTI with shifted geometric run-times of 7.6 and 227.7 seconds, respectively.

The CIP Models Based only on the results presented above, we would be justified in claiming that CIP is the best performing approach to the resource allocation/scheduling problems investigated. Both CIP models find more optimal solutions and better feasible solutions than the other techniques. The LBBD results, however, presented in [13] on the same problems sets, albeit using a different implementation, underlying solvers, and hardware, are superior to the CIP results here. Furthermore, Hooker [28] presents an alternative LBBD formulation for these problems with a tighter relaxation and Benders cuts. His empirical results, again using a different implementation and environment, are better than the LBBD results above but appear to be worse than our CIP results.

Therefore, we choose to be cautious in our claims: our empirical results demonstrate that a CIP approach to these scheduling problems is competitive with the LBBD approach while being considerably better than the MIP and CP models: CIP models currently represent the best non-decomposition-based approach to the problems studied. Together with the paper of Berthold et al. [10], these results provide strong evidence of the promise of CIP for scheduling.

Comparing the CIP Models. It may be useful to view the CIP models as identical except for their linear relaxations. In the CIP[MIP] model, the channeling constraints ensure that time-index variables and the start time variables are coherent and equivalent. Both models therefore have resource assignment variables and start time variables, bounds constraints, integrality constraints, and `optcumulative` constraints. However, CIP[MIP] has a substantially stronger and larger linear relaxation via the knapsack constraints (Constraints (5) and relaxed time-index variables) for each time point.

This perspective explains the relative performance of the two models. The LP relaxation for CIP[MIP] is harder to solve, due to its size, but provides better bounding and heuristic guidance. As a consequence, we see between 5 and 20 times fewer nodes in the CIP[MIP] runs than in the CIP[CP] runs (in shifted geometric mean on the UNARY and MULTI instances, respectively). Furthermore, while CIP[MIP] solves more problem instances, it tends to be much slower than CIP[CP] especially on instances with fewer than about 22 jobs. Table 4 supports this analysis by showing that CIP[MIP] spends a considerably larger percentage of its run-time solving the linear program than in propagating the `optcumulative` constraint. For CIP[CP], the reverse is true.

While the tighter but larger LP allowed the CIP[MIP] model to solve more instances than CIP[CP] here, it also represents an inherent weakness of the model. The CIP[MIP]

model, like the time-indexed MIP formulation, scales with the time granularity. For problem instances with longer horizons, therefore, we would expect the CIP[CP] model to out-perform CIP[MIP].

Comparing CIP and LBBD. There is also, of course, a relationship between the CIP and LBBD models: as the LBBD sub-problem consists of a single `cumulative` constraint, any linear sub-problem relaxation used in the LBBD master problem can be adapted for the `optcumulative` relaxation in the CIP model. However, there are three primary differences between the ways in which the two approaches behave:

1. In the CIP models, the `optcumulative` constraint is propagated *during* the search through the resource assignment variables while in LBBD the `cumulative` propagation only occurs during sub-problem solving.
2. In LBBD, the sub-problems are solved independently while that decomposition is not visible to the CIP models.
3. The hand-crafted Benders cuts in LBBD are likely much stronger than the no-goods derived by conflict analysis in CIP.

8 Conclusion

In this paper, we conducted two related studies. First, we replicated an experiment with three existing optimization models for a resource allocation and scheduling problem: mixed integer programming, constraint programming, and logic-base Benders decomposition. We used modern commercial solvers for the former two models and demonstrated that the progress in commercial MIP and CP solvers means that the decomposition-based approach is no longer the dominant approach to such problems. Furthermore, our results indicate that MIP models are, at the least, competitive with other existing scheduling models. The results showed that the CP model can quickly find high quality solutions over the whole test set. Whereas the MIP model is able to provide strong lower bounds. As CP scheduling researchers have tended to discount the usefulness of MIP for scheduling problems, these results suggest that we should reconsider MIP as one of the core technologies to solve scheduling problems.

Subsequently, motivated by our first experiment, we introduced two constraint integer programming (CIP) models for the same scheduling problem and compared them to MIP, CP and LBBD models. The basic goal was to couple the fast detection of feasible solutions with the strong lower bound computation. Our results demonstrated that on problems with unary capacity resources, both CIP models are able to solve more problems to optimality that any of the other approaches. On problems with non-unary resource capacity, both CIP models again out-performed the other models in terms of number of instances for which the optimal was found and proved and, for one CIP model, in terms of the quality of the solutions for the instances not solved to optimality. As the LBBD results presented are weaker than previous results [28,13], we conservatively conclude that the CIP models are at the least competitive with the state-of-the-art and represent the current best non-decomposition-based approaches to these problems. We believe that our results demonstrate that constraint integer programming is a promising technology for scheduling in general and therefore plan to pursue its application to a variety of scheduling problems.

