Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

STEFAN HEINZ* J. CHRISTOPHER BECK**

Reconsidering Mixed Integer Programming and MIP-based Hybrids for Scheduling

* Supported by the DFG Research Center MATHEON *Mathematics for key technologies* in Berlin.
** Department of Mechanical & Industrial Engineering, University of Toronto, Toronto, Canada

ZIB-Report 12-05 (February 2012)

Herausgegeben vom
Konrad-Zuse-Zentrum für Informationstechnik Berlin
Takustraße 7
D-14195 Berlin-Dahlem

Telefon: 030-84185-0
Telefax: 030-84185-125

e-mail: bibliothek@zib.de
URL: http://www.zib.de

ZIB-Report (Print) ISSN 1438-0064
ZIB-Report (Internet) ISSN 2192-7782

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 a parallel line of research, 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 a number of resource allocation and scheduling problems are based around logic-based Benders decomposition (LBBD) [5,6]. This loosely coupled hybrid approach seeks to decompose the global problem into a master problem and

⋆ 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].

a set of sub-problems, and then 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 approaches to hybridization that seek to combine the key elements of 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 are now substantially better than previously shown [6,11]. Overall, the improvements of MIP solvers lead to significantly better performance than both CP and LBBD. Second, based on our observations from this experiment, 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. Further experiments show that both CIP models achieve performance better than the three previous models, both in terms of the number of problems solved to optimality and the solving times.

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 problems are defined by a set of jobs, $\mathcal{J}$, and a set of resources, $\mathcal{K}$. Each job has a release date, $\mathcal{R}_j$, a deadline, $\mathcal{D}_j$, a resource-specific processing time, p_{jk}, a resource assignment cost, c_{jk}, and a resource requirement, r_{jk}. Each job, j, must be assigned to a resource, k, and scheduled to start at or after its release date, end at or before its due date, and execute for p_{jk} consecutive time units. Each resource, $k \in \mathcal{K}$, has a capacity, C_k, and an associated constraint which states that for each time point, the sum of the resource requirements of the executing jobs must not exceed the resource capacity. A feasible solution is an assignment where each job is placed on exactly one resource and

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

Model 1. Constraint programming model.

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.

In the UNARY problem, the capacity of each resource and the requirement of each job is one. In the MULTI problem, capacities and requirements may be non-unary.

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]

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