Single-Facility Scheduling over Long Time Horizons by Logic-based Benders Decomposition

Elvin Coban and J. N. Hooker

Tepper School of Business, Carnegie Mellon University
ecoban@andrew.cmu.edu, john@hooker.tepper.cmu.edu

Abstract. Logic-based Benders decomposition can combine mixed integer programming and constraint programming to solve planning and scheduling problems much faster than either method alone. We find that a similar technique can be beneficial for solving pure scheduling problems as the problem size scales up. We solve single-facility non-preemptive scheduling problems with time windows and long time horizons that are divided into segments separated by shutdown times (such as weekends). The objective is to find feasible solutions, minimize makespan, or minimize total tardiness.

1 Introduction

Logic-based Benders decomposition has been successfully used to solve planning and scheduling problems that naturally decompose into an assignment and a scheduling portion. The Benders master problem assigns jobs to facilities using mixed integer programming (MILP), and the subproblems use constraint programming (CP) to schedule jobs on each facility.

In this paper, we use a similar technique to solve pure scheduling problems with long time horizons. Rather than assign jobs to facilities, the master problem assigns jobs to segments of the time horizon. The subproblems schedule jobs within each time segment.

In particular, we solve single-facility scheduling problems with time windows in which the objective is to find a feasible solution, minimize makespan, or minimize total tardiness. We assume that each job must be completed within one time segment. The boundaries between segments might therefore be regarded as weekends or shutdown times during which jobs cannot be processed. In future research we will address instances in which jobs can overlap two or more segments.

Logic-based Benders decomposition was introduced in [2, 8]. Its application to assignment and scheduling via CP/MILP was proposed in [3] and implemented in [9]. This and subsequent work shows that the Benders approach can be orders of magnitude faster than stand-alone MILP or CP methods on problems of this kind [1, 7, 4–6, 10, 11]. For the pure scheduling problems considered here, we find that the advantage of Benders over both CP and MILP increases rapidly as the problem scales up.

2 The Problem

Each job j has release time, deadline (or due date) d_j, and processing time p_j. The time horizon consists of intervals $[z_i, z_{i+1}]$ for $i = 1, \ldots, m$. The problem is to assign each job j a start time s_j so that time windows are observed ($r_j \leq s_j \leq d_j - p_j$), jobs run consecutively ($s_j + p_j \leq s_k$ or $s_k + p_k \leq s_j$ for all $k \neq j$), and each job is completed within one segment ($z_i \leq s_j \leq z_{i+1} - p_j$ for some i). We minimize makespan by minimizing $\max_j\{s_j + p_j\}$. To minimize tardiness, we drop the constraint $s_j \leq d_j - p_j$ and minimize $\sum_j \max\{0, s_j + p_j - d_j\}$.

3 Feasibility

When the goal is to find a feasible schedule, the master problem seeks a feasible assignment of jobs to segments, subject to the Benders cuts generated so far. Because we solve the master problem with MILP, we introduce 0-1 variables y_{ij} with $y_{ij} = 1$ when job j is assigned to segment i. The master problem becomes

$$\begin{aligned} &\sum_i y_{ij} = 1, \quad \text{all } j \\ &\text{Benders cuts, relaxation} \\ &y_{ij} \in \{0,1\}, \quad \text{all } i, j \end{aligned} \tag{1}$$

The master problem also contains a relaxation of the subproblem, similar to those described in [4–6], that helps reduce the number of iterations.

Given a solution $\bar{y}_{ij}$ of the master problem, let $J_i = \{j \mid \bar{y}_{ij} = 1\}$ be the set of jobs assigned to segment i. The subproblem decomposes into a CP scheduling problem for each segment i:

$$\begin{aligned} &\left.\begin{aligned} r_j \leq s_j \leq d_j - p_j \\ z_i \leq s_j \leq z_{i+1} - p_j \end{aligned}\right\}, \text{all } j \in J_i \\ &\texttt{disjunctive}\,(\{s_j \mid j \in J_i\}) \end{aligned} \tag{2}$$

where the `disjunctive` global constraint ensures that the jobs assigned to segment i do not overlap.

Each infeasible subproblem generates a Benders cut as described below, and the cuts are added to the master problem. The master problem and corresponding subproblems are repeatedly solved until every segment has a feasible schedule, or until the master problem is infeasible, in which case the original problem is infeasible.

Strengthened nogood cuts. The simplest Benders cut is a nogood cut that excludes assignments that cause infeasibility in the subproblem. If there is no feasible schedule for segment i, we generate the cut

$$\sum_{j \in J_i} y_{ij} \leq |J_i| - 1, \quad \text{all } i \tag{3}$$

The cut can be strengthened by removing jobs one by one from J_i until a feasible schedule exists for segment i. This requires re-solving the ith subproblem repeatedly,

but the effort generally pays off because the subproblems are much easier to solve than the master problem. We now generate a cut (3) with the reduced J_i.

