

Single-Facility Scheduling by Logic-Based Benders Decomposition

Elvin Coban J. N. Hooker

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

September 2011

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. The Benders master problem assigns jobs to predefined segments of the time horizon, where the subproblem schedules them. In one version of the problem, jobs may not overlap the segment boundaries (which represent shutdown times, such as weekends), and in another version, there is no such restriction. The objective is to find feasible solutions, minimize makespan, or minimize total tardiness.

1 Introduction

Logic-based Benders decomposition [27] is a generalization of Benders decomposition that accommodates a much wider range of problems. In contrast with the classical Benders method, the subproblem can in principle be any combinatorial problem, not necessarily a linear or nonlinear programming problem. For example, it can be a scheduling problem solved by constraint programming (CP), a method well suited to scheduling.

This flexibility has led to the application of logic-based Benders decomposition to 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 subproblem uses CP to schedule jobs on each facility. This approach can reduce solution time by several orders of magnitude relative to methods that use MILP or CP alone [21, 22, 23, 24, 26, 29, 43].

In this paper, we investigate whether a similar technique can solve pure scheduling problems, which lack the obvious decomposition that one finds in planning and scheduling. Rather than break the problem into facility assignment and job scheduling, we decompose it into smaller scheduling subproblems on segments of the time horizon. The master problem assigns jobs to time segments, and subproblems schedule jobs within each segment. This allows us to deal with long time horizons that would otherwise make the problem intractable.

In particular, we solve single-facility non-preemptive scheduling problems with time windows in which the objective is to find a feasible solution, minimize makespan, or minimize total tardiness. In one version of the problem, which we call the *segmented* problem, 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 a second version of the problem, which we refer to as *unsegmented*, jobs can overlap two or more segments. We address both variants.

Obvious cases of single-facility scheduling include machine scheduling in a manufacturing plant or task scheduling in a computer with one processor. Less obvious cases occur when a complex plant is scheduled as one facility, as for example when a paint manufacturing plant must be committed to produce one color at a time [13]. A multistage process with a single bottleneck may also result in a single-facility model [37].

The paper is organized as follows. After a survey of previous work, we present a brief introduction to logic-based Benders decomposition, and state pure MILP and CP models for the single-facility scheduling problem. We then describe a logic-based Benders approach to the segmented feasibility, makespan, and tardiness problems. We do the same for the unsegmented feasibility and makespan problems, for which the Benders cuts are considerably more complex. We then present computational results. We conclude that the relative advantage of the Benders approach increases rapidly as the time horizon and number of jobs grow larger, particularly for segmented feasibility and makespan problems. The Benders method is not necessarily faster on unsegmented instances, but it is more robust and the only method to solve them all.

2 Previous Work

Logic-based Benders decomposition was introduced in [18, 28], and a general theory was presented in [20, 27]. Application to planning and scheduling was proposed in [20] and first implemented in [29].

Classical Benders decomposition derives Benders cuts from dual or Lagrange multipliers in the subproblem [5, 15]. However, this presupposes that the subproblem is a linear or nonlinear programming problem. Logic-based Benders decomposition has the advantage that Benders cuts can, at least in principle, be obtained from a subproblem of any form by solving its *inference dual* [19]. The solution of the dual is a *proof* of optimality for fixed values of the master problem variables (whence the name “logic-based”). The core idea of Benders

decomposition is that *this same proof* may establish a bound on the optimal value when the master problem variables take other values. The corresponding Benders cut enforces this bound in the master problem.

Logic-based Benders cuts must be designed specifically for each class of problems, but this provides an opportunity to exploit problem structure. The Benders framework is also natural for combining MILP and CP, because one method can be used to solve the master problem and the other the subproblem. This is particularly advantageous when the subproblem is a scheduling problem, for which CP methods are well suited [3, 25]. The combinatorial nature of the scheduling problem is no longer a barrier to generating Benders cuts.

Logic-based Benders cuts have some resemblance to cuts generated in oracle-based optimization (e.g., [2]) but differ in several respects. The Benders subproblem contains a different set of variables than the master problem and cuts, while in oracle-based optimization, the subproblem and cuts contain the same variables. The solution of the master problem, rather than a query point, defines the Benders subproblem, and Benders cuts can in principle be derived from any proof of optimality for the subproblem.

Additional applications of logic-based Benders include logic circuit testing [28], propositional satisfiability [27], multistage facility scheduling [17], dispatching of automated guided vehicles [11], steel production scheduling [16], real-time scheduling of computer processors [8], traffic diversion [9], batch scheduling in a chemical plant [35, 34] (and, in particular, polypropylene batch scheduling [44]), stochastic constraint programming [41], customer service with queuing [42], and scheduling of distributed processors for computation [6, 8].

In all of these applications, the subproblem is a feasibility problem rather than an optimization problem, which simplifies the task of designing Benders cuts. Effective Benders cuts can nonetheless be developed for optimization subproblems, based on somewhat deeper analysis of the inference dual. This is accomplished for planning and scheduling problems in [21, 22, 23, 24, 26], where the objective is to minimize makespan, the number of late jobs, or total tardiness. The subproblem is a cumulative scheduling problem, in which several jobs may be processed simultaneously subject to resource constraints.

Other applications of logic-based Benders to optimization include 0-1 programming [27, 9], mixed integer/linear programming [10], tournament scheduling [38, 39], location/allocation problems [12], shift selection with task sequencing [4], single- and multi-stage batch chemical processes [33], multimachine assignment scheduling with a branch-and-cut approach [40], and single-facility scheduling in the present paper. Temporal decomposition similar to that employed here is applied in [7] to large-scale vehicle routing problems with time windows. However, unlike a Benders method, the algorithm is heuristic and does not obtain provably optimal solutions. In addition, no cuts or nogoods are generated.

To our knowledge, logic-based Benders decomposition has not previously been applied to single-facility scheduling. A broad survey of single-facility scheduling methods for minimizing tardiness can be found in [32], which assumes that all release dates are equal. Four MILP formulations of single-facility

scheduling are analyzed in [31], where it is observed that the discrete-time model is most widely used and yields the tightest bounds. We use this model for our comparisons with pure MILP.

3 Logic-Based Benders Decomposition

Logic-based Benders decomposition is based on the concept of an *inference dual*. Consider an optimization problem

$$\begin{aligned} \min & f(x) \\ & C(x) \\ & x \in D \end{aligned} \tag{1}$$

where $C(x)$ represents a constraint set containing variables x , and D is the domain of x (such as \mathbb{R}^n or \mathbb{Z}^n). The inference dual is the problem of finding the tightest lower bound on the objective function that can be deduced from the constraints:

$$\begin{aligned} \max & v \\ & C(x) \stackrel{P}{\vdash} (f(x) \geq v) \\ & v \in \mathbb{R}, P \in \mathcal{P} \end{aligned} \tag{2}$$

Here $C(x) \stackrel{P}{\vdash} (f(x) \geq v)$ indicates that proof P deduces $f(x) \geq v$ from $C(x)$. The domain of variable P is a family \mathcal{P} of proofs, and the dual solution is a pair (v, P) . When the primal problem (1) is a feasibility problem with no objective function, the dual can be viewed as the problem finding a proof P of infeasibility.

If (1) is a linear programming (LP) problem $\min\{cx \mid Ax \geq b, x \geq 0\}$, the inference dual becomes the classical LP dual (assuming feasibility) for an appropriate proof family \mathcal{P} . Namely, each proof P corresponds to a tuple $u \geq 0$ of multipliers, and P deduces the bound $cx \geq v$ when the surrogate $uAx \geq ub$ dominates $cx \geq v$; that is, $uA \leq c$ and $ub \geq v$. The dual therefore maximizes v subject to $uA \leq c$, $ub \geq v$, and $u \geq 0$. Equivalently, it maximizes ub subject to $uA \leq c$ and $u \geq 0$, which is the classical LP dual.

Logic-based Benders decomposition applies to problems of the form

$$\begin{aligned} \min & f(x, y) \\ & C(x, y) \\ & x \in D_x, y \in D_y \end{aligned} \tag{3}$$

Fixing x to \bar{x} defines the subproblem

$$\begin{aligned} \min & f(\bar{x}, y) \\ & C(\bar{x}, y) \\ & y \in D_y \end{aligned} \tag{4}$$

Let proof P solve the inference dual of the subproblem by deducing the bound $f(\bar{x}, y) \geq v^*$. A Benders cut $v \geq B_{\bar{x}}(x)$ is derived by identifying a bound $B_{\bar{x}}(x)$

that the same proof P deduces for any given x . Thus, in particular, $B_{\bar{x}}(\bar{x}) = v^*$. The k th master problem is

$$\begin{aligned} \min \quad & v \\ & v \geq B_{x^i}(x), \quad i = 1, \dots, k-1 \\ & x \in D_x \end{aligned} \tag{5}$$

where x^1, \dots, x^{k-1} are the solutions of the first $k-1$ master problems. The optimal value v_k of the master problem is a lower bound on the optimal value of (3), and each $B_{x^i}(x^i)$ is an upper bound. The algorithm terminates when v_k is equal to the minimum of $B_{x^i}(x^i)$ over $i = 1, \dots, k$.

Classical Benders decomposition is the result of applying logic-based Benders decomposition to a problem of the form

$$\begin{aligned} \min \quad & f(x) + cy \\ & g(x) + Ay \geq b \\ & x \in D_x, \quad y \geq 0 \end{aligned} \tag{6}$$

The subproblem is an LP:

$$\begin{aligned} \min \quad & f(\bar{x}) + cy \\ & Ay \geq b - g(\bar{x}) \\ & y \geq 0 \end{aligned} \tag{7}$$

whose inference dual is the LP dual. Its solution u defines a surrogate $uAy \geq u(b - g(\bar{x}))$ that dominates $cy \geq v^*$ and therefore deduces that $f(\bar{x}) + cy \geq f(\bar{x}) + u(b - g(\bar{x}))$. The same u deduces $f(x) + cy \geq f(x) + u(b - g(x))$ for any x , and we have the classical Benders cut $v \geq f(x) + u(b - g(x))$. When the subproblem is infeasible, the dual has an extreme ray solution u that proves infeasibility because $uA \leq 0$ and $u(b - g(x)) > 0$. The Benders cut is therefore $u(b - g(x)) \leq 0$.