References

1. Baptiste, P., Pape, C.L., Nuijten, W.: Constraint-based Scheduling. Kluwer Academic Publishers (2001)
2. Bartak, R., Salido, M.A., Rossi, F.: New trends on constraint satisfaction, planning, and scheduling: a survey. The Knowledge Engineering Review **25**(3) (2010) 249–279
3. Koch, T., Achterberg, T., Andersen, E., Bastert, O., Berthold, T., Bixby, R.E., Danna, E., Gamrath, G., Gleixner, A.M., Heinz, S., Lodi, A., Mittelmann, H., Ralphs, T., Salvagnin, D., Steffy, D.E., Wolter, K.: MIPLIB 2010. Mathematical Programming Computation **3**(2) (2011) 103–163
4. Milano, M., Van Hentenryck, P., eds.: Hybrid Optimization: The Ten Years of CPAIOR. Springer (2010)
5. Hooker, J.N., Ottosson, G.: Logic-based Benders decomposition. Mathematical Programming **96** (2003) 33–60
6. Hooker, J.N.: Planning and scheduling to minimize tardiness. In van Beek, P., ed.: Principles and Practice of Constraint Programming – CP 2005. Volume 3709 of LNCS., Springer (2005) 314–327
7. Achterberg, T.: Constraint Integer Programming. PhD thesis, Technische Universität Berlin (2007)
8. Yunes, T.H., Aron, I.D., Hooker, J.N.: An integrated solver for optimization problems. Operations Research **58**(2) (2010) 342–356
9. Beck, J.C., Refalo, P.: A hybrid approach to scheduling with earliness and tardiness costs. Annals of Operations Research **118** (2003) 49–71
10. Berthold, T., Heinz, S., Lübbecke, M.E., Möhring, R.H., Schulz, J.: A constraint integer programming approach for resource-constrained project scheduling. In Lodi, A., Milano, M., Toth, P., eds.: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2010). Volume 6140 of LNCS., Springer (2010) 313–317
11. Heinz, S., Beck, J.C.: Solving resource allocation/scheduling problems with constraint integer programming. In Salido, M.A., Barták, R., Policella, N., eds.: Proceedings of the Workshop on Constraint Satisfaction Techniques for Planning and Scheduling Problems (COPLAS 2011). (2011) 23–30
12. Achterberg, T.: SCIP: Solving Constraint Integer Programs. Mathematical Programming Computation **1**(1) (2009) 1–41
13. Beck, J.C.: Checking-up on branch-and-check. In Cohen, D., ed.: Principles and Practice of Constraint Programming – CP 2010. Volume 6308 of LNCS., Springer (2010) 84–98
14. Wunderling, R.: Paralleler und objektorientierter Simplex-Algorithmus. PhD thesis, Technische Universität Berlin (1996)
15. Heinz, S., Beck, J.C.: Reconsidering mixed integer programming and MIP-based hybrids for scheduling. ZIB-Report 12-05, Zuse Institute Berlin (2012)
16. Hooker, J.N.: Integrated Methods for Optimization. Springer (2007)
17. Berthold, T., Heinz, S., Vigerske, S.: Extending a CIP framework to solve MIQCPs. ZIB-Report 09-23, Zuse Institute Berlin (2009)
18. Berthold, T., Heinz, S., Pfetsch, M.E.: Nonlinear pseudo-boolean optimization: relaxation or propagation? In Kullmann, O., ed.: Theory and Applications of Satisfiability Testing – SAT 2009. Volume 5584 of LNCS., Springer (2009) 441–446
19. Achterberg, T., Brinkmann, R., Wedler, M.: Property checking with constraint integer programming. ZIB-Report 07-37, Zuse Institute Berlin (2007)
20. Beck, J.C., Fox, M.S.: Constraint directed techniques for scheduling with alternative activities. Artificial Intelligence **121**(1–2) (2000) 211–250

21. Debruyne, R., Bessière, C.: Some practicable filtering techniques for the constraint satisfaction problem. In: Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI97). (1997) 412–417
22. Martin, P., Shmoys, D.B.: A new approach to computing optimal schedules for the job-shop scheduling problem. In: Proceedings of the Fifth Conference on Integer Programming and Combinatorial Optimization (IPCO 1996). Volume 1084 of LNCS., Springer (1996) 389–403
23. Achterberg, T., Berthold, T.: Hybrid branching. In van Hoeve, W.J., Hooker, J.N., eds.: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2009). Volume 5547 of LNCS., Springer (2009) 309–311
24. Marques-Silva, J.P., Sakallah, K.A.: GRASP: A search algorithm for propositional satisfiability. IEEE Transactions on Computers **48**(5) (1999) 506–521
25. Achterberg, T.: Conflict analysis in mixed integer programming. Discrete Optimization **4**(1) (2007) 4–20 Special issue: Mixed Integer Programming.
26. Schutt, A., Feydy, T., Stuckey, P., Wallace, M.: Explaining the cumulative propagator. Constraints (2010) 1–33
27. Heinz, S., Schulz, J.: Explanations for the cumulative constraint: An experimental study. In Pardalos, P.M., Rebennack, S., eds.: Experimental Algorithms. Volume 6630 of LNCS., Springer (2011) 400–409
28. Hooker, J.N.: Planning and scheduling by logic-based Benders decomposition. Operations Research **55** (2007) 588–602