The cut may be stronger if jobs less likely to cause infeasibility are removed from J_i first. Let the *effective time window* $[\tilde{r}_{ij}, \tilde{d}_{ij}]$ of job j on segment i be its time window adjusted to reflect the segment boundaries. Thus

$$\tilde{r}_{ij} = \max\{\min\{r_j, z_{i+1}\}, z_i\}, \quad \tilde{d}_{ij} = \min\{\max\{d_j, z_i\}, z_{i+1}\}$$

Let the *slack* of job j on segment i be $\tilde{d}_{ij} - \tilde{r}_{ij} - p_j$. We can now remove the jobs in order of decreasing slack.

4 Minimizing Makespan

Here the master problem minimizes μ subject to (1) and $\mu \geq 0$. The subproblems minimize μ subject to (2) and $\mu \geq s_j + p_j$ for all $j \in J_i$.

Strengthened nogood cuts. When one or more subproblems are infeasible, we use strengthened nogood cuts (3). Otherwise, for each segment i we use the nogood cut

$$\mu \geq \mu_i^* \left(1 - \sum_{j \in J_i} (1 - y_{ij})\right)$$

where μ_i^* is the minimum makespan for subproblem i. These cuts are strengthened by removing jobs from J_i until the minimum makespan on segment i drops below μ_i^*.

We also strengthen the cuts as follows. Let $\mu_i(J)$ be the minimum makespan that results when in jobs in J are assigned to segment i, so that in particular $\mu_i(J_i) = \mu_i^*$. Let Z_i be the set of jobs that can be removed, one at a time, without affecting makespan, so that $Z_i = \{j \in J_i \mid M_i(J_i \setminus \{j\}) = M_i^*\}$. Then for each i we have the cut

$$\mu \geq \mu_i(J_i \setminus Z_i) \left(1 - \sum_{j \in J_i \setminus Z_i} (1 - y_{ij})\right)$$

This cut is redundant and should be deleted when $\mu_i(J_i \setminus Z_i) = \mu_i^*$.

Analytic Benders Cuts. We can develop additional Benders as follows. Let $J_i' = \{j \in J_i \mid r_j \leq z_i\}$ be the set of jobs in J_i with release times before segment i, and let $J_i'' = J_i \setminus J_i'$. Let $\hat{\mu}_i$ be the minimum makespan of the problem that remains after removing the jobs in $S \subset J_i'$ from segment i. It can be shown as in [6] that

$$\mu_i^* - \hat{\mu}_i \leq p_S + \max_{j \in J_i'}\{\tilde{d}_j\} - \min_{j \in J_i'}\{\tilde{d}_j\} \tag{4}$$

where $p_S = \sum_{j \in S} p_j$. Thus if jobs in J_i' are removed from segment i, we have from (4) a lower bound on the resulting optimal makespan $\hat{\mu}_i$. If jobs in J_i'' are removed, there is nothing we can say. So we have the following Benders cut for each i:

$$\mu \geq \mu_i^* - \left(\sum_{j \in J_i'} p_j(1 - y_{ij}) + \max_{j \in J_i'}\{d_j\} - \min_{j \in J_i'}\{d_j\}\right) - \sum_{j \in J_i''} \mu_i^*(1 - y_{ij}) \tag{5}$$

when one or more jobs are removed from segment i, $\mu \geq 0$ when all jobs are removed, and $\mu \geq \mu_i^*$ otherwise. This can be linearized:

$$\mu \geq \mu_i^* - \sum_{j \in j_i'} p_j(1 - y_{ij}) - w_i - \sum_{j \in J_i''} \mu_i^*(1 - y_{ij}) - \mu_i^* q_i, \quad q_i \leq 1 - y_{ij},\ j \in J_i$$

$$w_i \leq \left(\max_{j \in J_i'}\{d_j\} - \min_{j \in J_i'}\{d_j\} \right) \sum_{j \in J_i'} (1 - y_{ij}), \qquad w_i \leq \max_{j \in J_i'}\{d_j\} - \min_{j \in J_i'}\{d_j\}$$

5 Minimizing Tardiness

Here the master problem minimizes τ subject to (1), and each subproblem minimizes $\sum_{j \in J_i} \tau_j$ subject to $\tau_j \geq s_j + p_j - d_j$ and $\tau_j \geq 0$.

Benders cuts. We use strengthened nogood cuts and relaxations similar to those used for minimizing makespan. We also develop the analytic Benders cuts

$$\tau \geq \sum_i \hat{\tau}_i$$

$$\hat{\tau}_i \geq \begin{cases} \tau_i^* - \sum_{j \in J_i} \left(r_i^{\max} + \sum_{\ell \in J_i} p_\ell - d_j \right)^+ (1 - y_{ij}), & \text{if } r_i^{\max} + \sum_{\ell \in J_i} p_\ell \leq z_{i+1} \\ \tau_i^* \left(1 - \sum_{j \in J_i} (1 - y_{ij}) \right), & \text{otherwise} \end{cases}$$

where the bound on $\hat{\tau}_i$ is included for all i for which $\tau_i^* > 0$. Here τ_i^* is the minimum tardiness in subproblem i, $r_i^{\max} = \max\{\max\{r_j \mid j \in J_i\}, z_i\}$, and $\alpha^+ = \max\{0, \alpha\}$.