In practice, the solution of the subproblem inference dual is the proof of optimality obtained while solving the subproblem. The simplest type of Benders cut is a *nogood cut*, which states that the solution of the subproblem cannot be improved unless certain x_j 's are fixed to different values. For example, we might observe that only part of the master problem solution appears as premises in the optimality proof, perhaps $x_j = \bar{x}_j$ for $j \in J$. Then the optimal value of the subproblem is at least v^* so long as $x_j = \bar{x}_j$ for $j \in J$. This yields a nogood cut $v \geq B_{\bar{x}}(x)$ with

$$B_{\bar{x}}(x) = \begin{cases} v^* & \text{if } x_j = \bar{x}_j \text{ for } j \in J \\ -\infty & \text{otherwise} \end{cases}$$

If the subproblem is infeasible, then perhaps $x_j = \bar{x}_j$ for $j \in J$ appear as premises in the proof of infeasibility. A nogood cut states simply that $x_j \neq \bar{x}_j$ for some $j \in J$.

Further analysis of the optimality proof may yield *analytic* Benders cuts that provide useful bounds when $x_j \neq \bar{x}_j$ for some $j \in J$. We may also be able to infer valid cuts by re-solving the subproblem when some of the premises $x_j = \bar{x}_j$ are dropped. All of these techniques are illustrated below.

4 The Problem

In the unsegmented problem, there are n jobs to be processed, and each job j has release time r_j , deadline (or due date) d_j , and processing time p_j . If we let $J = \{1, \dots, n\}$, the problem is to assign each job $j \in J$ a start time s_j so that time windows are observed:

$$r_j \leq s_j \leq d_j - p_j, \quad j \in J$$

and jobs run consecutively:

$$s_j + p_j \leq s_k \text{ or } s_k + p_k \leq s_j, \quad \text{all } j, k \in J \text{ with } j \neq k$$

We minimize makespan by minimizing $\max_{j \in J} \{s_j + p_j\}$. To minimize tardiness, we drop the deadline constraints $s_j \leq d_j - p_j$ and minimize

$$\sum_{j \in J} \max\{s_j + p_j - d_j, 0\} \quad (8)$$

The segmented problem is the same except for the additional constraint that each job must be completed within one segment $[a_i, a_{i+1}]$ of the time horizon:

$$a_i \leq s_j \leq a_{i+1} - p_j \text{ for some } i \in I, \quad \text{all } j \in J$$

Notation is summarized in Table 1.

4.1 MILP Formulation

In the discrete-time MILP formulation of the problem, binary variable $z_{jt} = 1$ when job j starts at time t . Let T be the set of discrete times, and assume $r_j, d_j \in T$ for each j . The unsegmented problem is written

$$\min \text{ [objective function]} \quad (a)$$

$$\sum_{t \in T} z_{jt} = 1, \quad j \in J \quad (b)$$

$$\sum_{j \in J} \sum_{\bar{t} \in T_{jt}} z_{j\bar{t}} \leq 1, \quad t \in T \quad (c)$$

$$z_{jt} = 0, \quad \text{all } t \in T \text{ with } t < r_j, \quad j \in J \quad (d)$$

$$z_{jt} = 0, \quad \text{all } t \in T \text{ with } t > d_j - p_j, \quad j \in J \quad (e)$$

$$z_{jt} \in \{0, 1\}, \quad j \in J, t \in T \quad (f)$$

where $T_{jt} = \{\bar{t} \mid t - p_j + 1 \leq \bar{t} \leq t\}$. Constraint (b) requires that every job be assigned a start time. The clique inequality [14, 36] in (c) ensures that jobs do not overlap. Constraint (d) prevents a job from starting before its release time, and (e) prevents it from ending after its deadline.

Table 1: Nomenclature for the segmented problem.

<i>Indices</i>	
$j \in J$	job
$i \in I$	segment of the time horizon
$t \in T$	time
<i>Sets</i>	
J	set of jobs ($\{1, 2, \dots, n\}$)
I	set of segments
T	set of discrete times
T_{jt}	set of discrete times at which job j running at time t can start
J_i	set of jobs assigned to segment i by the master problem
$J(t_1, t_2)$	set of jobs whose time windows fall within $[t_1, t_2]$
<i>Parameters</i>	
r_j	release time of job j
d_j	deadline (or due date) of job j
p_j	processing time of job j
a_i	start time of segment i
\bar{y}_{ij}	solution value of y_{ij} in the master problem
\tilde{r}_{ij}	starting time of the effective time window of job j on segment i
\tilde{d}_{ij}	ending time of the effective time window of job j on segment i
ϵ_{ij}	slack of job j on segment i
\bar{r}	tuple of distinct release times
\bar{d}	tuple of distinct deadlines
M_i^*	minimum makespan of segment i
<i>Variables</i>	
s_j	start time of job j
z_{jt}	= 1 if job j starts at time t
M	makespan over all segments
M_i	makespan of segment i
T_j	tardiness of job j
v_j	segment in which job j is processed
y_{ij}	= 1 if job j is processed in segment i

The feasibility problem seeks a feasible solution and consists of the constraints (b)–(f). The makespan problem minimizes the makespan M subject to (b)–(f) and

$$M \geq (t + p_j)z_{jt}, \quad j \in J, t \in T$$

The tardiness problem minimizes $\sum_{j \in J} T_j$ subject to (b)–(d), (f), and

$$\begin{aligned} T_j &\geq \sum_{t \in T} (t + p_j)z_{jt} - d_j, \quad j \in J \\ T_j &\geq 0, \quad j \in J \end{aligned} \tag{9}$$

where T_j is the tardiness of job j .

The segmented problem consists of the above and the additional constraints

$$z_{jt} = 0, \quad \text{for } t = a_{i+1} - p_j + 1, \dots, a_{i+1} - 1, \text{ all } i \in I, j \in J \tag{10}$$

4.2 CP Formulation

In principle, a CP model of the unsegmented problem is quite simple. Again letting s_j be the start time of job j , a model is

$$\begin{aligned} \min & \text{ [objective function]} & (a) \\ r_j &\leq s_j \leq d_j - p_j, \quad j \in J & (b) \\ \text{noOverlap} &(s, p) & (c) \end{aligned} \tag{11}$$

where $s = (s_1, \dots, s_n)$ and $p = (p_1, \dots, p_n)$ in constraint (c), and where noOverlap is a global constraint that requires the jobs to run sequentially. The makespan problem minimizes $\max_{j \in J} \{s_j + p_j\}$ subject to (b)–(c). The tardiness problem minimizes (8) subject to (b)–(c) without the upper bound in (b).

The segmented problem can be formulated by introducing a variable v_j with domain I that indicates which segment job j is assigned. We then add to (11) the constraints

$$a_{v_j} \leq s_j \leq a_{v_j+1} - p_j, \quad j \in J$$

5 Segmented Feasibility Problem

We now apply logic-based Benders decomposition to the problem of finding a feasible schedule. The master problem assigns jobs to time segments, and the subproblem decouples into a scheduling problem for each segment. We first address the segmented problem, for which the time horizon is already divided into segments $[a_i, a_{i+1}]$ for $i \in I$.

5.1 Master Problem

The master problem is an MILP formulation in which binary variable $y_{ij} = 1$ when job j is assigned to segment i .

$$\sum_{i \in I} y_{ij} = 1, \quad j \in J$$

Benders cuts

Relaxation

$$y_{ij} \in \{0, 1\}, \quad j \in J, i \in I$$

Benders cuts are developed below. The master problem also contains a relaxation of the problem, expressed in terms of the master problem variables. This results in more reasonable assignments in the early stages of the Benders algorithm and therefore reduces the number of iterations. The relaxation is described in Section 5.3 below.

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

$$\left. \begin{array}{l} r_j \leq s_j \leq d_j - p_j \\ a_i \leq s_j \leq a_{i+1} - p_j \end{array} \right\}, \quad j \in J_i$$

noOverlap($s(i, \bar{y}), p(i, \bar{y})$)

where $s(i, \bar{y})$ is the tuple of variables s_j for $j \in J_i$, and similarly for $p(i, \bar{y})$. The subproblems are solved by CP.

In each iteration of the Benders algorithm, the master problem is solved for \bar{y} . If the resulting subproblem is feasible on every segment, the algorithm stops with a feasible solution. Otherwise, one or more Benders cuts are generated for each segment on which the scheduling subproblem is infeasible. These cuts are added to the master problem, and the process repeats. If at some point the master problem is infeasible, so is the original problem.

5.2 Benders Cuts

The simplest Benders cut for an infeasible segment i is a nogood cut that prevents assigning the same jobs (perhaps among others) to that segment in subsequent iterations:

$$\sum_{j \in J_i} (1 - y_{ij}) \geq 1$$

This cut is quite weak, however, because it can be satisfied by omitting just one job in J_i from the future assignments to segment i . The cut can be strengthened by identifying a smaller set $\bar{J}_i \subset J_i$ of jobs that appear as premises in the proof of infeasibility for segment i . This yields the cut

$$\sum_{j \in \bar{J}_i} (1 - y_{ij}) \geq 1$$

Unfortunately, standard CP solvers do not provide this kind of dual information. We therefore seek to identify heuristically a smaller set \bar{J}_i that creates infeasibility. A simple heuristic is to remove jobs from J_i one at a time, checking each time if the scheduling problem on segment i is still infeasible. If it becomes feasible, the job is restored to J_i (Algorithm 1). This requires repeated solution of each subproblem, but the time required to do so tends to be small in practice.

Algorithm 1: Strengthening nogood cuts for the feasibility problem.

```

Let  $\bar{J}_i = J_i = \{j_1, \dots, j_k\}$ ;
for  $\ell = 1, \dots, k$  do
    if the jobs in  $\bar{J}_i \setminus \{j_\ell\}$  cannot be scheduled on segment  $i$  then
        Remove  $j_\ell$  from  $\bar{J}_i$ 

```

This heuristic may be more effective if jobs less likely to lead to feasibility are removed 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

$$\begin{aligned}\tilde{r}_{ij} &= \max\{\min\{r_j, a_{i+1}\}, a_i\} \\ \tilde{d}_{ij} &= \min\{\max\{d_j, a_i\}, a_{i+1}\}\end{aligned}$$

Let the *slack* of job j on segment i be

$$\epsilon_{ij} = (\tilde{d}_{ij} - \tilde{r}_{ij}) - p_j$$

We can remove the jobs in order of decreasing slack, which means they are indexed so that

$$\epsilon_{i1} \geq \dots \geq \epsilon_{ik}$$

5.3 Relaxation

The convergence of a Benders method can often be accelerated by augmenting the master problem with some valid inequalities in the master problem variables, resulting in a relaxation of the original problem.

A simple relaxation in the present case requires that jobs be scheduled so that for every time interval $[t_1, t_2]$, the set $J(t_1, t_2)$ of jobs whose time windows fall within $[t_1, t_2]$ has total processing time at most $t_2 - t_1$. Thus

$$J(t_1, t_2) = \{j \in J \mid t_1 \leq r_j, d_j \leq t_2\}$$

It suffices to enumerate, for each segment, distinct intervals of the form $[r_j, d_k]$ for all j, k with $r_j \leq d_k$. So we have the relaxation

$$\sum_{\ell \in J(r_j, d_k)} p_\ell y_{i\ell} \leq \tilde{d}_{ik} - \tilde{r}_{ij}, \quad \text{all } i \in I \text{ and all distinct } [r_j, d_k]$$

Algorithm 2 generates a relaxation with fewer redundant constraints. Note that it fixes $y_{ij} = 0$ when job j 's time window does not overlap segment i .

Algorithm 2: Generating a relaxation for the segmented feasibility problem.

Let $\bar{r}_1, \dots, \bar{r}_t$ be the distinct elements of $\{r_1, \dots, r_n\}$;
Let $\bar{d}_1, \dots, \bar{d}_u$ be the distinct elements of $\{d_1, \dots, d_n\}$;
for all i **do**
 for all $j = 1, \dots, t$ **with** $a_i < \bar{r}_j < a_{i+1}$ **do**
 for all $k = 1, \dots, u$ **with** $\bar{d}_k < a_{i+1}$ **do**
 Generate the inequality $\sum_{\ell \in J(\bar{r}_j, \bar{d}_k)} p_\ell y_{i\ell} \leq \bar{d}_k - \bar{r}_j$
 Generate the inequality $\sum_{\ell \in J(\bar{r}_j, \infty)} p_\ell y_{i\ell} \leq a_{i+1} - \bar{r}_j$;
 for all $k = 1, \dots, u$ **with** $a_i < \bar{d}_k < a_{i+1}$ **do**
 Generate the inequality $\sum_{\ell \in J(0, \bar{d}_k)} p_\ell y_{i\ell} \leq \bar{d}_k - a_i$
 Generate the inequality $\sum_{\ell} p_\ell y_{i\ell} \leq a_{i+1} - a_i$;
 Set $y_{i\ell} = 0$ for all $\ell \in J(0, a_i)$;
 Set $y_{i\ell} = 0$ for all $\ell \in J(a_{i+1}, \infty)$;

Table 2: Additional nomenclature for the unsegmented problem.

<i>Sets</i>	
J_{i0}	set of jobs that are fully processed in segment i
<i>Parameters</i>	
\bar{x}_{ij}	solution value of x_{ij} in the master problem
\bar{y}_{ij}	solution value of y_{ij} in the master problem
\bar{y}_{ijk}	solution value of y_{ijk} in the master problem, for $k \in \{0, 1, 2, 3\}$
\bar{a}_i	updated start time of the segment i
<i>Variables</i>	
x_{ij}	amount of time job j is processed during segment i
y_{ij}	= 1 if at least a portion of job j is processed in segment i
y_{ij0}	= 1 if all of job j is processed in segment i
y_{ij1}	= 1 if a portion of job j is processed at the start of segment i
y_{ij2}	= 1 if a portion of job j is processed at the end of segment i
y_{ij3}	= 1 if a portion of job j is processed throughout segment i
α_i	= 1 if $x_{ij} \leq \bar{x}_{ij}$ and segment i begins while job j is in process
β_i	= 1 if $x_{ij} \leq \bar{x}_{ij}$ and segment i ends while job j is in process
γ_i	= 1 if $x_{ij_1} + x_{ij_2} \leq x_i^*$ and segment i begins while job j_1 is in process and ends while job j_2 is in process

6 Unsegmented Feasibility Problem

We now suppose that the jobs are not required to fit inside segments of the time horizon. We create segments $[a_i, a_{i+1}]$ solely for purposes of decomposition, which means that a job can overlap two or more segments. This leads to several cases that complicate the master problem and the Benders cuts.

6.1 Master Problem

Let continuous variable x_{ij} indicate the amount of time that job j processes during segment i . Then

$$\begin{aligned} \sum_{i \in I} x_{ij} &= p_j, \quad j \in J \\ x_{ij} &\geq 0, \quad i \in I, j \in J \end{aligned} \tag{12}$$

The assignment of jobs (or portions of jobs) to segments is governed by several binary variables as introduced in Table 2.

We have the constraints

$$\begin{aligned} \sum_{i \in I} y_{ij} &\geq 1, \quad j \in J & (a) \\ y_{ij} &= y_{ij0} + y_{ij1} + y_{ij2} + y_{ij3}, \quad i \in I, j \in J & (b) \\ \sum_{j \in J} y_{ij1} &\leq 1, \quad \sum_{j \in J} y_{ij2} \leq 1, \quad \sum_{j \in J} y_{ij3} \leq 1, \quad i \in I & (c) \\ y_{ij1} &\leq y_{i-1,j,2} + y_{i-1,j,3}, \quad i \in I, i > 1, j \in J & (d) \\ y_{ij2} &\leq y_{i+1,j,1} + y_{i+1,j,3}, \quad i \in I, i < n, j \in J & (e) \\ y_{ij3} &\leq y_{i-1,j,3} + y_{i-1,j,2}, \quad i \in I, i > 1, j \in J & (f) \\ y_{ij3} &\leq y_{i+1,j,3} + y_{i+1,j,1}, \quad i \in I, i < n, j \in J & (g) \\ \sum_{i \in I} y_{ij0} &\leq 1, \quad \sum_{i \in I} y_{ij1} \leq 1, \quad \sum_{i \in I} y_{ij2} \leq 1, \quad j \in J & (h) \\ y_{1j1} &= y_{1j3} = y_{nj2} = y_{nj3} = 0, \quad j \in J & (i) \\ \sum_{i \in I} y_{ij3} &\leq \left\lfloor \frac{p_j}{a_{i+1} - a_i} \right\rfloor, \quad j \in J & (j) \\ y_{ij}, y_{ij0}, y_{ij1}, y_{ij2}, y_{ij3} &\in \{0, 1\}, \quad i \in I, j \in J & (k) \end{aligned} \tag{13}$$

Constraint (a) ensures that every job is assigned to at least one segment. Constraints (b) define y_{ij} . Constraints (c) ensure that at most one partial job is processed first, last, or throughout a segment. Constraints (d)–(g) require contiguity for the portions of a job. Constraints (h) say that a job can start, finish, or execute completely in at most one segment. Constraints (i) give boundary conditions. Constraints (j) are redundant but tighten the continuous relaxation of the MILP formulation.

To connect x_{ij} with the binary variables, note that we have the following disjunction for each i, j :

$$\begin{pmatrix} y_{ij} = 0 \\ x_{ij} = 0 \end{pmatrix} \vee \begin{pmatrix} y_{ij0} = 1 \\ x_{ij} = p_j \end{pmatrix} \vee \begin{pmatrix} y_{ij1} = 1 \\ x_{ij} \leq p_j \end{pmatrix} \vee \begin{pmatrix} y_{ij2} = 1 \\ x_{ij} \leq p_j \end{pmatrix} \vee \begin{pmatrix} y_{ij3} = 1 \\ x_{ij} = a_{i+1} - a_i \end{pmatrix}$$

Using the standard convex hull formulation of a disjunction of linear systems [30], this becomes:

$$\begin{aligned} x_{ij1} &\leq p_j y_{ij1}, & x_{ij2} &\leq p_j y_{ij2} \\ x_{ij} &= p_j y_{ij0} + x_{ij1} + x_{ij2} + (a_{i+1} - a_i) y_{ij3} \\ x_{ij1}, x_{ij2} &\geq 0 \end{aligned} \tag{14}$$

The master problem consists of (12)–(13), (14) for all $i \in I$ and $j \in J$, Benders cuts, and a relaxation. The Benders cuts are described in the next section.

To formulate the subproblem, suppose that the solution of the current master problem is $\bar{y}_{ij}, \bar{y}_{ij0}, \bar{y}_{ij1}, \bar{y}_{ij2}, \bar{x}_{ij}$ for all i, j . Define

$$\begin{aligned} J_i &= \{j \mid \bar{y}_{ij} = 1\} \\ J_{i0} &= \{j \mid \bar{y}_{ij0} = 1\} \end{aligned}$$

The subproblem must take into account whether the master problem assigned certain jobs to begin or end a segment. A general formulation of the subproblem on segment i is

$$\left. \begin{aligned} r_j &\leq s_j \leq d_j - p_j \\ \bar{a}_i &\leq s_j \leq \bar{a}_{i+1} - p_j \end{aligned} \right\}, \quad j \in J_{i0} \tag{15}$$

noOverlap($s(i, \bar{y}_0), p(i, \bar{y}_0)$)

where

$$\begin{aligned} \bar{a}_i &= \begin{cases} a_i + \bar{x}_{ij_1}, & \text{if } \bar{y}_{ij_11} = 1 \text{ for some } j_1 \in J \\ a_i, & \text{otherwise} \end{cases} \\ \bar{a}_{i+1} &= \begin{cases} a_{i+1} - \bar{x}_{ij_2}, & \text{if } \bar{y}_{ij_22} = 1 \text{ for some } j_2 \in J \\ a_{i+1}, & \text{otherwise} \end{cases} \end{aligned}$$

Also $s(i, \bar{y}_0)$ is the tuple of variables s_j for $j \in J_{i0}$, and similarly for $p(i, \bar{y}_0)$. Note that if a portion of some job j_3 spans the entire segment ($\bar{y}_{ij_33} = 1$), the constraint set is empty.