6 Problem Generation and Computational Results

Random instances are generated as follows. For each job j, r_j, $d_j - r_j$, and p_j are uniformly distributed on the intervals $[0,\ \alpha R]$, $[\gamma_1 \alpha R,\ \gamma_2 \alpha R]$, and $[0,\ \beta(d_j - r_j)]$, respectively. We set $R = 40\ m$ for tardiness problems, and otherwise $R = 100\ m$, where m is the number of segments. For the feasibility problem we adjusted β to provide a mix of feasible and infeasible instances. For the remaining problems, we adjusted β to the largest value for which most of the instances are feasible.

We formulated and solved the instances with IBM's OPL Studio 6.1, which invokes the ILOG CP Optimizer for CP models and CPLEX for MILP models. The MILP models are discrete-time formulations we have found to be most effective for this type of problem. We used OPL's script language to implement the Benders method.

Table 1 shows the advantage of logic-based Benders as the problem scales up. Benders failed to solve only four instances, due to inability to solve the CP subproblems.

Table 1. Computation times in seconds (computation terminated after 600 seconds). The number of segments is 10% the number of jobs. Tight time windows have $(\gamma_1, \gamma_2, \alpha) = (1/2, 1, 1/2)$ and wide time windows have $(\gamma_1, \gamma_2, \alpha) = (1/4, 1, 1/2)$. For feasibility instances, $\beta = 0.028$ for tight windows and 0.035 for wide windows. For makespan instances, $\beta = 0.025$ for 130 or fewer jobs and 0.032 otherwise. For tardiness instances, $\beta = 0.05$.

	Tight time windows									Wide time windows								
	Feasibility			Makespan			Tardiness			Feasibility			Makespan			Tardiness		
Jobs	CP	MILP	Bndrs	CP	MILP	Bndrs	CP	MILP	Bndrs	CP	MILP	Bndrs	CP	MILP	Bndrs	CP	MILP	Bndrs
50	0.91	8.0	1.5	0.09	9.0	4.0	0.05	1.3	1.1	0.03	7.7	2.5	0.13	13	3.5	0.13	1.3	1.1
60	1.1	12	2.8	0.09	18	5.5	0.14	1.8	1.5	0.05	12	1.6	0.94	29	5.7	0.11	2.3	1.4
70	0.56	17	3.3	0.11	51	6.7	1.3	3.9	2.1	0.13	17	2.3	0.11	39	6.2	0.16	3.0	1.9
80	600	21	2.8	600	188	7.6	0.86	6.0	4.5	600	24	5.0	600	131	7.3	1.9	6.4	5.0
90	600	29	7.5	600	466	10	21	11	4.6	600	32	9.7	600	600	8.5	5.9	9.5	11
100	600	36	12	600	600	16	600	11	2.0	600	44	9.7	600	600	19	600	24	22
110	600	44	20	600	600	17	600	600	600	600	49	17	600	600	24	600	600	600
120	600	62	18	600	600	21	600	15	3.3	600	80	15	600	600	23	600	12	3.1
130	600	68	20	600	600	29	600	17	3.9	600	81	43	600	600	31	600	18	3.9
140	600	88	21	600	600	30	600	600	600	600	175	27	600	*	35	600	600	14
150	600	128	27	600	600	79				600	386	8.5	600	600	43			
160	600	408	82	600	600	34				600	174	5.2	600	600	53			
170	600	192	5.9	600	600	37				600	172	5.9	600	600	600			
180	600	600	6.6	600	*	8.0				600	251	6.5	600	600	56			
190	600	600	7.2	600	*	8.5				600	600	7.3	600	*	78			
200	600	600	8.0	600	*	85				600	600	8.2	600	*	434			

*MILP solver ran out of memory.

References

1. I. Harjunkoski and I. E. Grossmann. Decomposition techniques for multistage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering*, 26:1533–1552, 2002.
2. J. N. Hooker. Logic-based benders decomposition. Technical report, CMU, 1995.
3. J. N. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York, 2000.
4. J. N. Hooker. A hybrid method for planning and scheduling. *Constraints*, 10:385–401, 2005.
5. J. N. Hooker. An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11:139–157, 2006.
6. J. N. Hooker. Planning and scheduling by logic-based benders decomposition. *Operations Research*, 55:588–602, 2007.
7. J. N. Hooker and G. Ottosson. Logic-based benders decomposition. *Mathematical. Programming*, 96:33–60, 2003.
8. J. N. Hooker and H. Yan. *Logic circuit verification by Benders decomposition*, pages 267–288. Principles and Practice of Constraint Programming: The Newport Papers, MIT Press (Cambridge, MA, 1995), 1995.
9. V. Jain and I. E. Grossmann. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing*, 13(4):258–276, 2001.
10. C. T. Maravelias and I. E. Grossmann. A hybrid MILP/CP decomposition approach for the continuous time scheduling of multipurpose batch plants. *Computers and Chemical Engineering*, 28:1921–1949, 2004.
11. C. Timpe. Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum*, 24:431–448, 2002.