The relaxation is similar to that for the segmented problem but must account for jobs that run in two or more segments. The relaxation is generated by Algorithm 3.

6.2 Benders Cuts

The Benders cuts generated for an infeasible segment i depend on whether there are partial jobs assigned to the beginning or end of the segment.

Algorithm 3: Generating a relaxation for the unsegmented feasibility problem.

Let $\bar{r}_1, \dots, \bar{r}_t$ be the distinct elements of $\{r_1, \dots, r_n\}$;
Let $\bar{d}_1, \dots, \bar{d}_u$ be the distinct elements of $\{d_1, \dots, d_n\}$;
for all i do
 for all $j = 1, \dots, t$ with $a_i < \bar{r}_j < a_{i+1}$ do
 for all $k = 1, \dots, u$ with $\bar{d}_k < a_{i+1}$ do
 Generate the inequality $\sum_{\ell \in J(\bar{r}_j, \bar{d}_k)} p_{\ell} y_{i\ell} \leq \bar{d}_k - \bar{r}_j$
 Generate the inequality $\sum_{\ell \in J(\bar{r}_j, \infty)} x_{i\ell} \leq a_{i+1} - \bar{r}_j$;
 for all $k = 1, \dots, u$ with $a_i < \bar{d}_k < a_{i+1}$ do
 Generate the inequality $\sum_{\ell \in J(0, \bar{d}_k)} x_{i\ell} \leq \bar{d}_k - a_i$
 Generate the inequality $\sum_{\ell} x_{i\ell} \leq a_{i+1} - a_i$;
 Set $y_{i\ell} = 0$ for all $\ell \in J(0, a_i)$;
 Set $y_{i\ell} = 0$ for all $\ell \in J(a_{i+1}, \infty)$;

- *Case 1.* There are no partial jobs in segment i . Then we can use the simple nogood cut

$$\sum_{j \in J_{i0}} (1 - y_{ij0}) \geq 1 \quad (16)$$

This can be strengthened as in the segmented problem by removing some jobs that are not necessary for infeasibility (Algorithm 1 with J_{i0} replacing J_i).

- *Case 2.* There is a partial job only at the start of the segment (say, job j_1). We solve a modified subproblem by maximizing x_{ij_1} rather than checking for feasibility with $x_{ij_1} = \bar{x}_{ij_1}$. That is, we maximize x_{ij_1} subject to (15) with $\bar{a}_i = a_i$ and $\bar{a}_{i+1} = a_{i+1}$.

- *Case 2a.* The modified subproblem is infeasible. Because feasibility is not restored by reducing x_{ij_1} to zero, we again use the cut (16) and strengthen it by removing some jobs that are not necessary for feasibility.

Suppose now that the modified subproblem is feasible, and let $x_{ij_1}^*$ be the maximum value of x_{ij_1} . We know $x_{ij_1}^* < \bar{x}_{ij_1}$, because otherwise segment i would be feasible.

- *Case 2b.* $x_{ij_1}^* = 0$. Here, job j_1 must be removed as the first job if the other jobs in the segment are completely processed. We have the Benders cut

$$(1 - y_{ij_1}) + \sum_{j \in J_{i0}} (1 - y_{ij0}) \geq 1$$

- *Case 2c.* $x_{ij_1}^* > 0$. Now the Benders cut can say that either $x_{ij_1} \leq \bar{x}_{ij_1}$ or one of the other jobs must be dropped. We introduce 0-1

variable α_i that is 1 when $x_{ij_1} \leq \bar{x}_{ij_1}$. Then we have the cut

$$\begin{aligned} \alpha_i + \sum_{j \in J_{i0}} (1 - y_{ij0}) &\geq 1 \\ x_{ij_1} &\leq \bar{x}_{ij_1} + p_{j_1}(1 - \alpha_i) \\ \alpha_i &\in \{0, 1\} \end{aligned} \tag{17}$$

The cut can be strengthened by removing jobs in J_{i0} until $x_{ij_1}^* \geq \bar{x}_{ij_1}$ (Algorithm 4).

Algorithm 4: Strengthening nogood cuts for the unsegmented feasibility problem.

```

Let  $\bar{J}_{i0} = J_{i0} = \{j_2, \dots, j_k\}$ ;
for  $\ell = 2, \dots, k$  do
    Let  $x_{ij_1}^*$  be the maximum of  $x_{ij_1}$  subject to (15) with  $\bar{a}_i = a_i$  and
     $J_{i0} = \bar{J}_{i0} \setminus \{j_\ell\}$ 
    if  $x_{ij_1}^* < \bar{x}_{ij_1}$  then
        | Remove  $j_\ell$  from  $\bar{J}_{i0}$ 

```

- *Case 3.* There is a partial job only at the end of the segment (say, job j_2). The cuts are similar to Case 2. The modified subproblem finds the maximum value $x_{ij_2}^*$ of x_{ij_2} subject to (15) with $\bar{a}_i = a_i$ and $\bar{a}_{i+1} = a_{i+1}$.
 - *Case 3a.* The modified subproblem is infeasible. We use cut (16) and strengthen it as before.
 - *Case 3b.* $x_{ij_2}^* = 0$. We have the Benders cut

$$(1 - y_{ij_22}) + \sum_{j \in J_{i0}} (1 - y_{ij0}) \geq 1$$

- *Case 3c.* $x_{ij_2}^* > 0$. We have the cut

$$\begin{aligned} \beta_i + \sum_{j \in J_{i0}} (1 - y_{ij0}) &\geq 1 \\ x_{ij_2} &\leq \bar{x}_{ij_2} + p_{j_2}(1 - \beta_i) \\ \beta_i &\in \{0, 1\} \end{aligned} \tag{18}$$

The cut can be strengthened by removing jobs in J_{i0} until $x_{ij_2}^* \geq \bar{x}_{ij_2}$.

- *Case 4.* There are two partial jobs in the segment, namely j_1 at the start and j_2 at the end. We solve a modified subproblem by finding the maximum value x_i^* of $x_{ij_1} + x_{ij_2}$ subject to (15) with $\bar{a}_i = a_i$ and $\bar{a}_{i+1} = a_{i+1}$.
 - *Case 4a.* The modified subproblem is infeasible. We use cut (16) and strengthen it as before.

- *Case 4b.* The modified subproblem is feasible but $x_i^* = 0$. Then j_1 and j_2 must be removed as the first and last jobs if the other jobs are completely processed, and we have the Benders cuts

$$(1 - y_{ij_11}) + \sum_{j \in J_{i0}} (1 - y_{ij0}) \geq 1$$

$$(1 - y_{ij_22}) + \sum_{j \in J_{i0}} (1 - y_{ij0}) \geq 1$$

- *Case 4c.* The modified subproblem is feasible and $0 < x_i^* < \bar{x}_{ij_1} + \bar{x}_{ij_2}$. In this case the Benders cuts says that either (a) $x_{ij_1} + x_{ij_2} \leq x_i^*$, or (b) j_1 or j_2 must be removed as the first or last job if the other jobs are completely processed:

$$\gamma_i + (1 - y_{ij_11}) + (1 - y_{ij_22}) + \sum_{j \in J_{i0}} (1 - y_{ij0}) \geq 1$$

$$x_{ij_1} + x_{ij_2} \leq x_i^* + (p_{j_1} + p_{j_2})(1 - \gamma_i)$$

$$\gamma_i \in \{0, 1\}$$

This cut can be strengthened by removing jobs from J_{i0} until $x_i^* \geq \bar{x}_{ij_1} + \bar{x}_{ij_2}$.

- *Case 4d.* The modified subproblem is feasible and $x_i^* \geq \bar{x}_{ij_1} + \bar{x}_{ij_2}$. Then if $x_{ij_1}^*, x_{ij_2}^*$ are defined as before, we must have either $x_{ij_1}^* < \bar{x}_{ij_1}$ as in Case 2, or $x_{ij_2}^* < \bar{x}_{ij_2}$ as in Case 3. This is because the feasible set for (x_{ij_1}, x_{ij_2}) is a box with the upper right corner possibly cut off with a 45° line. If $x_{ij_1}^* < \bar{x}_{ij_1}$, we add cut (17) as in Case 2c. If $x_{ij_2}^* < \bar{x}_{ij_2}$, we add cut (18) as in Case 3c. Either cut can be strengthened as before.

7 Segmented Makespan Problem

We now address the problem of minimizing makespan when jobs must complete processing within a time segment. The subproblem is itself a minimization problem and therefore generates two types of Benders cuts: strengthened no-good cuts similar to those developed above, and cuts that bound the makespan. The bounding cuts can themselves be based on a no-good principle or a deeper analysis of the inference dual (analytic Benders cuts).

7.1 Master Problem

The master problem for segmented makespan problem is

$$\begin{aligned} & \min M \\ & \sum_i y_{ij} = 1, \quad j \in J \\ & \text{Benders cuts} \\ & \text{Relaxation} \\ & y_{ij} \in \{0, 1\}, \quad \text{all } i, j \end{aligned}$$

Given a solution \bar{y}_{ij} of the master problem, the subproblem decomposes into a minimum makespan problem for each segment i . It minimizes M_i subject to (5.1) and

$$M_i \geq s_j + p_j, \quad \text{all } j \in J \text{ with } \bar{y}_{ij} = 1$$

Thus the makespan on a segment is understood to be the completion time of the last job on the segment, not the completion time minus a_i . If M_i^* is the minimum makespan on segment i , the optimal value of the original problem is $\max_{i \in I} M_i^*$. Note that if no jobs are assigned to segment i , the constraint set is empty, and $M_i^* = -\infty$. Obviously, the latest segment that contains one or more jobs controls the overall makespan.

The relaxation described for the feasibility problem is also valid for the makespan problem. In addition, we can give the following bound on the makespan for each distinct r_j :

$$M \geq \tilde{r}_{ij} + \sum_{\ell \in J(r_j, \infty)} x_{i\ell}, \quad i \in I, j \in J$$

Algorithm 5 generates a relaxation with fewer redundant constraints.

Algorithm 5: Generating a relaxation for the makespan problem.

```

Let  $\bar{r}_1, \dots, \bar{r}_t$  be the distinct elements of  $\{r_1, \dots, r_n\}$ ;
for all  $i$  do
    for all  $j = 1, \dots, t$  with  $a_i < \bar{r}_j$  do
         $\lfloor$  Generate the inequality  $M \geq \bar{r}_j + \sum_{\ell \in J(\bar{r}_j, \infty)} x_{i\ell}$ 
    Generate the inequality  $M \geq w_i + \sum_{\ell \in J} x_{i\ell}$ ;
    for  $j = 1, \dots, n$  do
         $\lfloor$  Generate the inequality  $w_i \geq a_i y_{ij}$ 

```

7.2 Strengthened Nogood Cuts

If the scheduling problem on segment i is infeasible, we use the same strengthened nogood cuts as in the segmented feasibility problem. If segment i has a

feasible schedule with minimum makespan M_i^* , the simplest nogood cut is

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

This says that the makespan for subproblem i cannot be less than M_i^* unless at least one job is removed from J_i . The cut can be strengthened to

$$M \geq M_i^* \left(1 - \sum_{j \in \bar{J}_i} (1 - y_{ij}) \right) \quad (19)$$

where $\bar{J}_i \subset J_i$ is a smaller set of jobs that results in the same minimum makespan M_i^* . A simple heuristic for computing \bar{J}_i appears as Algorithm 6. The jobs can be removed in order of increasing $\tilde{r}_{ij} + p_j$, and we therefore index the jobs in \bar{J}_i so that

$$\tilde{r}_{i1} + p_1 \leq \dots \leq \tilde{r}_{ik} + p_k$$

Algorithm 6: Strengthening nogood cuts for the makespan problem.

```

Let  $\bar{J}_i = J_i = \{j_1, \dots, j_k\}$ ;
for  $\ell = 1, \dots, k$  do
    if the minimum makespan is  $M_i^*$  when jobs in  $\bar{J}_i \setminus \{j_\ell\}$  are assigned to
    segment  $i$  then
         $\bar{J}_i$  Remove  $j_\ell$  from  $\bar{J}_i$ 

```

Another way to obtain additional cuts is to use a two-tiered bound. Let $M_i(J)$ be the minimum makespan that results when jobs in J are assigned to segment i , so that in particular $M_i(J_i) = M_i^*$. 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

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

in addition to (19). This cut is redundant and should be deleted when $M_i(J_i \setminus Z_i) = M_i^*$.

7.3 Analytic Benders Cuts

The reasoning we use to obtain analytic Benders cuts is similar to that used in [26]. Let P_i be the minimum makespan problem on segment i , with minimum

makespan M_i^* . Let $J'_i = \{j \in J_i \mid r_j \leq a_i\}$ be the set of jobs in J_i with release times before segment i , and let $J''_i = J_i \setminus J'_i$. Suppose we remove the jobs in $S \subset J'_i$ from segment i and let \hat{M}_i be the minimum makespan of the problem \hat{P}_i that remains. We first show that

$$M_i^* - \hat{M}_i \leq p_S + \max_{j \in J'_i} \{\tilde{d}_{ij}\} - \min_{j \in J'_i} \{\tilde{d}_{ij}\} \quad (20)$$

where $p_S = \sum_{j \in S} p_j$.

Consider any optimal solution of \hat{P}_i and extend it to a solution s of P_i by scheduling the tasks in S sequentially after \hat{M}_i . Because $S \subset J'_i$, these jobs start after their release time. The makespan of s is $\hat{M}_i + p_S$. If $\hat{M}_i + p_S \leq \min_{j \in J'_i} \{\tilde{d}_{ij}\}$, then s is clearly feasible for P_i , which means $M_i^* \leq \hat{M}_i + p_S$ and (20) follows. On the other hand, if $\hat{M}_i + p_S > \min_{j \in J'_i} \{\tilde{d}_{ij}\}$, we have

$$\hat{M}_i + p_S + \max_{j \in J'_i} \{d_j\} - \min_{j \in J'_i} \{d_j\} > \max_{j \in J'_i} \{d_j\} \quad (21)$$

Because $M_i^* \leq \max_{j \in J'_i} \{d_j\}$, (21) implies (20).

Thus if the jobs in $S \subset J'_i$ are removed from segment i , we have from (20) a lower bound on the resulting optimal makespan \hat{M}_i . If one or more jobs in J''_i are removed, this bound is no longer valid. We have the following Benders cut

$$M \geq M_i^* - \left(\sum_{j \in J'_i} p_j(1 - y_{ij}) + \max_{j \in J'_i} \{d_j\} - \min_{j \in J'_i} \{d_j\} \right) - M_i^* \sum_{j \in J''_i} (1 - y_{ij}) \quad (22)$$

when one or more jobs are removed from segment i , and $M \geq M_i^*$ otherwise, provided $y_{ij} = 1$ for at least one $j \in J_i$. The second summation in (22) takes care of the case where one or more jobs in J''_i are removed. If $y_{ij} = 0$ for all $j \in J_i$, the cut is simply $M \geq 0$.

We can linearize this cut by writing the following for each i :

$$\begin{aligned} M &\geq M_i^* - \sum_{j \in J'_i} p_j(1 - y_{ij}) - w_i - M_i^* \sum_{j \in J''_i} (1 - y_{ij}) - M_i^* q_i \\ q_i &\leq 1 - y_{ij}, \quad j \in J_i \\ w_i &\leq \left(\max_{j \in J'_i} \{d_j\} - \min_{j \in J'_i} \{d_j\} \right) \sum_{j \in J'_i} (1 - y_{ij}) \\ w_i &\leq \max_{j \in J'_i} \{d_j\} - \min_{j \in J'_i} \{d_j\} \end{aligned}$$

where binary variable $q_i = 1$ when $y_{ij} = 0$ for all $j \in J_i$. The cut can be strengthened by replacing J_i , J'_i and J''_i with \bar{J}_i , \bar{J}'_i , and \bar{J}''_i , where \bar{J}_i is computed as in Algorithm 6, $\bar{J}'_i = \{j \in \bar{J}_i \mid r_j \leq a_i\}$, and $\bar{J}''_i = \bar{J}_i \setminus \bar{J}'_i$.

8 Unsegmented Makespan Problem

The master problem for the unsegmented makespan problem minimizes makespan M subject to (12)–(14), Benders cuts, and a relaxation. The subproblem on each segment i minimizes makespan M_i subject to (15) and

$$\begin{aligned} M_i &\geq s_j + p_j, \quad \text{all } j \in J \text{ with } \bar{y}_{ij0} = 1 \\ M_i &\geq a_i + \bar{x}_{ij}, \quad \text{all } j \in J \text{ with } \bar{y}_{ij1} = 1 \\ M_i &\geq a_{i+1}, \quad \text{if } \bar{y}_{ij2} = 1 \text{ or } \bar{y}_{ij3} = 1 \text{ for some } j \in J \end{aligned}$$

The relaxation is the same as for the segmented makespan problem.

8.1 Strengthened Nogood Cuts

If the scheduling problem on segment i is infeasible, we generate the same strengthened nogood cuts as in the unsegmented feasibility problem. Suppose, then, that segment i has a feasible schedule with minimum makespan M_i^* . As before, we let $J_i = \{j \mid \bar{y}_{ij} = 1\}$ and $J_{i0} = \{j \mid \bar{y}_{ij0} = 1\}$.

- *Case 1.* There are no partial jobs in segment i . Then we have the nogood cut

$$M \geq M_i^* \left(1 - \sum_{j \in J_i} (1 - y_{ij0}) \right)$$

This cut can be strengthened as in the segmented problem. We also have the cut

$$M \geq M_i(J_i \setminus Z_i) \left(1 - \sum_{j \in J_i \setminus Z_i} (1 - y_{ij0}) \right)$$

where Z_i is computed as before.

- *Case 2.* A partial job j_2 is assigned to the end of the segment. In this case the minimum makespan on the segment is $M_i^* = a_{i+1}$ unless j_2 is removed from the end. The Benders cut is simply

$$M \geq M_i^* y_{ij_2 2} \tag{23}$$

- *Case 3.* There is no partial job at the end of the segment but a partial job j_1 at the start. The nogood cut is

$$M \geq M_i^* \left(1 - (1 - y_{ij_1 1}) - \sum_{j \in J_{i0}} (1 - y_{ij0}) \right)$$

which can be strengthened by heuristically removing jobs from J_{i0} .

8.2 Analytic Benders Cuts

To analyze the subproblem more deeply we partition J_{i0} as follows:

$$\begin{aligned} J'_{i0} &= \{j \in J_{i0} \mid r_j \leq a_i\} \\ J''_{i0} &= \{j \in J_{i0} \mid r_j > a_i\} \end{aligned}$$

We consider the same three cases as above.

- *Case 1.* There are no partial jobs in segment i . The nogood cut is very similar to that obtained for the segmented case:

$$\begin{aligned} M &\geq M_i^* - \sum_{j \in J'_{i0}} p_j (1 - y_{ij0}) - w_i - M_i^* \sum_{j \in J''_{i0}} (1 - y_{ij0}) - M_i^* q_i \\ q_i &\leq 1 - y_{ij0}, \quad j \in J_{i0} \\ w_i &\leq \left(\max_{j \in J'_{i0}} \{d_j\} - \min_{j \in J''_{i0}} \{d_j\} \right) \sum_{j \in J'_{i0}} (1 - y_{ij0}) \\ w_i &\leq \max_{j \in J'_{i0}} \{d_j\} - \min_{j \in J''_{i0}} \{d_j\} \end{aligned}$$

The cut can be strengthened as before.

- *Case 2.* This yields only the nogood cut (23).
- *Case 3.* There is no partial job at the end of the segment but there is a partial job j_1 at the start of the segment. We will investigate the effect on makespan of reducing job j_1 's processing time in segment i below its current assignment \bar{x}_{ij_1} . Let $\delta = \bar{x}_{ij_1} - x_{ij_1}$ be the amount of the reduction. The Benders cuts generated depend on whether one or more jobs start at their release times in the minimum makespan schedule.

- *Case 3a.* Some job $j \in J_i$ starts at its release time in the optimal solution s^* of the subproblem on segment i . That is, $s_j^* = r_j$ for some $j \in J_{i0}$. Let k be the first such job. We may assume that j_1 and all jobs between j_1 and k are scheduled contiguously in s^* (i.e., there is no idle time between them). Now suppose the jobs between j_1 and k are scheduled δ earlier, and δ is increased to the value δ^* at which one of these jobs hits its release time. Thus

$$\delta^* = \min_{j \in J_{i0}} \{s_j^* - \tilde{r}_{ij} \mid s_j^* < s_k\}$$

This increases by δ^* the gap Δ_i between the job k' immediately before k and job k , where $\Delta_i = r_k - s_{k'}^* - p_{k'}$. Suppose further that for a given $\delta \leq \delta^*$, no job after k is short enough to be moved into the gap $\Delta_i + \delta$ while observing its release time. Then we know that the minimum makespan remains at M_i^* when job j_1 's processing time is reduced by δ , assuming no jobs are removed from segment i .

To write the corresponding Benders cut, let p_{\min} be the processing time of the shortest job that can be moved into the gap $\Delta_i + \delta^*$. Thus

$$p_{\min} = \min_{j \in J_{i0}} \{p_j \mid s_j^* > s_k^*, \tilde{r}_{ij} + p_j \leq r_k, p_j \leq \Delta_i + \delta^*\}$$

Then the minimum makespan remains M_i^* if $p_{\min} > \Delta_i + \delta$ (again assuming no jobs are removed from segment i). We introduce a binary variable η_i that is 0 when this inequality is satisfied. This yields the Benders cut

$$\begin{aligned} M &\geq M_i^*(1 - \eta_i) - M_i^* \sum_{j \in J_i} (1 - y_{ij}) \\ \bar{x}_{ij_1} - x_{ij_1} + \Delta_i &\leq p_{\min} + (\bar{x}_{ij_1} + \Delta_i - p_{\min})\eta_i - \epsilon \\ \bar{x}_{ij_1} - x_{ij_1} + \Delta_i &\geq p_{\min} - (p_{j_1} - \bar{x}_{ij_1} + p_{\min} - \Delta_i)(1 - \eta_i) \end{aligned}$$

where $\epsilon > 0$ is a small number. When $\eta_i = 1$, this cut is of no value, but a simpler cut becomes useful:

$$M \geq \left(a_i + \sum_{j \in J_{i0}} p_j \right) \eta_i + x_{ij_1} - M_i^* \sum_{j \in J_i} (1 - y_{ij})$$

It says that the jobs after j_1 can at best be scheduled contiguously.

- *Case 3b.* No job starts at its release time in the optimal solution of the subproblem. That is, $s_j^* > r_j$ for all $j \in J_{i0}$. Thus all jobs are scheduled contiguously. As $\delta = \bar{x}_{ij_1} - x_{ij_1}$ increases, minimum makespan decreases by an equal amount, at least until a job in J_{i0} hits its release time. At this point we can increase δ by another ϵ and check whether minimum makespan continues to decrease by ϵ . If so, we can further increase δ until another job hits its release time, and so forth until makespan stops decreasing at the margin, at which point we revert to Case 3a. We now write a Benders cut that allows makespan to decrease at the same rate δ increases up to $\delta \leq \delta_{k^*}$.

To make this more precise, let $s^*(\delta)$ be the minimum makespan solution on segment i when x_{ij_1} is fixed to $\bar{x}_{ij_1} - \delta$ rather than \bar{x}_{ij_1} , and let $M_i^*(\delta)$ be the corresponding minimum makespan. Let δ_0 be the value of δ at which the first job hits its release time, so that

$$\delta_0 = \min_{j \in J_{i0}} \{s^*(0) - \tilde{r}_{ij}\}$$

Now define

$$\delta_k = \delta_{k-1} + \min_{i \in J_{i0}} \{s^*(\delta_{k-1}) - \tilde{r}_{ij}\}$$

and let k^* be the smallest k for which $M_i^*(\delta_k + \epsilon) = M_i^*(\delta)$. Then we generate the Benders cut

$$\begin{aligned} \bar{x}_{ij_1} - x_{ij_1} &\leq \delta_{k^*} + (\bar{x}_{ij_1} - \delta_{k^*})\lambda_i \\ M &\geq M_i^*(1 - \lambda_i) - (\bar{x}_{ij_1} - x_{ij_1}) - M_i^* \sum_{j \in J_i} (1 - y_{ij}) \end{aligned}$$

where $\lambda_i \in \{0, 1\}$. This cut is useless when $\lambda_i = 1$, but the following cut is helpful in this case:

$$M \geq a_i \lambda_i + x_{ij_1} + \sum_{j \in J_{i_0}} p_j \lambda_i - M_i^* \sum_{j \in J_i} (1 - y_{ij})$$

After generating these cuts, we move to Case 3a.

9 Segmented Tardiness Problem

The objective in the tardiness problem is to minimize total tardiness. We study the segmented version of the problem. Analysis of the unsegmented tardiness problem is more complex and is left to future research.

9.1 Master Problem

The master problem for the segmented tardiness problem is

$$\begin{aligned} & \min \sum_{i \in I} T_i \\ & T_i \geq 0, \quad i \in I \\ & \sum_i y_{ij} = 1, \quad j \in J \\ & \text{Benders cuts} \\ & \text{Relaxation} \\ & y_{ij} \in \{0, 1\}, \quad \text{all } i, j \end{aligned}$$

Given a solution \bar{y}_{ij} of the master problem, the subproblem for each segment i is:

$$\begin{aligned} & \min \sum_{j \in J_i} \max\{s_j + p_j - d_j, 0\} \\ & \left. \begin{aligned} & r_j \leq s_j \\ & a_i \leq s_j \leq a_{i+1} - p_j \end{aligned} \right\} \quad j \in J_i \\ & \text{noOverlap}(s(i, \bar{y}), p(i, \bar{y})) \end{aligned} \tag{24}$$

The relaxation for the feasibility problem must be modified to make it valid for the tardiness problem, because deadlines are now due dates. The only hard deadlines are the upper bounds a_{i+1} of the segments. We have the relaxation

$$\sum_{\ell \in J(r_j, \infty)} p_\ell y_{i\ell} \leq a_{i+1} - \tilde{r}_{ij}, \quad i \in I, \quad j \in J$$

Algorithm 7 generates a relaxation with fewer redundant constraints.

Algorithm 7: Generating a relaxation for the tardiness problem.

Let $\bar{r}_1, \dots, \bar{r}_t$ be the distinct elements of $\{r_1, \dots, r_n\}$;
for all i **do**
 for all $j = 1, \dots, t$ with $a_i < \bar{r}_j$ **do**
 Generate the inequality $\sum_{\ell \in J(\bar{r}_j, \infty)} p_\ell y_{i\ell} \leq a_{i+1} - \bar{r}_j$
 Generate the inequality $\sum_{\ell} p_\ell y_{i\ell} \leq a_{i+1} - a_i$

We also use a bound on tardiness that is developed in [26].

$$T \geq \sum_{i \in I} \underline{T}_i$$

$$\underline{T}_i \geq \sum_{j \in J} T'_{ij}, \quad i \in I$$

$$T'_{ij} \geq a_i + \sum_{\ell=1}^j p_{\pi(\ell)} y_{ij} - d_j - (1 - y_{ij}) \left(a_i + \sum_{\ell \in H_{ij}} p_{\pi(\ell)} - d_j \right), \quad i \in I, j \in J$$

$$T'_{ij} \geq 0, \quad \text{all } i, j$$

Here $H_{ij} = \{1, \dots, j\} \setminus (J(0, a_i) \cup J(a_{i+1}, \infty))$ is the set of jobs with time windows that overlap segment i . The jobs are indexed so that $d_1 \leq \dots \leq d_n$, and π is a permutation of the indices for which $p_{\pi(1)} \leq \dots \leq p_{\pi(n)}$.

9.2 Strengthened Nogood Cuts

If the tardiness problem on segment i is infeasible, we use the same strengthened nogood cuts as in the feasibility problem. Otherwise, let T_i^* be the minimum tardiness. If $T_i^* > 0$, we use the strengthened nogood cut

$$T_i \geq T_i^* \left(1 - \sum_{j \in \bar{J}_i} (1 - y_{ij}) \right) \quad (25)$$

where $\bar{J}_i \subset J_i$ is a smaller set of jobs that result in the same minimum tardiness T_i^* . A simple heuristic for computing \bar{J}_i appears as Algorithm 8. The jobs can be removed in order of decreasing tightness ϵ_{ij} , but note that ϵ_{ij} can be negative.

Algorithm 8: Strengthening nogood cuts for the tardiness problem.

Let $\bar{J}_i = J_i = \{j_1, \dots, j_k\}$;
for $\ell = 1, \dots, k$ **do**
 if the minimum tardiness is T_i^* when jobs in $\bar{J}_i \setminus \{j_\ell\}$ are assigned to segment i **then**
 Remove j_ℓ from \bar{J}_i

Another way to strengthen the cuts is to use a two-tiered bound similar to the makespan case. Let $T_i(J)$ be the minimum makespan that results when jobs in J are assigned to segment i . If we define

$$Z_i = \{j \in J_i \mid T_i(J_i \setminus \{j\}) = T_i^*\}$$

then we have the cut

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

in addition to (25). This cut is redundant and should be deleted when $T_i(J_i \setminus Z_i) = T_i^*$.

9.3 Analytic Benders Cuts

Let P_i be the subproblem on a segment i with a feasible schedule, and let T_i^* be the minimum tardiness of P_i . When $T_i^* > 0$, we generate an analytic Benders cut as follows. Let \hat{P}_i be the minimum tardiness problem that results when the jobs in $S \subset J_i$ are removed from P_i . Let \hat{T}_i be the minimum tardiness of \hat{P}_i , and \hat{F}_i a solution of \hat{P}_i that achieves tardiness \hat{T}_i . Let

$$r_i^{\max} = \max \{ \max \{ r_j \mid j \in J_i \}, a_i \}$$

be the last release time of the jobs in J_i , or a_i , whichever is larger. Because P_i is feasible, we know $r_i^{\max} \leq a_{i+1}$.

Let \hat{M}_i be the makespan of an optimal solution of \hat{P}_i . We construct a solution F_i of P_i by adding the jobs in S to \hat{F}_i . In particular, we schedule the jobs in S contiguously after $\max \{ r_i^{\max}, \hat{M}_i \}$, in arbitrary order. The makespan of F_i is at most

$$r_i^{\max} + \sum_{\ell \in J_i} p_\ell$$

because in the worst case, all the jobs in the optimal solution of \hat{P}_i are scheduled after r_i^{\max} . The tardiness incurred in F_i by each job $j \in S$ is therefore at most

$$\left(r_i^{\max} + \sum_{\ell \in J_i} p_\ell - d_j \right)^+$$

where $(\alpha)^+ = \max\{\alpha, 0\}$. Thus the total tardiness of F_i is at most

$$\hat{T}_i + \sum_{j \in S} \left(r_i^{\max} + \sum_{\ell \in J_i} p_\ell - d_j \right)^+ \quad (26)$$

F_i is feasible if all the jobs finish before a_{i+1} . So we know F_i is feasible if

$$r_i^{\max} + \sum_{\ell \in J_i} p_\ell \leq a_{i+1} \quad (27)$$

In this case, the tardiness (26) is an upper bound on T_i^* , and we have

$$\hat{T}_i \geq T_i^* - \sum_{j \in S} \left(r_i^{\max} + \sum_{\ell \in J_i} p_\ell - d_j \right)^+$$

This leads to the Benders cut

$$\hat{T}_i \geq \begin{cases} T_i^* - \sum_{j \in J_i} \left(r_i^{\max} + \sum_{\ell \in J_i} p_\ell - d_j \right)^+ (1 - y_{ij}), & \text{if (27) holds} \\ T_i^* \left(1 - \sum_{j \in J_i} (1 - y_{ij}) \right), & \text{otherwise} \end{cases}$$

10 Problem Generation

We generated random instances by selecting parameters uniformly from intervals as follows:

$$\begin{aligned} r_j &\in [0, \alpha R] \\ d_j - r_j &\in [\gamma_1 \alpha R, \gamma_2 \alpha R] \\ p_j &\in [0, \beta(d_j - r_j)] \end{aligned}$$

where R is the length of the time horizon, measured by the number of segments. Thus γ_1 and γ_2 control the width of the time windows, and β controls the processing time relative to the window width.

We set parameters as indicated in Table 3. We distinguished tight from wide time windows for segmented problems, because wider windows could result in less effective propagation and/or relaxations. The remaining parameters were chosen to obtain instances that are (a) nontrivial to solve and (b) usually feasible for the optimization problems (less often feasible for the feasibility problems). To accomplish this, we adjusted β and γ_2 empirically to sample instances near a phase transition where average problem difficulty peaks and there is a mix of feasible and infeasible instances. This required adjusting β to different values as the problem scaled up.

11 Computational Results

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. We used OPL's script language to implement the Benders method. We generated 10 instances for each problem size and type, for a total of 580 instances.

Tables 4 and 5 display computational results for the segmented problems. The advantage of logic-based Benders increases rapidly as the problem scales up, relative to both CP and MILP. The Benders method failed to solve only 4 of 420 instances within ten minutes, while CP failed to solve 247 and MILP failed to solve 113.

Table 3: Parameters for generation of problem instances.

	<i>Segmented Problems</i>			<i>Unsegmented</i>
	<i>Feasibility</i>	<i>Makespan</i>	<i>Tardiness</i>	<i>Feas. & Makespan</i>
α	0.5			
γ_1	0.5 for tight windows 0.25 for wide windows			0.25
γ_2	1.0			
β	Tight windows: 1/20 for 6–8 segs. 1/24 for 10 segs. 1/28 for 12 segs. 1/32 for 14 segs. Wide windows: 0.035	0.025 for ≤ 13 segs. 0.032 for > 13 segs.	0.05	1/15 for 5-8 segs. 1/20 for 9-12 segs. 1/30 for 12-16 segs. 1/35 for >16 segs.

Table 4: Computation times in seconds for the segmented problem with tight time windows. The number of segments is 10% the number of jobs. Ten instances of each size are solved.

Jobs	Feasibility			Makespan			Tardiness		
	CP	MILP	Bndrs	CP	MILP	Bndrs	CP	MILP	Bndrs
60	0.1	14	1.9	60	7.7	6.4	0.1	16	3.0
80	181*	45	2.7	420*	147	11	63*	471*	20
100	199*	58	4.3	600*	600	17	547*	177*	11
120	272*	137	4.8	600*	600	39	600*	217*	2.9
140	306*	260*	6.8	600*	432*†	33	600*	373*	5.0
160	314*	301*	8.0	600*	359*	14			
180	600*	350*†	4.8	600*	557*†	5.3			
200	600*	†	5.8	600*	600*†	6.6			

*Solution terminated at 600 seconds for some or all instances.

†MILP solver ran out of memory for some or all instances, which are omitted from the average solution time.

Table 6 displays computational results for the unsegmented problems. The Benders method continues to have a substantial advantage over MILP, but it is considerably slower than CP on the easier problems. However, examination of the individual instances reveals that the Benders method is more robust. The Benders method solved all 160 unsegmented instances, while CP failed to solve 20 instances within ten minutes. CP was very fast for the instances it solved (average of 0.79 seconds), but Benders solved the remaining instances in an average of only 5.94 seconds. This suggests that one ought to try CP first, and if it fails to solve the problem in a few seconds, switch to Benders.

The volatility of CP may be due to the fact that filtering and bounds propagation can be effective on a long time horizon when time windows interact in a certain way, but when this does not occur, a huge search tree is generated. This

Table 5: Average computation times in seconds for the segmented problem with wide time windows. The number of segments is 10% the number of jobs. Ten instances of each size are solved.

Jobs	Feasibility			Makespan			Tardiness		
	CP	MILP	Bndrs	CP	MILP	Bndrs	CP	MILP	Bndrs
60	0.05	12	1.9	0.2	16	5.8	0.2	8.0	2.3
80	0.28	22	2.5	180*	59	9.0	1.5	94	3.7
100	0.14	37	3.8	360*	403*	14	79*	594*	85*
120	0.13	61	5.0	540*	600*	25	600*	251*	183*
140	61*	175	7.0	600*	600*	107	600*	160*	4.3
160	540*	216*	4.8	600*	562*	157			
180	600*	375* [†]	4.5	600*	535*	10			
200	600*	†	5.5	600*	560*	6.9			

*Solution terminated at 600 seconds for some or all instances.

[†]MILP solver ran out of memory for some or all instances, which are omitted from the average solution time.

Table 6: Average computation times in seconds for the unsegmented problem. The number of segments is 10% the number of jobs. Ten instances of each size are solved,

Jobs	Feasibility			Makespan		
	CP	MILP	Bndrs	CP	MILP	Bndrs
60	0.10	11	2.8	0.2	24	5.1
80	0.14	21	3.7	0.7	376*	8.7
100	0.25	35	7.0	1.1	600*	21
120	0.43	57	23	0.4	600*	93
140	0.72	97	65	1.2	600*	115
160	420*	188	9.0	241*	549*	67
180	123*	307*	79	61*	600*	168
200	180*	410*	29	180*	587*	21

*Solution terminated at 600 seconds for some or all instances.

phenomenon may not affect the Benders method because the scheduling segments are small enough to result in limited search trees even when propagation is ineffective.

We also investigated the effectiveness of analytic Benders cuts. They incur greater overhead than nogood cuts, because each analytic cut requires multiple inequalities in the master problem. This could offset faster convergence. To test this hypothesis, we re-solved all the instances with nogood cuts but without analytic cuts (Table 7). We found that for segmented problems, the analytic cuts make little difference on the average when time windows are narrow. However, they bring significant and occasionally dramatic reductions in computation time for wide time windows. Because these cuts do no harm (on the average) in either case and are advantageous for wider time windows, it is advisable to use them.

Table 7: Effect of analytic Benders cuts on computation time. The last three columns show the percent of instances in which analytic cuts reduced computation time by more than the stated amount.

Problem class	% reduction		% of instances with reduction		
	Average	Maximum	> 0%	> 20%	> 50%
Segmented makespan:					
tight windows	0	45	46	14	0
wide windows	12	85	79	46	11
Segmented tardiness:					
tight windows	-4*	37	60	6	0
wide windows	7	99	62	36	8
Unsegmented:					
makespan	12	64	76	59	8

*Reflects three very negative outliers.

As for unsegmented problems, the analytic cuts are clearly beneficial and should be used.

12 Conclusion

We adapted logic-based Benders decomposition to a pure scheduling problem that lacks the natural decomposability of the planning and scheduling problems to which the method has been previously applied. The master problem assigns jobs to segments of the time horizon rather than to machines or other resources.

We find that for single-facility scheduling, logic-based Benders scales up more effectively than state-of-the-art CP and MILP solvers. This is especially true for the segmented problem, in which jobs are not permitted to overlap segment boundaries. The Benders method solves much larger instances of the feasibility and makespan problems, and its speed advantage increases rapidly as the problem size increases. It is somewhat faster on the tardiness problem.

The Benders master problem becomes more complex for the unsegmented problem, in which jobs may overlap segment boundaries. Benders decomposition continues to dominate MILP while being much slower than CP on most of the smaller instances. However, CP begins to lose its ability to solve instances as they scale up, whereas Benders continues to solve them, usually in a few seconds. Benders is therefore not necessarily the fastest method but clearly the most robust.

CP solves unsegmented instances quickly if it solves them at all. This suggests a strategy of applying CP first, and if it fails to solve the problem with a few seconds, switch to Benders. For segmented instances, Benders is always superior and should be used from the start.

Possible future research includes the development of Benders cuts for the unsegmented tardiness problem. In addition, convergence might be accelerated with the generation of multiple cuts, or by a “warm start” that adds a collection

of Benders cuts to the initial master problem, as in [1, 33]. The length of time segments might be adjusted dynamically for better performance. Finally, other forms of decomposition can be explored.

References

- [1] A. Aggoun and A. Vazacopoulos. Solving sports scheduling and timetabling problems with constraint programming. In S. Butenko, J. Gil-Lafuente, and P. M. Pardalos, editors, *Economics, Management and Optimization in Sports*, pages 243–264. Springer, New York, 2004.
- [2] F. Babonneau, C. Beltran, A. Haurie, C. Tadonki, and J. P. Vial. Proximal-ACCPM: A versatile oracle based optimization method. In E. J. Kon- toghiorghes and C. Gatu, editors, *Optimisation, Econometric and Financial Analysis*, volume 9 of *Advances in Computational Management Science*, pages 69–92. Springer, New York, 2007.
- [3] P. Baptiste, C. Le Pape, and W. Nuijten. *Constraint-Based Scheduling: Applying Constraint Programming to Scheduling Problems*. Kluwer, Dor- drecht, 2001.
- [4] A. Y. Barlatta, A. M. Cohn, and O. Gusikhinc. A hybridization of mathe- matical programming and dominance-driven enumeration for solving shift- selection and task-sequencing problems. *Computers and Operations Re- search*, 37:1298–1307, 2010.
- [5] J. F. Benders. Partitioning procedures for solving mixed-variables pro- gramming problems. *Numerische Mathematik*, 4:238–252, 1962.
- [6] L. Benini, D. Bertozzi, A. Guerri, and M. Milano. Allocation and scheduling for MPSoCs via decomposition and no-good generation. In *Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *Lecture Notes in Computer Science*, pages 107–121. Springer, 2005.
- [7] R. Bent and P. Van Hentenryck. Spatial, temporal, and hybrid decom- positions for large-scale vehicle routing with time windows. In D. Cohen, editor, *Principles and Practice of Constraint Programming (CP 2010)*, vol- ume 6308 of *Lecture Notes in Computer Science*, pages 99–113. Springer, 2010.
- [8] H. Cambazard, P.-E. Hladik, A.-M. Déplanche, N. Jussien, and Y. Trinquet. Decomposition and learning for a hard real time task allocation problem. In M. Wallace, editor, *Principles and Practice of Constraint Programming (CP 2004)*, volume 3258 of *Lecture Notes in Computer Science*, pages 153– 167. Springer, 2004.
- [9] Y. Chu and Q. Xia. Generating Benders cuts for a class of integer pro- gramming problems. In J. C. Régim and M. Rueher, editors, *Integration*

- of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004), volume 3011 of *Lecture Notes in Computer Science*, pages 127–141. Springer, 2004.
- [10] G. Codato and M. Fischetti. Combinatorial Benders’ cuts for mixed-integer linear programming. *Operations Research*, 54:756–766, 2006.
 - [11] A. I. Corréa, A. Langevin, and L. M. Rousseau. Dispatching and conflict-free routing of automated guided vehicles: A hybrid approach combining constraint programming and mixed integer programming. In J. C. Régim and M. Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, volume 3011 of *Lecture Notes in Computer Science*, pages 370–378. Springer, 2004.
 - [12] M. M. Fazel-Zarandi and J. C. Beck. Solving a location-allocation problem with logic-based Benders’ decomposition. In I. P. Gent, editor, *Principles and Practice of Constraint Programming (CP 2009)*, volume 5732 of *Lecture Notes in Computer Science*, pages 344–351. Springer, 2009.
 - [13] S. French. *Sequencing and Scheduling*. John Wiley & Sons, 1982.
 - [14] D. R. Fulkerson. Blocking and anti-blocking pairs of polyhedra. *Mathematical Programming*, 1:168–194, 1971.
 - [15] A. M. Geoffrion. Generalized Benders decomposition. *Journal of Optimization Theory and Applications*, 10:237–260, 1972.
 - [16] I. Harjunkoski and I. E. Grossmann. A decomposition approach for the scheduling of a steel plant production. *Computers and Chemical Engineering*, 25:1647–1660, 2001.
 - [17] I. Harjunkoski and I. E. Grossmann. Decomposition techniques for multi-stage scheduling problems using mixed-integer and constraint programming methods. *Computers and Chemical Engineering*, 26:1533–1552, 2002.
 - [18] J. N. Hooker. Logic-based Benders decomposition. In *INFORMS National Meeting (INFORMS 1995)*, 1995.
 - [19] J. N. Hooker. Inference duality as a basis for sensitivity analysis. In E. C. Freuder, editor, *Principles and Practice of Constraint Programming (CP 1996)*, volume 1118 of *Lecture Notes in Computer Science*, pages 224–236. Springer, 1996.
 - [20] J. N. Hooker. *Logic-Based Methods for Optimization: Combining Optimization and Constraint Satisfaction*. Wiley, New York, 2000.
 - [21] J. N. Hooker. A hybrid method for planning and scheduling. In M. Wallace, editor, *Principles and Practice of Constraint Programming (CP 2004)*, volume 3258 of *Lecture Notes in Computer Science*, pages 305–316. Springer, 2004.

- [22] J. N. Hooker. A hybrid method for planning and scheduling. *Constraints*, 10:385–401, 2005.
- [23] J. N. Hooker. Planning and scheduling to minimize tardiness. In *Principles and Practice of Constraint Programming (CP 2005)*, volume 3709 of *Lecture Notes in Computer Science*, pages 314–327. Springer, 2005.
- [24] J. N. Hooker. An integrated method for planning and scheduling to minimize tardiness. *Constraints*, 11:139–157, 2006.
- [25] J. N. Hooker. *Integrated Methods for Optimization*. Springer, 2007.
- [26] J. N. Hooker. Planning and scheduling by logic-based Benders decomposition. *Operations Research*, 55:588–602, 2007.
- [27] J. N. Hooker and G. Ottosson. Logic-based Benders decomposition. *Mathematical Programming*, 96:33–60, 2003.
- [28] J. N. Hooker and H. Yan. Logic circuit verification by Benders decomposition. In V. Saraswat and P. Van Hentenryck, editors, *Principles and Practice of Constraint Programming: The Newport Papers*, pages 267–288, Cambridge, MA, 1995. MIT Press.
- [29] V. Jain and I. E. Grossmann. Algorithms for hybrid MILP/CP models for a class of optimization problems. *INFORMS Journal on Computing*, 13:258–276, 2001.
- [30] R. G. Jeroslow. Representability in mixed integer programming, I: Characterization results. *Discrete Applied Mathematics*, 17:223–243, 1987.
- [31] A. B. Keha, K. Khowala, and J. W. Fowler. Mixed integer programming formulations for single machine scheduling problems. *Computers and Industrial Engineering*, 56:357–367, 2009.
- [32] C. Koulamas. The single-machine total tardiness scheduling problem: Review and extensions. *European Journal of Operational Research*, 202:1–7, 2010.
- [33] C. T. Maravelias. A decomposition framework for the scheduling of single- and multi-stage processes. *Computers and Chemical Engineering*, 30:407–420, 2006.
- [34] 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.
- [35] C. T. Maravelias and I. E. Grossmann. Using MILP and CP for the scheduling of batch chemical processes. In J. C. Régin and M. Rueher, editors, *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems (CPAIOR 2004)*, volume 3011 of *Lecture Notes in Computer Science*, pages 1–20. Springer, 2004.

- [36] M. Padberg. On the facial structure of set packing polyhedra. *Mathematical Programming*, 5:199–215, 1973.
- [37] M. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Prentice Hall, 1995.
- [38] R. Rasmussen. Scheduling a triple round robin tournament for the best Danish soccer league. *European Journal of Operational Research*, 185:795–810, 2008.
- [39] R. Rasmussen and M. A. Trick. A Benders approach to the constrained minimum break problem. *European Journal of Operational Research*, 177:198–213, 2007.
- [40] R. Sadykov and L. A. Wolsey. Integer programming and constraint programming in solving a multimachine assignment scheduling problem with deadlines and release dates. *INFORMS Journal on Computing*, 18:209–217, 2006.
- [41] S. A. Tarim and I. Miguel. A hybrid Benders’ decomposition method for solving stochastic constraint programs with linear recourse. In B. Hnich, M. Carlsson, F. Fages, and F. Rossi, editors, *Recent Advances in Constraints (CSCLP 2005)*, volume 3978 of *Lecture Notes in Computer Science*, pages 133–148. Springer, 2006.
- [42] D. Terekhov, J. C. Beck, and K. N. Brown. Solving a stochastic queueing design and control problem with constraint programming. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2005)*, pages 261–266, 2005.
- [43] E. Thorsteinsson. Branch and check: A hybrid framework integrating mixed integer programming and constraint logic programming. In T. Walsh, editor, *Principles and Practice of Constraint Programming (CP 2001)*, volume 2239 of *Lecture Notes in Computer Science*, pages 16–30. Springer, 2001.
- [44] C. Timpe. Solving planning and scheduling problems with combined integer and constraint programming. *OR Spectrum*, 24:431–448, 2002.